Contents lists available at ScienceDirect

Future Generation Computer Systems

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

MQDS: An energy saving scheduling strategy with diverse QoS constraints towards reconfigurable cloud storage systems

Xindong You [a], Dawei Sun [b,*], Xueqiang Lv [a], Shang Gao [c], Rajkumar Buyya [d]

[a] *Beijing Key Laboratory of Internet Culture and Digital Dissemination Research, Beijing Information Science and Technology University, Beijing, 100101, PR China*
[b] *School of Information Engineering, China University of Geosciences, Beijing, 100083, PR China*
[c] *School of Information Technology, Deakin University, Waurn Ponds, Victoria, 3216, Australia*
[d] *Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, The University of Melbourne, Australia*

A R T I C L E   I N F O

Article history:
Received 16 December 2020
Received in revised form 8 September 2021
Accepted 24 November 2021
Available online 9 December 2021

Keywords:
Reconfigurable disk
Quality of Service
Energy efficiency
Cloud storage
Scheduling algorithms

A B S T R A C T

Nowadays, cloud storage systems are usually constructed by heterogeneous disks with reconfigurable speed for energy efficiency. Scheduling tasks with diverse QoS (Quality of Service) requirements to disks with different speeds is seldom considered in state-of-the-art mainstream disk scheduling algorithms. This paper proposes a multi-QoS disk scheduling strategy (MQDS), aiming to support energy-saving and diverse QoS constraints towards reconfigurable heterogeneous cloud storage systems. Three algorithms are designed in MQDS: Time Prior Disks Algorithm (TPDS) for time-sensitive tasks, Cost Prior Disks Algorithm (CPDS) for cost-sensitive tasks, and Benefit Function-based Disks Algorithm (BFDS) for tasks with time-varying QoS requirements. CloudSimDisk simulator is extended to evaluate the response time, energy consumption and cost expenditure performance of the proposed algorithms, for comparison with the extended Robin Round Disk scheduling algorithm (E-RRDS). Extensive experimental results demonstrate that the three proposed algorithms outperform E-RRDS. TPDS consumes shortest response time and CPDS has the least cost; and BFDS achieves balance between TPDS and CPDS, which provide more options to users. It is worth mentioning that all the algorithms in MQDS are more energy efficient than E-RRDS. As a whole, the proposed MQDS is energy efficient and able to accommodate diverse QoS constraints well.

1. Introduction

Cloud computing offers utility-oriented computing services to users on a pay-as-you-go basis [1,2]. One of the important reasons for its success is that it provides transparent computing and storing services to users with diverse QoS requirements. In cloud computing environments, different users have different QoS requirements. Some tasks are time-sensitive, such as live video and scientific computing tasks. For those time-sensitive tasks, users usually increase the budget in exchange for rapid data access. Some tasks are not time-sensitive but more cost concerned, such as file replication and data archiving tasks. Furthermore, there are tasks switching between time or cost sensitivity in different time intervals. In other words, their QoS requirements are time-varying. As is well-known, how to reasonably schedule resources to satisfy users' diverse QoS is one of the most challenging problems in resource scheduling domain [3–6]. Users with different QoS requirements can be charged for different rates in cloud computing systems. Economic mechanism is an efficient means for managing distributed resources [7–9]. Moreover, the cloud storage systems for energy efficiency are usually constructed by reconfigurable disks [10–12]. The disks with different speeds are with different transfer rates and different energy consuming rates. One the one hand, how to schedule the tasks with diverse QoS requirements to the reconfigurable heterogeneous cloud storage system is becoming more and more important, which was seldom considered in the state-of-the-art mainstream disk scheduling algorithms. On the other hand, the energy consumption of the storage parts in a datacenter is rapidly growing. Many studies show that the energy consumed by the storage parts is exceeding the energy consumed by the computing parts in datacenters [13–17]. Most recently, Ding et al. proposed a dynamic task scheduling method based on Q-learning for energy-efficient cloud computing [18], however, the factors of energy consumption and diverse QoS requirements have not been fully addressed. It is the same in current state-of-the-art mainstream disk scheduling algorithms for cloud storage systems. Under this situation, accommodating diverse QoS requirements to speed reconfigurable disks and achieving energy efficiency is a n urgent

* Corresponding author.
E-mail address: sundaweicn@cugb.edu.cn (D. Sun).

https://doi.org/10.1016/j.future.2021.11.025
0167-739X/© 2021 Elsevier B.V. All rights reserved.

issue to be resolved, not only for improving users' experience, but also for reducing the rapid growing energy consumption of cloud storage.

In this paper, a data scheduling strategy with three different disk scheduling algorithms is designed for a reconfigurable heterogeneous cloud storage system. It is energy efficient and can satisfy diverse QoS requirements. The key contributions of the proposed data scheduling strategy are as follows:

(i) An energy-saving data scheduling strategy with diverse QoS constraints towards reconfigurable heterogeneous disks architecture (MQDS) is designed and implemented, where users can submit different QoS requirements according to the characteristics of their tasks.

(ii) Time Prior Disk scheduling algorithm (TPDS), Cost Prior Disk scheduling algorithm (CPDS) and Benefit Function-based Disk scheduling algorithm (BFDS) are implemented in the MQDS framework, aiming to accommodate diverse QoS requirements and achieve the energy efficiency.

(iii) Effective extensions are implemented in CloudSimDisk simulator. In the extended simulator, tasks with diverse QoS requirements can be scheduled to disks with reconfigurable speeds through the proposed TPDS, CPDS, BFDS and E-RRDS (Extended from the current mainstream disk scheduling algorithm for cloud storage system with speed reconfigurable disks). Furthermore, the traditional performance metric of response time, the new performance metric of energy consumption and cost expenditure can be obtained in the extended simulator.

(iv) Extensive experiments are conducted in the extended CloudSimDisk simulator to evaluate the three proposed disk scheduling algorithms (TPDS, CPDS, BFDS) against the E-RRDS. Driven by the real wiki-workload with 5000 tasks requests with diverse QoS requirements, the obtained simulation experimental results demonstrate that the proposed MQDS is more energy efficient than the E-RRDS while accommodating all the diverse QoS requirements. The simulation experimental results provide valuable references to implement new forms of cloud resource charging with diverse QoS requirements in real cloud environments. Moreover, although the proposed MQDS are evaluated on HDDs (Hard Disk Drive) with reconfigurable speeds, it can be generalized to storage resources with reconfigurable modes, such as SDDs (Solid State Drive) or the emerging NVMs (Non-Volatile Memory) [19,20], as the MQDS is a high level scheduling algorithm abstracted upon the storage resources.

The rest of the paper is organized as follows: Section 2 discusses the related work on task scheduling and disk scheduling. Section 3 describes the proposed scheduling strategy (MQDS) in detail, where the framework of MQDS and the three algorithms are explained. Performance evaluation of MQDS and comparison with the extended Robin Round Disk scheduling algorithm E-RRDS are conducted in Section 4. Finally, our paper is concluded in Section 5 with the future work.

2. Related work

Resource accounting is one of the key tasks for a distributed grid computing system. In cloud computing era, public clouds have been providing services with resource accounting. Services are selected not only depending on the performance and capacity of the cloud system (e.g. computing power, storage and transmission capacity), but also on the cost that users are willing to pay. Users can submit different QoS requirements according to the characteristics of their tasks, aiming to achieve an optimal balance of these arguments. Furthermore, employing an economic mechanism can help achieve the maximum profit for the whole system. To address the time or cost-based computing problems, Rajkumar Buyya research group firstly proposed and designed the scheduling algorithms with time prior or cost prior for grid computing systems. Heterogeneous resources were allocated from the economic perspective in their scheduling algorithms, which was a new form of resource scheduling and achieved good performance [21]. GridSim simulator was designed to evaluate their proposed scheduling algorithms, where the resources were priced. Resources with different computing power or capacity are charged at different rates. Users pay price for the chosen resources. On the one hand, when submitting jobs (Gridlets in the simulator) to their system, parameters of Deadline and Budget should be submitted in the meantime. The submitted jobs must be finished within the Deadline. The overall cost of all the tasks must be under the Budget. On the other hand, users can set the scheduling parameter as OPTIMIZATION_TIME to choose the time prior scheduling algorithm, and set the parameter as OPTIMIZATION_COST to choose the cost prior scheduling algorithm. Time prior or cost prior scheduling algorithm can satisfy users' QoS requirements in some degree.

Time prior scheduling algorithm always performs better than the cost prior algorithm in terms of response time, while the cost prior scheduling algorithm performs better than the time prior algorithm in terms of cost expense. Users can choose the time prior one to execute their emergent jobs, exchanging cost for time. If a job is time-insensitive, exchanging time for cost. Srikumar Venugopa extended the time prior and cost prior scheduling algorithms by considering the position of input data [22]. A resource allocation and science workflow scheduling strategy was also proposed in IaaS cloud [23], where an optimization technology based on meta-heuristic and particle swarm algorithms was used to reduce the execution cost of the whole workflow with the budget and deadline constraints. The implementation process of ARIMA model was described in [24], where the workload of a Web server was predicted by ARIMA. It was verified that ARIMA could improve the resource utilization under QoS constraints.

Moreover, a dynamic pricing based energy cost optimization mechanism was proposed by Wang et al. in 2013 [25], where a uniform relationship model between the service price and energy cost was constructed. It aimed to maximize the profit of the data center, in which users' QoS requirements were not considered. Based on the research of distributed storage systems in cloud computing environments, Liao et al. designed a QoS-aware dynamic replica deleting strategy (DRDS). Experimental results demonstrated that the proposed DRDS could save the disk storage space and distributed storage system maintenance cost while satisfying the user's QoS [26]. However, the energy consumption and dynamic QoS requirements were not considered in DRDS, which are important factors in current Cloud Storage Systems. Considering the dynamic characteristics of computing grid environments, Jiang et al. integrated the Multi-agent system (MAS) cooperative technology and market biding game model to conduct resource scheduling. It was proved that the resource could be allocated reasonably [27]. However, the energy consumption was not the optimization objective in the system. A resource scheduling algorithm (called Senior) with dual constraints of deadline and bandwidth was proposed by Chen et al. The proposed Senior could achieve an optimal resource utilization with QoS constraints [28]. In Senior, dynamic users' QoS requirements were not considered. And the energy consumption was not adopted as the performance index. A SLA-aware resource algorithm was proposed in [33], in which node space utilization and I/O throughput were its optimization objectives. A dynamic management framework for IoT devices in cloud (DMFIC) algorithm was proposed to evaluate and schedule requests and sensor data [34], in which the energy consumption but without diverse QoS requirement was considered. More recent works were also reviewed [35,36], but none of them considered energy consumption or dynamic QoS requirements. The aforementioned resource

Table 1
Comparison between the proposed MQDS and related main work.

