PAPER

Dynamic Throughput Allocation among Multiple Servers for Heterogeneous Storage System

Zhisheng HUO[†,††a)], **Limin XIAO**[†,††b)], ***Nonmembers***, **Zhenxue HE**[†††c)], ***Member***, **Xiaoling RONG**[††††d)], ***and*** **Bing WEI**[†,††e)], ***Nonmembers***

SUMMARY Previous works have studied the throughput allocation of the heterogeneous storage system consisting of SSD and HDD in the dynamic setting where users are not all present in the system simultaneously, but those researches make multiple servers as one large resource pool, and cannot cope with the multi-server environment. We design a dynamic throughput allocation mechanism named DAM, which can handle the throughput allocation of multiple heterogeneous servers in the dynamic setting, and can provide a number of desirable properties. The experimental results show that DAM can make one dynamic throughput allocation of multiple servers for making sure users' local allocations in each server, and can provide one efficient and fair throughput allocation in the whole system.

key words: *dynamic throughput allocation, multiple servers, heterogeneous storage system, fair, efficient*

1. Introduction

The fair and efficient throughput allocation of storage system plays a key role on users' performance. While in reality users are not all present in the system simultaneously, how to allocate the throughput of the storage system in dynamic setting is one of current research hotspots. However, the following reasons bring new challenges to the throughput allocation of storage systems in the dynamic setting.

Firstly, SSD is used to improve the performance of storage systems mainly consisting of HDD, which leads to the heterogeneity of storage medium [1], [2].

Secondly, modern datacenters are likely to be constructed from a variety of server classes, with different configurations in term of the throughput capacities of storage servers, namely, there are different throughput capacities between storage servers [3]. For example, Google's storage cluster [4], [5], the same situation has also been observed in Amazon EC2 and Rackspace [6], [7], Facebook [8]. The throughput gap between storage servers means that it is infeasible to treat multiple storage servers as a resource pool.

Lastly, the number of users is sharply increasing, that owes to growing popularity of virtualized data centers hosted on shared physical resources in the storage system