Work or Name	Optimizing objective	Task scheduling	Disk scheduling	Deadline and budget constraint	Flexible QoS considered	Disk speed-aware	Energy consumption-aware
Buyya et al [21].	Time or Cost	✔	–	✔	–	–	–
Venugopa et al [22]	Time or Cost	✔	–	✔	–	–	–
Buyya et al [23]	Time	✔	–	✔	–	–	–
ARIMA model was described in [24]	Resource utilization	✔	–	✔	–	–	–
Wang et al in 2013 [25]	Energy Cost	✔	–	✔	–	–	✔
DRDS [26]	Storage space and system maintenance cost	✔	–	–	–	–	–
MAS [27]	Resource utilization	✔	–	–	–	–	–
Senior [28].	Resource utilization	✔	–	–	–	–	–
Minerva [29]	Overall throughput	–	✔	–	–	–	–
Dash et al [30].	Seek time, rotational latency and transfer time	–	✔	–	–	–	–
Sarkar et al [31]	Traditional performance parameters	–	✔	–	–	–	–
RRDS [32]	Response Time	–	✔	–	–	–	–
Our work	Energy consumption and Benefit	–	✔	✔	✔	✔	✔

scheduling algorithms are usually focusing on scheduling the task with different QoS requirements to nodes but not to the disks with reconfigurable speeds. The allocation mechanisms and the description of diverse QoS requirements instead provide valuable references for designing our reconfigurable disk scheduling algorithms, where the QoS information should be perceived at disk level.

Compared with the attention gained in tasks scheduling to nodes, disk scheduling algorithms received less concerns in cloud computing environments [37,38]. According to the QoS in multitier and multitenant cloud environments, Malensek et al. designed a proactive disk scheduling algorithm named Minerva [29]. Minerva is based on predictive model and client-side coordination, to deal with the scheduling problem on high-capacity mechanical disks, which can influence the overall throughput and responsiveness of a cluster in data-intensive computing environments. However, the heterogeneous properties of the cloud storage disks have not been considered. Dash et al. proposed an optimized disk scheduling algorithm in order to manage the bad-sector of a disk, aiming to reduce the seek time, rotational latency and transfer time [30]. The diverse QoS requirements and the disks' heterogeneities in cloud environments were not considered in their disk scheduling algorithm. Sarkar et al. used disk scheduling technique to enhance the performance of cloud based storage [31], which optimized the traditional performance parameters. Known from our investigation, the heterogeneities of disks have not been fully considered when scheduling tasks to disks. The mainstream disk scheduling algorithm used in current cloud storage platforms is the Robin Round Disk scheduling (RRDS) [32], which is disk speed aware. In this paper, we extend RRDS to E-RRDS, with both the diverse QoS requirements and disks' speed reconfigurable properties considered. E-RRDS is chosen as the baseline to evaluate our proposed disk scheduling algorithms TPDS, CPDS and BFDS.

As a whole, compared with the existing task scheduling algorithms and resource allocation strategies, the disk scheduling strategy with diverse QoS constraints (MQDS) proposed in this paper has the following distinct differences.

(i) In previous research, computing resources were usually the main factor to be considered in QoS constrained resource allocation. Some researchers focused on the storage resource allocation, but only the storage capacity was considered. In our proposed MQDS, the rotation rate of the disk, the transfer rate and energy consumption rate of the disk in heterogeneous cloud computing environments are also considered, besides of these aforementioned factors.

(ii) Users' QoS requirements are represented in a more flexible manner in our MQDS framework. Besides of the time prior or cost prior QoS requirements, a benefit function is constructed to profile the time-varying QoS requirements, expressing users' actual demands more preciously.

(iii) Traditional system performance metrics such as average response time are the main optimization objectives of current QoS constraint related research. While constructing scheduling algorithms, the reconfigurable property of the storage resources is considered in the proposed MQDS, aiming to reduce the energy consumption while satisfying the diverse QoS requirements, which is particularly important for cloud storage systems.

Furthermore, the comparison between the proposed MQDS and related main work is illustrated in Table 1.

3. MQDS: Multi-QoS Disk scheduling strategy

The Energy-saving scheduling strategy with diverse QoS constraints (MQDS) proposed in this paper aims to reduce the energy consumption of the cloud storage systems consisting of reconfigurable disks. Furthermore, it schedules tasks to disks in different running modes to satisfy users' diverse QoS requirements. The proposed framework, procedure and algorithms will be introduced in the following subsections.

3.1. Terminologies and definitions

To help understand the following subsections, important terms and definitions used are listed in Table 2.

Table 2
Terminologies and definitions.

Terms	Definitions
Budget	Cost that can be spent by a user to finish their data request tasks. It is usually measured in units $. Budget is one of the important constraint conditions for a multi-speed scheduler. If the cost spent on one certain disk exceeds the Budget, the scheduler will not schedule tasks to that disk.
Deadline	By the time that a user's tasks must be finished. It is usually measured in unit of seconds. If the completion time exceeds the deadline, the tasks will not be scheduled to the disk by the scheduler.
Time Prior OoS requirement	Short for TP, designed for time-sensitive tasks. The scheduler schedules tasks to disks with the fastest speed under the Budget constraint.
Cost Prior OoS requirement	Short for CP, designed for time-insensitive but cost-sensitive tasks. The scheduler schedules tasks to disks with the lowest cost under the Deadline constraint.
Benefit Function-based QoS requirement	Short for BFB, designed for tasks with dynamic QoS requirements. The QoS requirements are encapsulated in a benefit function. The multi-speed scheduler schedules tasks to disks according to the benefit function, where the benefit is time-varying. Under this condition, our scheduler tries to maximize the benefit under the dual constraints of Budget and Deadline.

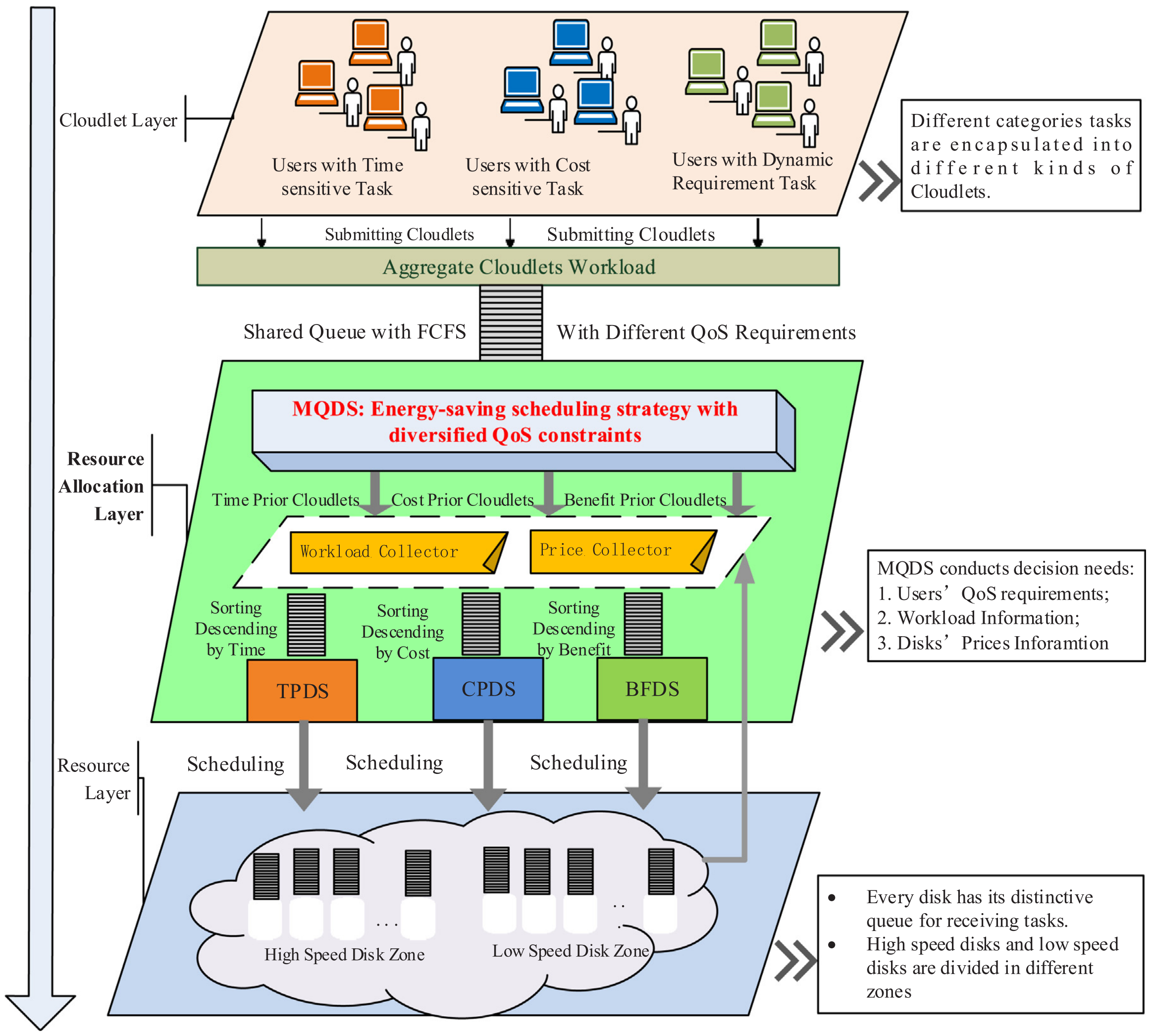
Cloudlet Layer
Users with Time sensitive Task
Users with Cost sensitive Task
Users with Dynamic Requirement Task
Different categories tasks are encapsulated into different kinds of Cloudlets.
Submitting Cloudlets
Submitting Cloudlets
Aggregate Cloudlets Workload
Shared Queue with FCFS
With Different QoS Requirements
Resource Allocation Layer
MQDS: Energy-saving scheduling strategy with diversified QoS constraints
Time Prior Cloudlets
Cost Prior Cloudlets
Benefit Prior Cloudlets
Workload Collector
Price Collector
Sorting Descending by Time
Sorting Descending by Cost
Sorting Descending by Benefit
TPDS
CPDS
BFDS
MQDS conducts decision needs:
1. Users' QoS requirements;
2. Workload Information;
3. Disks' Prices Inforamtion
Resource Layer
Scheduling
Scheduling
Scheduling
High Speed Disk Zone
Low Speed Disk Zone
Every disk has its distinctive queue for receiving tasks.
High speed disks and low speed disks are divided in different zones

Fig. 1. Framework of the Multi-QoS Disk scheduling strategy (MQDS).

3.2. Framework of MQDS

The framework of MQDS is shown in Fig. 1. There are three layers: Cloudlets Layer, Resource Allocation Layer and Resource Layer. At the Cloudlets Layer, different kinds of tasks will be encapsulated by Cloudlets and submitted to the Resource Allocation Layer. Usually, tasks in cloud related environments can be categories into 3 types according to their QoS requirements: time-sensitive tasks, cost-sensitive tasks and dynamic requirement tasks. User submits different types of tasks (TP, CP or BFB) to the MQDS through a shared queue. The proposed data scheduling strategy MQDS is implemented at the Resource Allocation Layer, which schedules the aggregated Cloudlets to disks with different running mode according to their QoS' requirements, and disks' workload and price information. The real-time workload information on the every disk is collected by the Workload Collector,

Table 3
Symbol meanings in the three algorithms.

Term	Symbol	Meaning
Disk Set	$\mathrm{DN} = \{\mathrm{dn}_1, \mathrm{dn}_2, \ldots \mathrm{dn}_n\}$	Total set of disks in a cloud storage system
Data _Set	$\mathrm{DU} = \{\mathrm{du}_1, \mathrm{du}_2, \ldots \mathrm{du}_u\}$	Data set submitted by a user for storage
Data _Size	$\mathrm{DS} = \{\mathrm{ds}_1, \mathrm{ds}_2, \ldots \mathrm{ds}_u\}$	Data size of every data in a data set
Partitioning _Number	$\mathrm{PN} = \{\mathrm{pn}_1, \mathrm{pn}_2, \ldots \mathrm{pn}_u\}$	Partitioning number of every data in a data set
parallel_number	$\mathrm{D} = \{\mathrm{d}_1, \mathrm{d}_2, \ldots \mathrm{d}_u\}$	Parallel number calculated based on the Partitioning Number
Workload _List	$\mathrm{WL} = \{\mathrm{wl}_1, \mathrm{wl}_2, \ldots \mathrm{wl}_n\}$	Workload list of every disk in a disk set
Disk _Price _List	$\mathrm{DP} = \{\mathrm{dp}_1, \mathrm{dp}_2, \ldots \mathrm{dp}_n\}$	Price list of every disk in a disk set
Response_Time_Estimator	$\mathrm{RT} = \{\mathrm{rt}_1, \mathrm{rt}_2, \ldots \mathrm{rt}_n\}$	Estimating the response time of every disk in a disk set
Candidate Disk	CN	Disks that can satisfy the QoS requirements of a user
Bandwidth	BW_{s-t}	The bandwidth from the source node of cloudlet submitted to the target node to be scheduled
Waiting Queue	$\mathrm{Q} = \{\mathrm{q}_{1,}\mathrm{q}_{2,} \cdots \mathrm{q}_{\mathrm{L}}\}$	Waiting queue of the ith disk
Transfer rate of disk	TF_{disk}	The transfer rate of a disk to process certain data request
Initial price of disk	IP_{disk}	The initial price of a disk set in the beginning of the experiments
DB_Function	DB_Function	Function to calculate the benefit of time consumption
BB_Function	BB_Function	Function to calculate the benefit of cost consumption

while the consumed cost of the corresponding disk is collected by the Price Collector. According to the QoS requirements and the collected information, MQDS chooses different disk scheduling algorithms to allocate the tasks onto suitable disks, with the aim of minimizing the energy consumption of the whole cloud storage system and satisfying the QoS requirements.

3.3. Algorithms in MQDS

According to users' different QoS requirements, the MQDS framework executes different scheduling algorithms: Time Prior Disk scheduling Algorithm (TPDS), Cost Prior Disk scheduling Algorithm (CPDS), or Benefit Function Based Disk scheduling Algorithm (BFDS). They are described in Algorithm 1, Algorithm 2, and Algorithm 3, respectively.

Symbols used in the three algorithms are listed in Table 3.

3.3.1. Time Prior Disk scheduling algorithm (TPDS)

The main idea of the Time Prior Disk scheduling algorithm is optimizing the energy consumption and sub-optimizing the response time while satisfying users' QoS requirements. That is when sorting candidate disks (with QoS requirements satisfied), the energy consumption is ordered the first, the response time is the second and the cost expenditure is the last. The proposed TPDS is described in ALGORITHM 1.

3.3.2. Cost Prior Disk scheduling algorithm (CPDS)

The main idea of the Cost Prior disk scheduling algorithm is optimizing the energy consumption and sub-optimizing the cost while satisfying the users' QoS requirements. That is when sorting candidate disks (with QoS requirements satisfied), the energy consumption is ordered the first, the cost is the second and the response time is the last. The proposed Cost Prior Disk scheduling algorithm is described in ALGORITHM 2.

3.3.3. Benefit Function-based Disk scheduling algorithm (BFDS)

As described before, time-sensitive tasks will be scheduled by TPDS and cost-sensitive tasks will be scheduled by CPDS. However, there are tasks that are more sensitive to time at a certain point. Beyond this point, they may be more sensitive to cost, or vice versa. That is the task's QoS requirements are time-varying, e.g. the data archiving tasks. When allocating this type of tasks, if they are cost-sensitive at the beginning, they will be scheduled to disks in low speed mode with cheaper cost and slow transfer rate. When the time is close to the Deadline and they become time-sensitive, they can be scheduled to disks in high speed mode. Similar situations exist in the scientific computing tasks or deep learning training tasks. They are time-sensitive initially and need to be scheduled to disks in high speed mode at expensive cost. When the cost is close to the Budget, they may become cost-sensitive and prefer disks in the low speed mode. To the best of our knowledge, there are still no scheduling algorithms dealing with this kind of scenarios up to now. The proposed BFDS schedules tasks with time-varying QoS requirements first, with profiling the dynamic QoS requirements by the benefit function.

The benefit function is constructed first in BFDS. The objective is using the function to profile user's varying benefits by the elapsed time and expended cost. It provides a basis for MQDS to select a suitable algorithm to maximize user's benefit. User can define different benefit functions to profile the dynamic QoS requirements. Fig. 2 depicts four common forms of benefit functions, in which β is the varying benefit value with the elapsed time or expended cost. The cross point of the horizontal axis and the vertical axis represents the Deadline or Budget point.

$$\beta = \begin{cases} a, & t < bD \\ a - c\,(t - bD)\,, & t \geq bD \end{cases} \tag{1}$$

The function form in Fig. 1 and Fig. 2(c) is adopted in our BFDS, which represents user's requirements in cloud environments more precisely according to our observation. It is expressed by formula (1), where a, b, c are constants defined by user. More details about the benefit function and the values selected for *a*, *b*, *c* and *d* in the formula (1) will be discussed in Performance Evaluation Section, in formula (2) (3) and Table 5

The main idea of the Benefit Function based Disk scheduling Algorithm is reducing the energy consumption to the maximum extent while satisfying the Budget and Deadline constraints. Based on the time-based benefit function or cost-based benefit function submitted by user, BFDS chooses TPDS or CPDS to schedule the time-varying requests to maximize the benefits.

The proposed Benefit Function-based Disk scheduling algorithm is described in ALGORITHM 3.

4. Performance evaluation

To evaluate the MQDS on metrics such as response time, energy consumption and cost expenditure, we further extend the CloudSimDisk simulator for simulation experiments. The related parameter settings and adopted workload are listed in Table 4 or profiled in Fig. 3 and Fig. 4 respectively.

The time-based benefit function and the cost-based benefit function are represented by formula (2) and formula (3), respectively.

$$D_{Benefit} = \begin{cases} a1 \quad if\ d1 \times T \leq b1 \times Deadline \\ a1 - c1 \times (d1 \times T - b1 \times Deadline) \\ \quad if\ d1 \times T > b1 \times Deadline \end{cases} \tag{2}$$

ALGORITHM 1: Time Prior Disk scheduling algorithm (TPDS)

```
Input:  DU, DS;Time Prior QoS requirements; Deadline; Budget
Output: Disks assigned to store the input data set
Begin
1: Initialize Time_used=0; Budget_used=0; Disks_assigned={};
2: for each data $d_i \in$ DU do
3:    calculate the parallel_number of $d_i \rightarrow pn_i$ based on the PN (Partitioning _Number);
4:    Partition data $d_i$ into $pn_i$ data block;
5:    Use the Response_Time_Estimator to calculate the RT according to the WL (Workload_List);
      /* RT is the basis for sorting Candidate Disks in step 18, and Workload_List is return by
      Workload_Collector. */
6:    Estimate the cost incurred in disk usage of every disk DP;
7:    cost{}= size of data $ds_i/\{dp_1, dp_2, \cdots dp_n\}$ /* cost{} is the base value for sorting Candicate Disks in
        step 20 */
8:    Compute the remaining Budget and time against Deadline;
9:      Budget_remaining=Budget-Budget_used; / *Budget_used is initialized with zero, and updated in
          step 27 */
10:     Time_remaining=Deadline-Time_used; /* Time_used is initialized with zero, and updated in step
          26*/
11:   for every disk $dn_i \in$ DN
12:     if $rt_i$ < Time_remaining and $dp_i$< Deadline-Time_used
13:         Add $dn_i$ to candicate disk set CN;
14:     End if
15:   End for
16:   Sort Candicate Disks CN in ascending order by energy consumption rate;
17:     if energy consumption rate is the same as that of otherCandidate Disks
18:         Sort the disks in ascending order by the expected response time; /* The disks in high speed
            mode will be at the front to minimize the response time */
19:         if expected response time is the same as that of other Candidate Disks
20:             Sort the disks in ascending order by cost{}; /* The disks in low speed mode (more
            cheaper) will be at the front to minimize the cost expenditure with QoS satisfied */
21:         end if
22:     end if
23:   Select the first $pn_i$ disks in CN to parallel process data $d_i$;
24:   Collect the actual Time_used and Cost_used of data $d_i$;
25:   Update the Time_used and Cost_used;
26:     Time_used= Time_used+maximal time used by the $pn_i$ data blocks of data $d_i$;
27:     Cost_used= sum of the cost used by all the $pn_i$ data blocks of data $d_i$;
28:   if Time_used > Deadline or Cost_used > Budget
29:      printing error information: the request can't be satisfied;
30:      exit with error;
31:   end if
32:   else
33:     printing success information: the request has been satisfied;
34:     Disks_assigned= Disks_assigned+the assigned $pn_i$ disks;
35:   end else
36: End for
37: Return Disks_assigned;
End
```

$$B_{Benefit} = \begin{cases} a2 \quad if\ d2 \times Cost \leq b2 \times Budget \\ a2 - c2 \times (d2 \times Cost - b2 \times Budget) \\ \quad if\ d2 \times Cost > b2 \times Budget \end{cases} \tag{3}$$

The parameters in the formulas are listed in Table 5.

Evaluating strategies or algorithms using simulator is an efficient way to testing their efficiency and effectiveness for large scale distributed environments. As we know, CloudSim [39] is a simulator designed for simulating the cloud environment to evaluate scheduling strategies. Its effectiveness has been widely verified in past years [40]–[41]. CloudSimDisk is one of its extensions for evaluating energy-aware disk scheduling algorithms [32]. CloudSimkDisk is commonly used and chosen as the simulator in this paper.

ALGORITHM 2: Cost Prior Disk scheduling algorithm (CPDS)

```
Input:  DU, DS, Cost Prior QoS requirements; Deadline; Budget;
Output: Disks assigned to store the input data set
Begin
1: Initialize Time_used=0; Budget_used=0; Disks_assigned={};
2: for each data $d_i \in$ DU do
3:   calculate the parallel_number of $d_i \rightarrow pn_i$ based on the PN (Partitioning _Number);
4:   Partition data $d_i$ into $pn_i$ data block;
5:   Use the Response_Time_Estimator to calculate the RT according to the WL (Workload_List);
     /* RT is the basis for sorting Candicate Disks in step 20, and Workload_List is return by
     Workload_Collector.*/
6:   Estimate the cost incurred in disk usage of every disk DP;
7:   cost{}= size of data $ds_i/\{dp_1, dp_2, \cdots dp_n\}$ /* cost{} is the base value for sorting Candicate Disks in
       step 18 */
8:   Compute the remaining Budget and time against Deadline;
9:     Budget_remaining=Budget-Budget_used; // Budget_used is initialized to zero, and updated in step
          27
10:    Time_remaining=Deadline-Time_used; // Time_used is initialized to zero, and updated in step 26
11:   for every disk $dn_i \in$ DN
12:     if $rt_i$ < Time_remaining and $dp_i$< Deadline-Time_used;
13:          Add $dn_i$ to candidate disk set CN;
14:     End if
15:   End for
16:   Sort Candicate Disks CN in ascending order by energy consumption rate;
17:     if energy consumption rate is the same as that of other Candidate Disks
18:         Sort the disks in ascending order by cost{}; / * The disks in low speed mode (more cheaper) will
            be at the front */
19:          if disk price is the same as that of other Candidate Disks
20:             Sort the disks in ascending order by its expected response time; /*  The disks in high speed
            mode will be at the front to minimize the response time with QoS satisfied */
21:          end if
22:     end if
23:   Select the first $pn_i$ disks in CN to parallel process data $d_i$;
24:   Collect the actual Time_used and Cost_used of data $d_i$;
25:   Update the Time_used and Cost_used
26:     Time_used= Time_used+maximal time used by the $pn_i$ data blocks of data $d_i$;
27:     Cost_used=sum of the cost used by all the $pn_i$ data blocks of data $d_i$;
28:   if Time_used > Deadline or Cost_used > Budget
28:      Print error information: the request can't be satisfied;
29:      exit with error;
30:   end if
31:   else
32:     Print success information: the request has been satisfied;
33:     Disks_assigned= Disks_assigned+the assigned $pn_i$ disks;
34:   end else
35: End for
36: Return Disks_assigned;
End
```

Table 4
Software and hardware configuration in the experiments.

Equipment/software	Type/version
CPU	Intel(R) Core(TM) i5-4590 CPU @ 3.30 GHz 3.3 GHz
Memory size	4.0 GB
Hard disk	1TB(TOSHIBA DT01ACA100 ATA Device)
Network card	Realtek PCIe GBE Family Controller
Operating system Energy-aware Disk Simulator	Windows 10 CloudSimDisk1.0
Cloud environment simulator	CloudSim 4.0

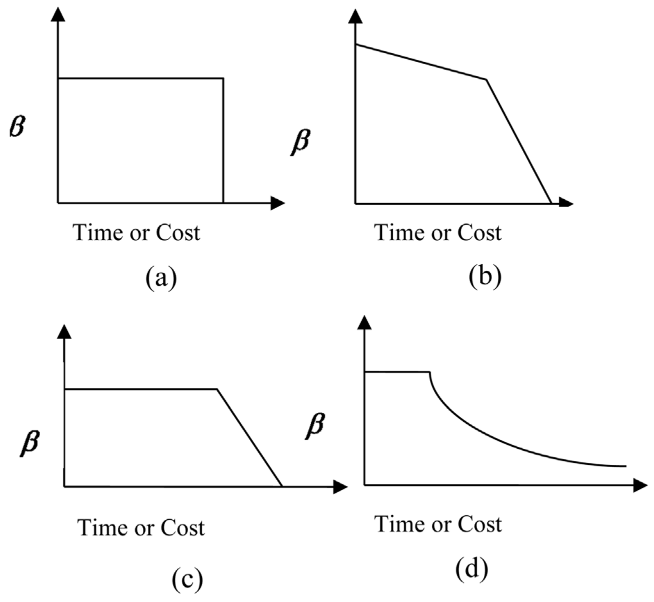

Fig. 2. Common forms of benefit functions.

ALGORITHM 3: Benefit Function-based Disk scheduling Algorithm (BFDS)

```
Input:  DU, DS, Benefit Function-based QoS requirements, DB_Function, BB_Function, Deadline, Budget;
Output:  Disks assigned to store the input data set.
Begin
1: Initialize Time_used=0; Budget_used=0; Disks_assigned={};
2: for each data $d_i$ ∈ DU do
3:    compute the benefit based on elapsed time;
4:    DV= DB_Function (Time_used); /* which will be discussed in detail in the performance evaluation section (Section 4) */
5:    compute the benefit based on expended cost;
6:    CV= BB_Function (Cost_used); /* which will be discussed in detail in the performance evaluation section (Section 4) */
7:    if DV > CV
8:       choose algorithm TPDS;
9:       update the parameters of Time_used and Cost_used;
10:      update the Disks_assigned set;
11:   end if
12:   if DV < CV
13:      choose algorithm CPDS;
14:      update the parameters of Time_used and Cost_used;
15:      update the Disks_assigned set;
16:   end if
17:   if DV = = CV
18:      randomly choose an algorithm from TPDS or CPDS
19:      update the parameters of Time_used and Cost_used
20:      update the Disks_assigned set;
21:   end if
22: End for
23: Return Disks_assigned;
End
```

To evaluate the three disk scheduling algorithms (TPDS, CPDS, and BFDS) in our MQDS and compare with the extended mainstream state-of-the-art disk scheduling algorithm (E-RRDS), the synthetic workload actuated by a real access trace from wiki (wiki-workload) is used in our experiments, which is one kind of classical workload in real cloud environments. In the CloudSimDisk simulator, every request in wiki-workload is represented as a Cloudlet. As shown in Fig. 3, 5000 Cloudlets are submitted within 2 s through 1200 iterations, and the arrival time of every Cloudlet is profiled in Fig. 3. We simulate the way of submission according to the numbers shown in Fig. 3 during the experiments. The file size requested by each Cloudlet is different. The file size distribution of the 5000 requests is profiled in Fig. 4, which randomly falls within the range of 1 MB–10 MB. The average size is about 6 MB, which is important information for constructing our cloud storage system in the simulated environment. Based on that information, the capacity of the storage system needed in the simulated environment can be obtained.

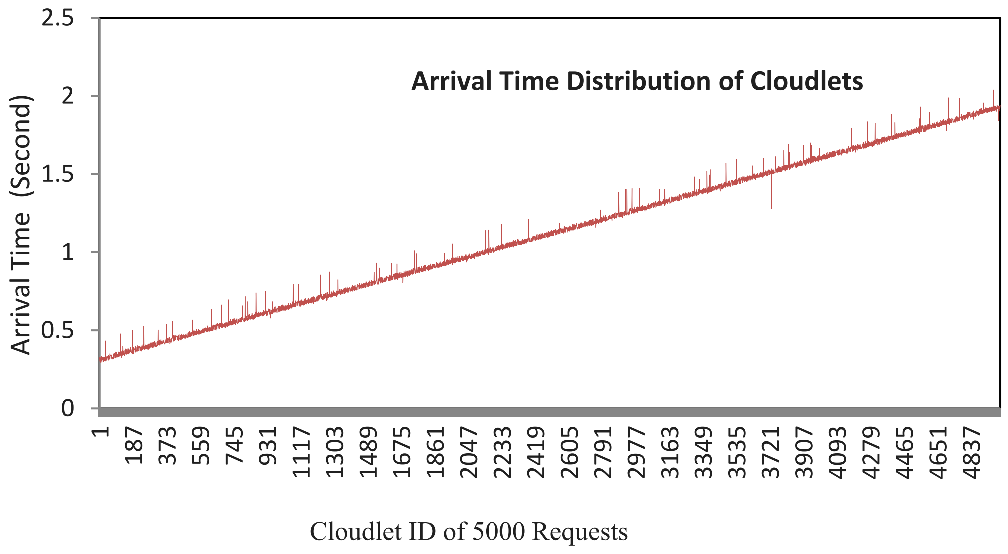

Fig. 3. Time distribution of wiki-workload in the experiments.

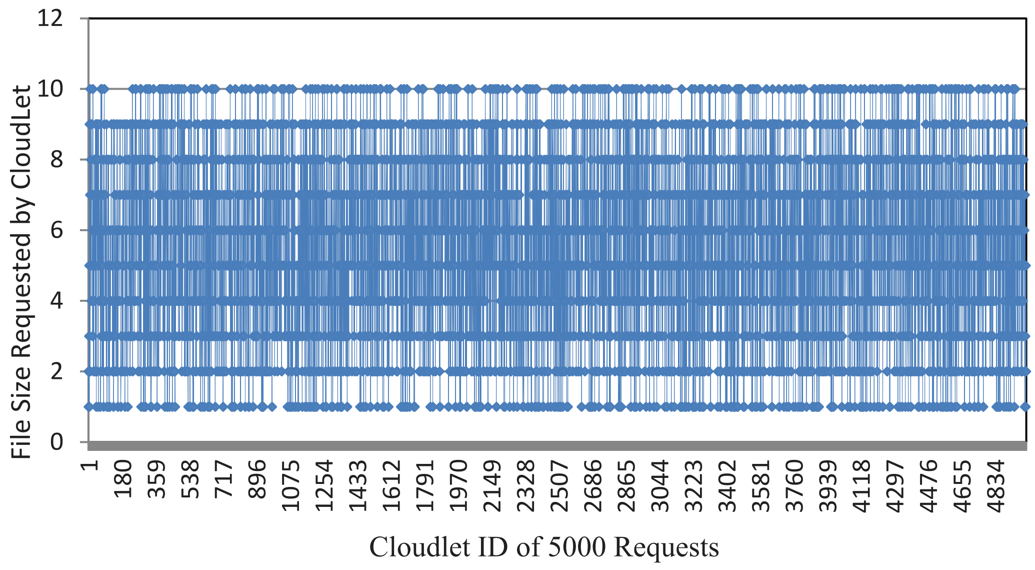

Fig. 4. File size distribution of the Cloudlet requests.

Table 5
Parameters and values related to the benefit functions.

Parameter	Value
Deadline	10000 s
Budget	1000000$
$a1$	1000
$b1$	1
$c1$	1
$d1$	1000
$a2$	1000
$b2$	0.001
$c2$	1
$d2$	1

The three disk scheduling algorithms (TPDS, CPDS, BFDS) are compared with the extended Robin Round Disk scheduling algorithm (E-RRDS) under the above parameters settings. All the four algorithms are evaluated in terms of response time, cost consumption, and energy consumption under two types of cloud environments. The first is a cloud storage system consisting of homogeneous speed reconfigurable disks, and the other consists of heterogeneous speed reconfigurable disks. Although HDDs are selected in our experiments, other kinds of storage devices with reconfigurable ability are also applicable, such as SDDs or NVMs, as the framework is devices independent. The reason of selecting a certain type of storage device is that it is easy to obtain the publicly available parameter details.

4.1. Experiments in the cloud storage system with homogeneous disks with reconfigurable speeds

The type of disks used in this part of experiments is HUC109090CSS600 produced by Hitachi Company. Because our proposed strategy is designed for speed reconfigurable disks, the disks in the simulated cloud storage systems have different modes: high speed and low speed. In order to evaluate the performance of response time, energy consumption and cost of the proposed algorithms, we set the disk capacity and related parameters to calculate these metrics values in each mode. Therefore, the following parameters are set:

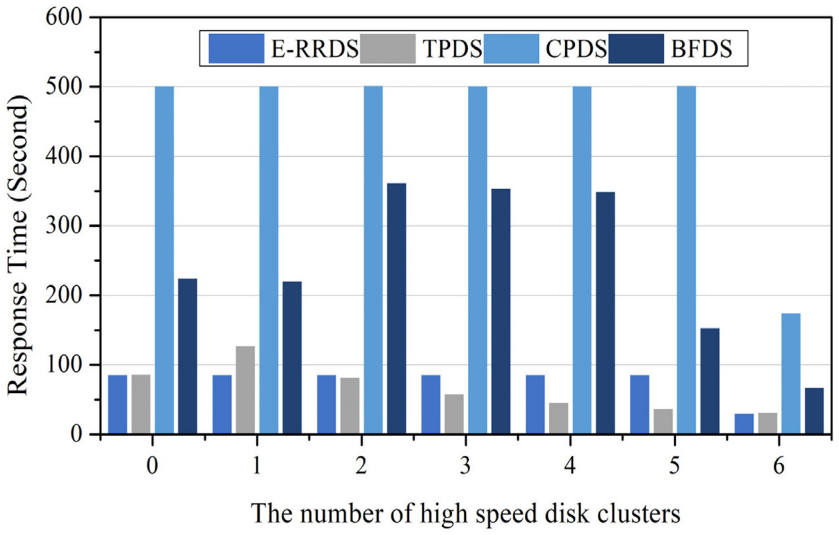

Fig. 5. Response time of the four scheduling algorithms under different high-speed disk clusters configuration.

Table 6
Parameter list of Disk HUC109090CSS600 by Hitachi.

Parameter	Value
Capacity	900G
Average rotate latency	0.003 s
Average location latency	0.004 s
Maximum transfer rate in high speed mode	198 MB/s
Minimum transfer rate in low speed mode	59 MB/s
Cost for storing file	0.003 Cent/MB*day
Cost for processing file in high speed mode	10.0 Cent/MB
Cost for process file in low speed mode	3.3 Cent/MB
Power at active state in high speed mode	5.8 W
Power at idle state in high speed mode	3.0 W
Power at active state in low speed mode	4.1 W
Power at idle state in low speed mode	1.3 W

- High Speed Mode: Capacity, Average Rotate Latency, Average Location Latency, Maximum Transfer Rate; Cost for Storing File, Cost for Processing File; Power at Active State; and Power at Idle State;
- Low Speed Mode: Minimum Transfer Rate; Cost for Process File; Power at Active State; and Power at Idle State.

The disk related parameters are extracted from the storage model in literature [42]. The energy model of the multi-speeds disks is chosen according to the model in literature [10]. The cost related parameters are obtained from the approximate average price of Amazon Cloud, Aliyun and Tencent Cloud after investigation. The number of disk clusters used in the experiments is six (we think it is suitable for testing performance of disks running in different speed modes with consideration of the testing time), each of which can be set in high or low speed mode. The number of high speed disk clusters ranges from 0 to 6, and the number of low speed disk clusters ranges from 6 to 0. Detailed parameters of HUC109090CSS600 are demonstrated in Table 6. With the above settings, the experiments are conducted 10 times, each taking about one day. The average evaluation results (Response Time, Energy Consumption, Cost) of the three algorithms (TPDS, CPDS, BFDS) and the E-RRDS are listed in Table 7.

As shown in Table 7, under different disk speeds configurations (high or low), the proposed TPDS, CPDS and BFDS algorithm have consistent outcome: TPDS algorithm has the best performance on response time for systems with mixed high and low speed storage for wiki workloads, especially when the numbers of the high speed clusters and the low speed cluster are roughly equal. CPDS algorithm consumes the least cost. For BFDS, the performance on response time and cost expense is between those of TPDS and CPDS. It can also be observed that when the numbers of high speed disk clusters are 2–5, TPDS outperforms E-RRDS on response time and energy consumption. Only when all of the disk clusters are set in high speed mode or in low speed mode, E-RRDS has slight advantage than TPDS. Table 7 lists the results of the response time, energy consumption and cost. Meanwhile, for a more straightforward comparison, the metrics of response time, energy consumption and cost expenditure of the four disk scheduling algorithms are visualized in Figs. 5–7.

As shown in Fig. 5, in terms of response time, TPDS has the best performance. Its advantage is outstanding when the ratio of high speed and low speed disk is comparable. However, CPDS has the worst performance on response time because CPDS usually chooses low speed disks to execute tasks, resulting in longer storage and processing latency. Fortunately, all of the tasks are finished before the Deadline by CPDS, proving it is more suitable for cost-sensitive tasks. The response time of BFDS is longer than TPDS, but is lower than CPDS.

As shown in Fig. 6, CPDS is the most cost effective except under the conditions that all of the disks are high speed or low speed disks. TPDS has the worst performance on cost effectiveness because it usually deploys high speed disks at relatively higher prices to conduct the tasks. BFDS can achieve good balance between cost and time.

As shown in Fig. 7, TPDS consumes the least energy when compared with the other three disk scheduling algorithms. CPDS has the worst performance on energy consumption. CPDS usually chooses the cheapest disks at lower speed, which increases the process latency and results in more energy consumed. Similarly, BFDS can achieve good balance among cost, time and energy consumption.

According to our disk scheduling algorithms, TPDS always schedules tasks to the fastest disks. When there is only one high speed cluster, all the tasks are scheduled in that cluster, which may trigger overload. This is the reason why TPDS is slower than RRDS in the scenario of only one high speed cluster. Due to the design of our CPDS, if there are cheaper clusters (slower clusters), the Cloudlet will be scheduled to the cheapest cluster (slowest cluster) to save energy consumption. In this case, Cloudlets is scheduled to the only cluster with low speed disks, resulting in CPDS's RT being ∼500 s for 0–5 high speed clusters. E-RRDS schedules the Cloudlets by Round Robin, balancing the

Table 7
Experimental results of the four disk scheduling algorithms with different disk clusters configurations.

The number of high speed disk clusters	The number of low speed disk clusters	Scheduling algorithm	Response time (Second)	Energy consumption (Joule)	Cost (Cent)
0	6	E-RRDS	85.42731921	2060.435344	94047.648
0	6	TPDS	85.75055167	2093.826944	94047.648
0	6	CPDS	500.5680729	2059.516938	94047.648
0	6	BFDS	224.0230587	2096.227098	94047.648
1	5	E-RRDS	85.38338564	1886.013038	124659.948
1	5	TPDS	126.4767141	1272.932627	233514.848
1	5	CPDS	500.468222	2059.59755	94047.648
1	5	BFDS	219.9745911	1469.598858	198648.048
2	**4**	**E-RRDS**	**85.38894818**	**1709.198656**	**155821.648**
2	**4**	**TPDS**	**80.90268669**	**1109.009019**	**264783.748**
2	**4**	**CPDS**	**500.9154927**	**2061.200985**	**94047.648**
2	**4**	**BFDS**	**360.9112241**	**1743.803663**	**150959.6813**
3	**3**	**E-RRDS**	**85.41676969**	**1533.387912**	**186802.448**
3	**3**	**TPDS**	**57.42395752**	**1073.872822**	**272796.948**
3	**3**	**CPDS**	**500.7023951**	**2060.299767**	**94047.648**
3	**3**	**BFDS**	**352.9429159**	**1731.490785**	**153630.748**
4	**2**	**E-RRDS**	**85.36614608**	**1359.213578**	**217582.248**
4	**2**	**TPDS**	**44.68035514**	**1064.464464**	**275925.848**
4	**2**	**CPDS**	**500.4989279**	**2059.425275**	**94047.648**
4	**2**	**BFDS**	**348.5594036**	**1727.771671**	**154673.7147**
5	**1**	**E-RRDS**	**85.42459546**	**1189.789187**	**247531.248**
5	**1**	**TPDS**	**36.60511542**	**1058.889154**	**277788.448**
5	**1**	**CPDS**	**501.2170156**	**2059.911143**	**94637.248**
5	**1**	**BFDS**	**152.7580905**	**1309.144651**	**232000.648**
6	0	E-RRDS	29.66908043	1009.019718	278860.448
6	0	TPDS	31.17488594	1062.669222	278860.448
6	0	CPDS	173.9446841	1007.185567	278860.448
6	0	BFDS	66.86733548	1048.798309	278860.448

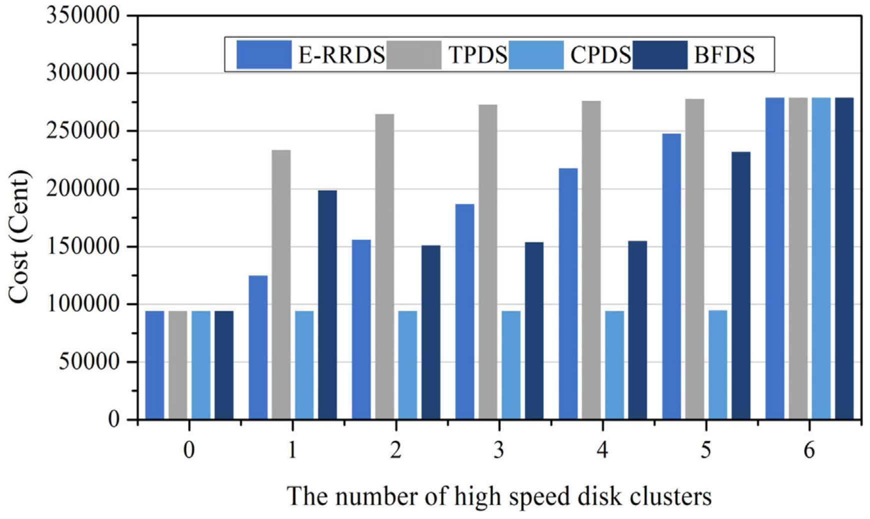

Fig. 6. Cost expenditure of the four scheduling algorithms under different high-speed disk clusters configuration.

energy consumption constraint and some randomness of the workload. Furthermore, as RRDS is the default scheduling algorithm in CloudDisksim Simulator, we only extend the properties of energy consumption and cost of Cloudlet for speed reconfigurable disks for E-RRDS. The response time is achieved directly from the system, which is ~84 s for 0–5 high speed clusters.

As a whole, when the numbers of high speed and low speed disks are comparable, the three proposed algorithms in MQDS are more energy efficient than the E-RRDS. In our experiments, when the numbers of high speed disk clusters and the number of low speed disk clusters are both 3, MQDS has an outstanding performance on energy efficiency, It can be well observed in Figs. 8–10.

As shown in Fig. 8, although the finishing time of an individual cloudlet is fluctuating for all the four disk scheduling algorithms, as a whole, the response time of TPDS (red dots) is the shortest, followed by E-RRDS (blue dots). CPDS (green dots) is the longest. Similarly, the performance of BFDS (purple dots) is between TPDS and CPDS.

As shown in Fig. 9, although the energy consumption of an individual cloudlet is fluctuating for all the four disk scheduling algorithms, as a whole, TPDS (red dots) has the best performance on energy consumption when finishing the wiki-workload. CPDS (green dots) consumes the most energy. The energy consumption of BFDS (purple dots) is still between TPDS and CPDS. The energy consumption of E-RRDS (blue dots) is fluctuating for different Cloudlets. The reason is also obvious: E-RRDS only round-robins the disks and ignores the processing speed or energy consumption rate.

As shown in Fig. 10, although the cost expenditure of an individual cloudlet is fluctuating for all the four disk scheduling algorithms, as a whole, CPDS (green dots) expends the least cost

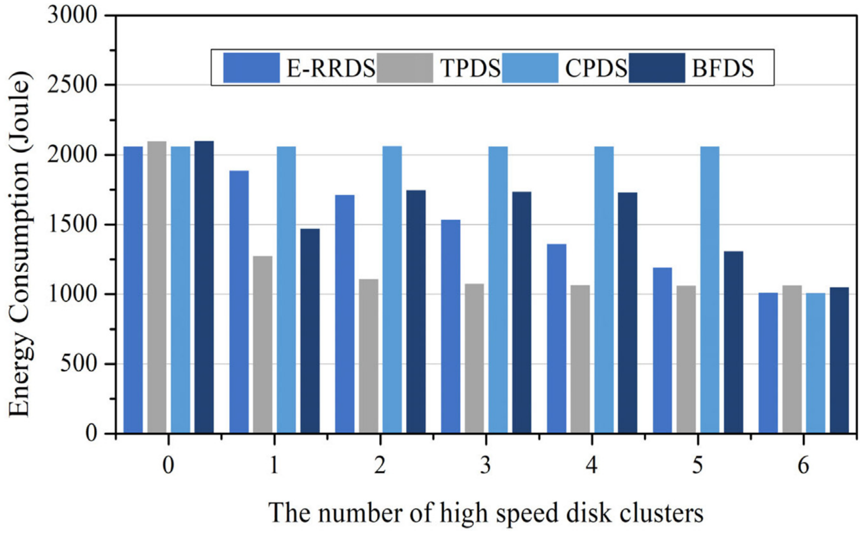

Fig. 7. Energy consumption of the four scheduling algorithms under different high-speed disk clusters configuration.

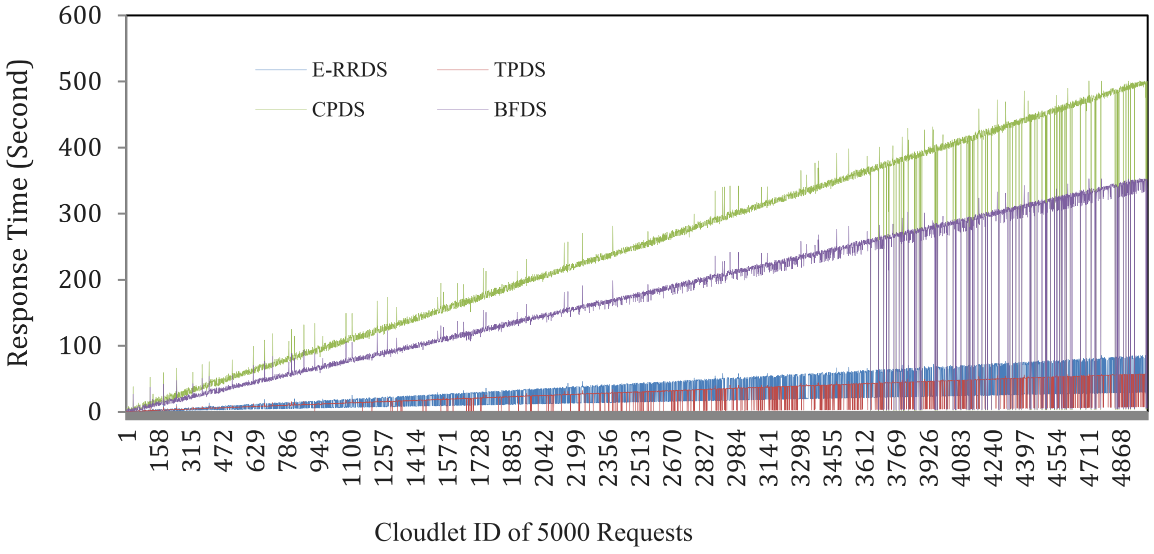

Fig. 8. Finishing time of different Cloudlets under the homogeneous disk configuration.

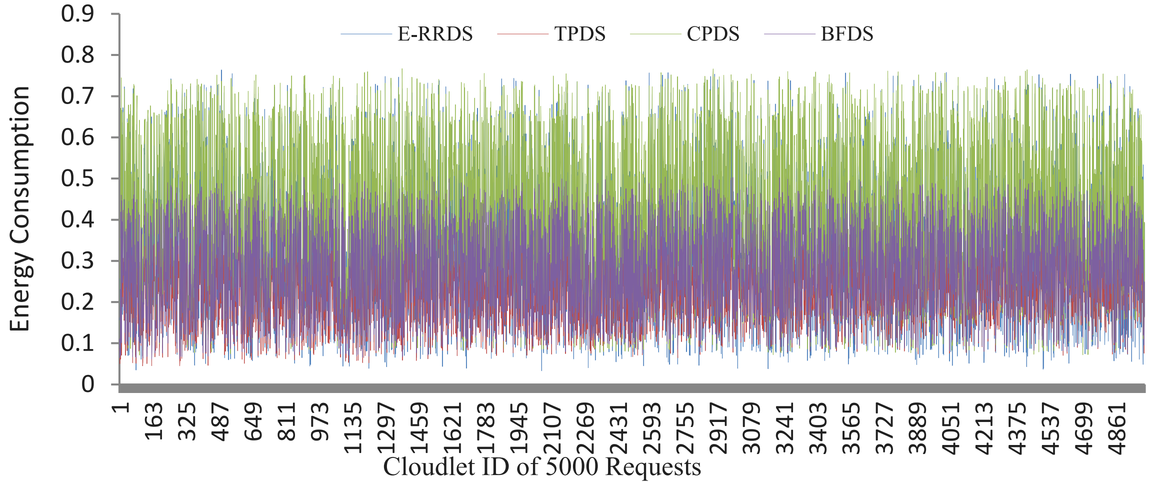

Fig. 9. Energy consumption of different Cloudlets under the homogeneous disk configuration.

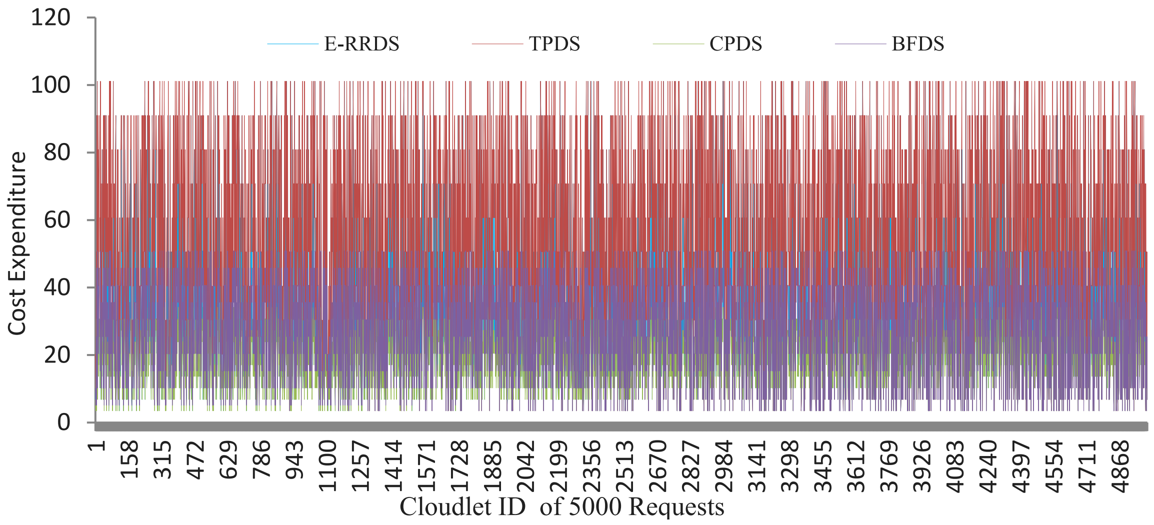

Fig. 10. Cost expenditure of different Cloudlets under the homogeneous disk configuration.

when compared to the other three algorithms. On the contrary, the cost of TPDS (red dots) is the highest while BFDS (purple dots) is in between. The cost of E-RRDS (blue dots) is also fluctuating. Costs consumed by some Cloudlets are comparable to CPDS (the least), and some are comparable to TPDS (the most), which are also due to the fact that E-RRDS does not consider the price or disk speed when scheduling tasks.

As shown in the above tables and figures, under the homogeneous disk configuration, the proposed algorithms in the MQDS are energy efficient than the current mainstream disk scheduling algorithm (E-RRDS). Moreover, the proposed MQDS provides more options for different QoS requirements in the cloud environment.

4.2. Experiments in the cloud storage system with heterogeneous disks with reconfigurable speed

Disks adopted in this part of experiments are heterogeneous. As observed from the homogeneous disk experiments, when the numbers of high speed clusters and the low speed clusters are rough equal, MQDS can achieve the biggest advantage. Due to the paper length limit, we evaluate the performance of the four disk scheduling algorithms when the numbers of the high speed and of the low speed disk clusters are both 3 with the disks being heterogeneous. Two of them are HUC109090CSS60 by Hitachi Company (1 is set in high speed mode, the other in low), two are ST6000VN0001 by Seagate company (1 in high speed mode, the other in low), the remaining 2 disk clusters are MG04SCA500E by Toshiba company (1 in high speed mode, the other in low). The disk related parameters of HUC109090CSS60 are extracted from the storage model in literature [42]. The parameters of ST6000VN0001 are extracted from literature [43], and the parameters of MG04SCA500E are extracted from literature [44]. The energy models of the three types of disks are chosen from literature [10]. The detailed parameters of three types are demonstrated in Table 4, Table 8, Table 9, respectively. Under the parameter settings, the performance of metrics such as response time, energy consumption and cost expenditure of the four disk scheduling algorithms (TPDS, CPDS, BFDS, E-RRDS) are evaluated with 5000 Cloudlets finished in 2 s. The experimental results are demonstrated in Table 10.

As shown in Table 10, TPDS has the best performance on response time and energy consumption under the heterogeneous disk architecture (disks from different vendors and in different speed modes). CPDS performs best in cost effectiveness, with cost consumed being about 30% of the other algorithms. BFDS achieves a balance between TPDS and CPDS in terms of performance. E-RRDS outperforms CPDS and BFDS in terms of response time, but underperforms TPDS algorithm. Meanwhile, E-RRDS consumes the most energy among the four. Its cost consumption is less than

Table 8
Parameters list of Disk ST6000VN0001 by Seagate.

Parameter	Value
Capacity	600G
Average rotate latency	0.00416 s
Average location latency	0.0085
Maximum transfer rate in high speed mode	216 MB/s
Minimum transfer rate in low speed mode	64 MB/s
Cost for storing file	0.004 Cent/MB*day
Cost for processing file in high speed mode	12.0 Cent/MB
Cost for process file in low speed mode	4.0 Cent/MB
Power at active state in high speed mode	11.27 W
Power at idle state in high speed mode	6.9 W
Power at active state in low speed mode	8.0 W
Power at idle state in low speed mode	3.0 W

Table 9
Parameters list of Disk MG04SCA500E by Toshiba.

Parameter	Value
Capacity	500G
Average rotate latency	0.00417 s
Average location latency	0.009
Maximum transfer rate in high speed mode	215 MB/s
Minimum transfer rate in low speed mode	64 MB/s
Cost for storing file	0.005 Cent/MB*day
Cost for processing file in high speed mode	11.0 Cent/MB
Cost for process file in low speed mode	3.8 Cent/MB
Power at active state in high speed mode	11.3 W
Power at idle state in high speed mode	6.2 W
Power at active state in low speed mode	8.0 W
Power at idle state in low speed mode	2.7 W

Table 10
Experimental results of different types of disk configuration.

Scheduling algorithm	Response time (Second)	Energy consumption (Joule)	Cost (Cent)
E-RRDS	84.90188538	2565.270668	205865.663
TPDS	66.6532607	1933.050016	301667.6175
CPDS	500.1972857	2067.603483	94825.1965
BFDS	355.6826107	2022.752327	163772.6702

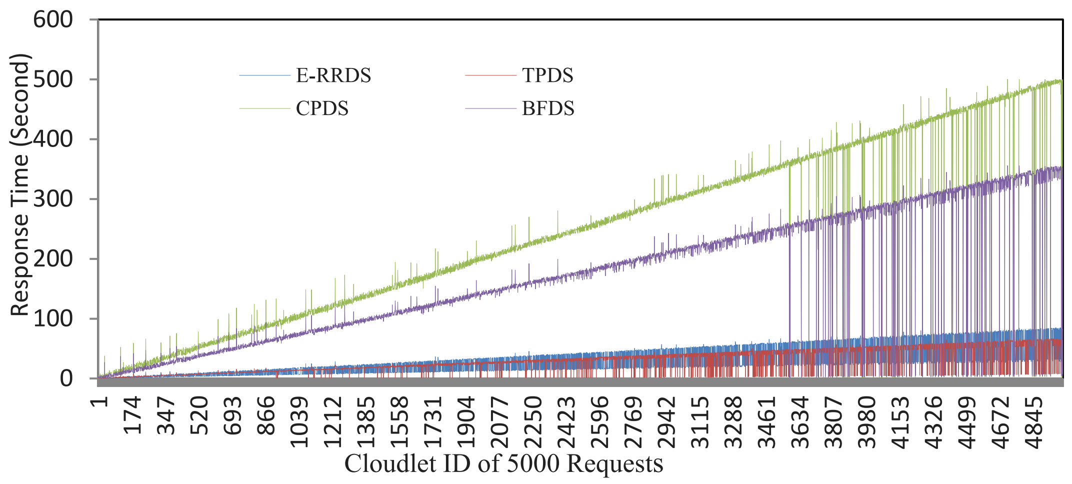

Fig. 11. Finishing time of different Cloudlets under the heterogeneous disks configuration.

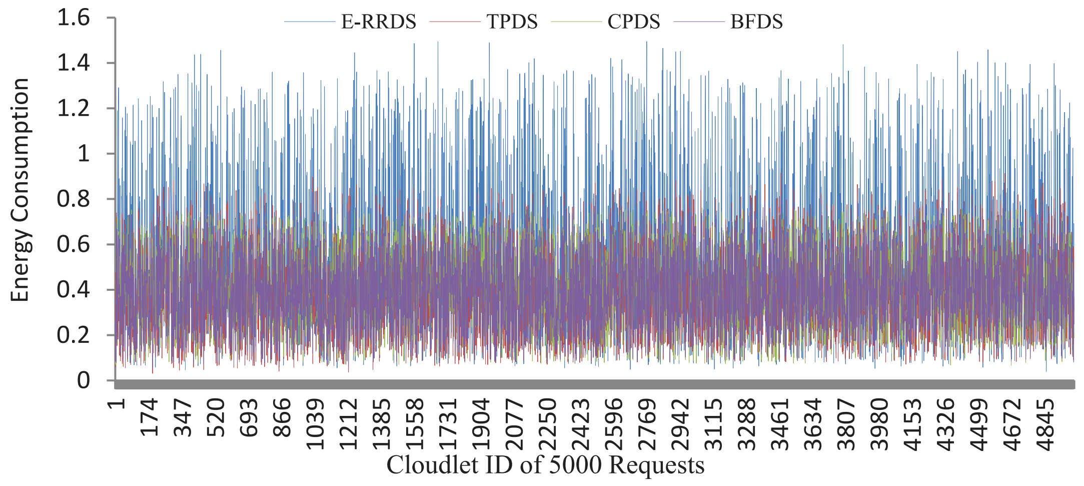

Fig. 12. Energy consumption of different Cloudlets under the heterogeneous disks configuration.

that of TPDS, but greater than those of CPDS and BFDS. It can be seen that under heterogeneous disk configuration, our proposed energy-saving disk scheduling strategy with diverse QoS constraints is more energy efficient than E-RRDS. Furthermore, it provides more flexible QoS options to users.

Similarly, the intermediate results of the four algorithms finishing each Cloudlet under wiki-workload are profiled in Figs. 11–13 in terms of response time, energy consumption and cost expenditure.

As shown in Fig. 11, although the finishing time of an individual cloudlet is fluctuating for all the four disk scheduling algorithms, as a whole, TPDS (red dots) outperforms the other three while CPDS (green dots) has the worst performance in terms of response time for each Cloudlet. Likewise, BFDS (purple dots) performs between TPDS and CPDS. And E-RRDS (blue dots) underperforms TPDS.

As shown in Fig. 12, the energy consumption of an individual cloudlet is fluctuating, however, when it comes to the total energy consumption, all of the three proposed algorithms are more energy efficient than E-RRDS (blue dots). As to the energy consumption of each Cloudlet, TPDS (red dots) and CPDS (green dots) take the lead in turn.

As shown in Fig. 13, although the cost expenditure of an individual cloudlet is fluctuating, as a whole, TPDS (red dots) consumes the highest cost when compared with the other three algorithms. CPDS (green dots) always performs the best in terms of cost effectiveness. Cost expended by BFDS (purple dots) is between CPDS and TPDS. The performance of E-RRDS (blue dots) is comparable to that of TPDS.

Extensive experimental results demonstrate that the proposed three algorithms in MQDS outperform the mainstream E-RRDS. Whenever under the homogeneous or heterogeneous disk configuration, TPDS consumes shortest response time and CPDS has the least cost. BFDS achieves balance between TPDS and CPDS, providing more options to users. Furthermore, all the algorithms in MQDS are more energy efficient than E-RRDS. That is to say that the proposed MQDS is energy efficient and can accommodate diverse QoS constraints well (e.g. requests are processed before Deadline and under Budget). The advantage of MQDS is very important for a cloud environment, where diverse QoS requirements are to be satisfied.

5. Conclusions and future work

An energy-saving scheduling strategy with diverse QoS constraints towards reconfigurable disk architecture (MQDS) is proposed in this paper. It supports three algorithms: Time Prior Disks Algorithm (TPDS) for time-sensitive tasks, Cost Prior Disks Algorithm (CPDS) for cost-sensitive tasks, and Benefit Function-based Disks Algorithm (BFDS) for tasks with time-varying QoS requirements. To verify the proposed strategy, substantial experiments are conducted. Datacenter consisting of speed reconfigurable disks is simulated in CloudSimDisk simulator. 5000

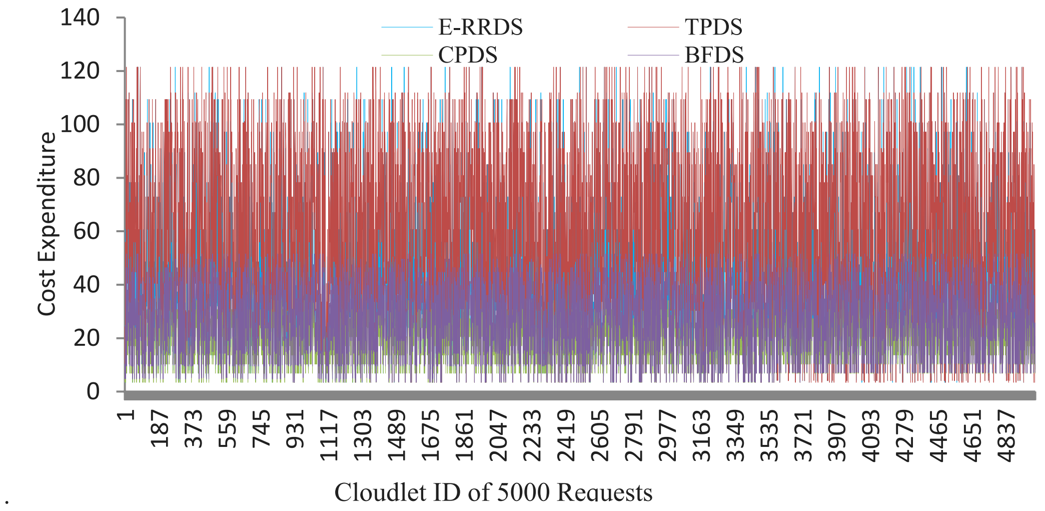

Fig. 13. Cost expenditure of different Cloudlets under the heterogeneous disks configuration.

Cloudlets of wiki-workload are submitted to the disk scheduling algorithms, seeking suitable disks allocation under different QoS requirements. The experimental results demonstrate whenever under the homogeneous or the heterogeneous disks configuration, the three disk scheduling algorithms (TPDS, CPDS, and BFDS) in our proposed MQDS have performance advantages than E-RRDS. TPDS performs the best in terms of response time while CPDS has the best performance on cost effectiveness. As to energy consumption, all of the three algorithms outperform E-RRDS, which verifies the energy efficiency of the proposed MQDS. Moreover, BFDS achieves a good balance between TPDS and CPDS in terms of response time, energy consumption and cost expenditure. BFDS provides more flexible QoS requirement options to users. When the numbers of different speed disks are roughly equal, our proposed MQDS achieves the biggest advantage than the current mainstream algorithm E-RRDS. Moreover, it is worth mentioning that although our proposed MQDS is evaluated on the HDDs with reconfigure speeds, it is also applicable to other reconfigurable storage devices, such as the SDDs or NVMs, because of the high level abstracting ability of the proposed MQDS. Although public wiki-workload is used in our simulation experiments, the designed disk scheduling algorithms are workload independent. The MQDS is designed in a cloud environment to achieve the flexible QoS requirements and energy consumption reduction. If the real workload trace can be obtained, it is also applicable to other workload types, e.g. scientific, deep learning and video apps. Moreover, it also supports dynamic disk modes, as the algorithms collect disks' status in real-time.

In large scale distributed computing environments, conducting substantial experiments in Simulator is an important procedure before implementing the algorithms in real environments. It provides valuable reference for designing a heterogeneous cloud storage system constructed by reconfigurable storage devices. In the future, we will collect more types of workload trace for our simulation experiments to verify its applicability. The dynamic disk modes will be simulated in the extended CloudDiskSim simulator. Furthermore, we will try to integrate the MQDS into the real cloud environments to further verify its energy efficiency and the ability of supporting diverse QoS constraints.

CRediT authorship contribution statement

Xindong You: Conceptualization, Methodology, Validation, Writing – original draft, Funding acquisition. **Dawei Sun:** Formal analysis, Investigation, Data curation, Writing – review & editing, Funding acquisition. **Xueqiang Lv:** Conceptualization, Supervision, Validation, Investigation, Writing – review & editing. **Shang Gao:** Data curation, Investigation, Writing – review & editing. **Rajkumar Buyya:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We acknowledge the support of Natural Science Foundation of Beijing, PR China under Grants No. 4212020, National Natural Science Foundation of China under Grants No. 61972364, 62171043, Defense-related Science and Technology Key Lab Fund, PR China project under Grants No. 6412006200404, Qin Xin Talents Cultivation Program of Beijing Information Science and Technology University, PR China under grant No. QXTCP B201908 and Melbourne-Chindia Cloud Computing (MC3) Research Network, Australia, the Fundamental Research Funds for the Central Universities, PR China under Grant No. 2652021001.

References

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.
[2] R. Rajkumar, C. Lee, J.P. Lehoczky, D.P. Siewiorek, A resource allocation model for QoS management, in: Proceedings 18th Real-Time Systems Symposium. San Francisco, 1997, pp. 330-339.
[3] R. Rajkumar, C. Lee, J.P. Lehoczky, D.P. Siewiorek, Practical solutions for QoS-based resource allocation problems, in: Proceedings 19th IEEE Real-time Systems Symposium, Madrid, 1998, pp. 296-306.
[4] L.L. Wu, S.K. Garg, R. Buyya, SLA-Based resource allocation for software as a service provider (SaaS) in cloud computing environments, in: Proceeding of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011). Washington, DC, USA, May, Vol. 23-26 2011, pp. 195-204.
[5] F. Karaniavoura, MagoutisK, Decision-making approaches for performance QoS in distributed storage systems: A survey, IEEE Trans. Parallel Distrib. Syst. 30 (8) (2019) 1019–1906.
[6] J.F. Chen S. Delimirou C. Martinez, PARTIES: QoS-Aware resource partitioning for multiple interactive services, in: ASPLOS'19 Proceedings of the Twenty-Fourth International Conference on Architecture Support for Programming Languages and Operating Systems. Providence, RI, USA, April, Vol. 13-17, 2019, 107-120.

[7] X. Li, J. Wan, H.N. Dai, et al., A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing, IEEE Trans. Ind. Inf. 15 (7) (2019) 4225–4234.
[8] R. Bhan, A. Singh, R. Pamula, et al., Auction Based Scheme for Resource Allotment in Cloud Computing, Springer, Cham, 2019, pp. 119–141.
[9] Q. Wang, S. Guo, J. Liu, et al., Profit maximization incentive mechanism for resource providers in mobile edge computing, IEEE Trans. Serv. Comput. (2019).
[10] T. Xie, SEA: A striping-based energy-aware strategy for data placement in RAID-structured storage system, IEEE Trans. Comput. 57 (6) (2008) 748–761.
[11] Ashwin A. Mendon, Andrew G. Schmidt, S. Ron, A hardware filesystem implementation with multidisk support, Int. J. Reconfigurable Comput. (10) (2009) 1–23.
[12] K. Mershad, H. Artail, Saghir M.A.R, Hajj H., M. Awad, A study of the performance of a cloud datacenter server, IEEE Trans. Cloud Comput. 5 (4) (2017) 590–603.
[13] M. Mills, The Cloud Begins with Coal: Big Data, Big Networks, Big Infrastructure and Big Power, National Mining Association & American Coalition for clean Coal Electricity, 2013.
[14] Envantage, New report reveals warehouses overspend on energy by £190 m, 2013, Data by World Resources Institute. (14th August, 2013). Retrieved September 13.
[15] Y. Ebadi, Jafari Navimipour N., An energy-aware method for data replication in the cloud environments using a Tabu search and particle swarm optimization algorithm, Concurr. Comput.: Pract. Exper. 31 (1) (2019) e4757.
[16] M. Xu, R. Buyya, Managing renewable energy and carbon footprint in multi-cloud computing environments, J. Parallel Distrib. Comput. 135 (2020) 191–202.
[17] H. Qiu, H. Noura, M. Qiu, et al., A user-centric data protection method for cloud storage based on invertible DWT, IEEE Trans. Cloud Comput. (2019).
[18] Ding. D., Fan. X.C., Zhao. Y.H., Kang. K.X., et al., Q-learning based dynamic task scheduling for energy-efficient cloud computing, Future Gener. Comput. Syst. 108 (2020) 361–371.
[19] Chen. C.H., Chen. S.H., Liang. Y.P., Chen. T.Y., et al., Facilitating external sorting on SMR-based large-scale storage systems, Future Gener. Comput. Syst. 116 (2021) (2021) 333–348.
[20] X.Z. Chen, E.H.M. Sha, W.W. Jiang, C.S. Yang, et al., Refinery swap: An efficient swap mechanism for hybrid DRAM-NVM systems, Future Gener. Comput. Syst. 77 (2017) (2017) 52–64.
[21] R. Buyya, Economic-based distributed resource management and scheduling for grid computing, (Ph.D. thesis), Monash University, Melbourne, Australia, 2002.
[22] S. Venugopal, Scheduling Distributed Data-Intensive Applications on Global Grids, (Ph.D. thesis), Melbourne University, Melbourne, Australia, 2006.
[23] D Poola, S.K. Garg, R. Buyya, Y. Yang, R. Kotagiri, Robust scheduling of scientific workflows with deadline and budget constraints in clouds, in: 2014 IEEE 28th International Conference on Advanced Information Networking and Applications, Victoria, BC, Canada, Vol. 13-16, 2014, pp. 858-865.
[24] R.N. Calheiros, E. Masoumi, R. Ranjan, et al., Workload prediction using ARIMA model and its impact on cloud application' QoS, IEEE Trans. Cloud Comput. 3 (4) (2015) 1.
[25] W. Wang, J.Z. Luo, Song a b dynamic pricing based energy cost optimization in data center environments, Chinese J. Comput. 36 (3) (2013) 599–612.
[26] B. Liao, J. Yu, H. Sun, et al., A QoS-aware Dynamic Data Replica Deletion Strategy for Distributed Storage Systems under Cloud Computing Environments, in: International Conference on Cloud & Green Computing. Xiangtan, China, 1-3 Nov, 2012, pp. 219-225.
[27] W.J. Jiang, P. Wang, Research on a grid resource allocation algorithm based on MAS non-cooperative bidding game, J. Comput. Res. Dev. 44 (1) (2007) 29–36.
[28] Chen J Kong L F, X. Pan, Research on grid resource scheduling algorithm intergrating forecast mechanism with QoS constraint, J. Comput. Res. Dev. (45) (2008) 11–16.
[29] Matthew Malensek, Sangmi Pallickara, Shrideep Pallickara, Minerva: proactive disk scheduling for QoS in multitier, multitenant cloud environments, IEEE Internet Comput. 20 (3) (2016) 19–27.
[30] Amar Ranjan Dash, Sandipta Kumar Sahu, B. Kewal, An optimized disk scheduling algorithm with bad-sector management, 2019, arXiv preprint arXiv:1908.01167.
[31] Saswati Sarkar, Anirban Kundu, Performance enhancement of cloud based storage using disk scheduling technique, Int. J. Cloud Appl. Comput. (IJCAC) 10 (1) (2020) 46–63.
[32] B. Louis, CloudsimDisk: Energy-Aware Storage Simulation in Cloudsim [D] Master thesis, Luleå University of Technology, 2015.
[33] Y. Wang, X. Tao, F. Zhao, B. Tian, A.M.V.V. Sai, SLA-aware resource scheduling algorithm for cloud storage, EURASIP J. Wireless Commun. Networking 2020 (1) (2020) 1–10.
[34] S. Pandiyan, T.S. Lawrence, V. Sathiyamoorthi, M. Ramasamy, Q. Xia, Y. Guo, A performance-aware dynamic scheduling algorithm for cloud-based IoT applications, Comput. Commun. 160 (2020) 512–520.
[35] W. Ahmad, B. Alam, S. Ahuja, S. Malik, A dynamic VM provisioning and de-provisioning based cost-efficient deadline-aware scheduling algorithm for big data workflow applications in a cloud environment, Cluster Comput. 24 (1) (2021) 249–278.
[36] D.K. Sinha, Designing energy management aware task scheduling algorithm and model for cloud data centers, Turk. J. Comput. Math. Educ. (TURCOMAT) 12 (12) (2021) 1008–1016.
[37] Sebastian Sabogal, Alan George, Christopher Wilson, Reconfigurable framework for environmentally adaptive resilience in hybrid space systems, ACM Trans. Reconfigurable Technol. Syst. 13 (3) (2020) 1–32, 2020.
[38] Julian Oppermann, Melanie Reuter-Oppermann, Lukas Sommer, Andreas Koch, Oliver Sinnen, Exact and practical modulo scheduling for high-level synthesis, ACM Trans. Reconfigurable Technol. Syst. 12 (2) (2019) 1–26.
[39] R.N. Calheiros, R. Ranjan, A. Beloglazov, et al., Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. - Pract. Exp. 41 (1) (2011) 23–50.
[40] A.F. Antonescu, T. Braun, Modeling and simulation of concurrent workload processing in cloud-distributed enterprise information systems, in: Proceedings of the 2014 ACM SIGCOMM workshop on Distributed cloud computing, 2014, pp. 11–16.
[41] N. Mansouri, R. Ghafari, B.M.H. Zade, Cloud computing simulators: A comprehensive review, Simul. Model. Pract. Theory (2020) 102144.
[42] HGST Storage Model [EB/OL], http://www.storagereview.com/hgst_ultrastar_c10k900_review, (Accessed 20 May 2018).
[43] Seagate Storage Model [EB/OL], http://www.storagereview.com/seagate_enterprise_nas_hdd_review, (Accessed 20 May 2018).
[44] [Toshiba Storage Model [EB/OL], http://www.storagereview.com/toshiba_mg04sca_enterprise_hdd_review, (Accessed 20 May 2018).

Xindong You is currently an associate professor of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research at Beijing Information Science & Technology University, China. She holds a post-doctoral position at Tsinghua University with the Beijing Institute of Graphic Communication, from 2016 to 2018. Before as a post-doctoral, she is an Associate Professor in Hangzhou Dianzi University. Before joining Hangzhou Dianzi University, she was a Ph.D. student with Northeastern University, from 2002 to 2007. She received her PhD degree in 2007. She is in charge of the National Nature Science Foundation of China from 2014 to 2017 and Nature Science Funding of Zhejiang Province from 2013 to 2015. She has authored about 30 papers in the international conference or journals, most of them are indexed by EI or SCI database. Her current research areas include Natural Language Processing, Image Processing, Distributed Computing, Cloud Storage, Energy Management, Data Replica Management, etc.

Dawei Sun is an associate professor in the School of Information Engineering, China University of Geosciences, Beijing, P.R. China. He received his Ph.D. degree in computer science from Northeastern University, China in 2012, and conducted the Postdoctoral research in the department of computer science and technology at Tsinghua University, China in 2015. His current research interests include big data computing, cloud computing, and distributed systems. He has authored or co-authored over 60 journal and conference papers in the above areas.

Xueqiang Lv is a professor in Beijing Information Science & Technology University. Before joining in Beijing Information & Technology University, he is a post-doctoral of Peking University from 2003 to 2005. Before as a post-doctoral, he is a Ph.D. candidate in Northeastern University from 1998 to 2003. He received his Ph.D. degree in 2003. Until now, he has been in charge of the National Nature Science Foundation of China three times. He has authored about 60 papers in the international conference or journals, most of them are indexed by EI or SCI database. His current research areas include Cloud Computing, Distributed Computing, Natural Language Processing, Image Processing, Information retrieval, Machine Learning, Deep Learning, etc.

Shang Gao received her Ph.D. degree in computer science from Northeastern University, China in 2000. She is currently a senior lecturer in the School of Information Technology, Deakin University, Geelong, Australia. Her current research interests include distributed system, cloud computing, and cyber security.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He has authored over 850 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. Dr. Buyya is one of the highly cited authors in computer science and software engineering worldwide (h-index=151, g-index=331, and 119,400+ citations). Dr. Buyya is recognised as Web of Science "Highly Cited Researcher" for six consecutive years since 2016, IEEE Fellow, Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier, and the "Best of the World" twice for research fields (Computing Systems in 2019 and Software Systems in 2021), by The Australian Research Review. He is also recognised as "Lifetime Achiever" and "Superstar of Research" in "Engineering and Computer Science" discipline twice (2019 and 2021) by the Australian Research. Recently, he received "Research Innovation Award" from IEEE Technical Committee on Services Computing and "Research Impact Award" from IEEE Technical Committee on Cloud Computing. Software technologies for Grid, Cloud, and Fog computing developed under Dr. Buyya's leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 50+ countries around the world. Manjrasoft's Aneka Cloud technology developed under his leadership has received "Frost New Product Innovation Award". He served as founding Editor-in-Chief of the IEEE Transactions on Cloud Computing. He is currently serving as Editor-in-Chief of Software: Practice and Experience, a long standing journal in the field established 50+ years ago. For further information on Dr. Buyya, please visit his cyberhome: www.buyya.com.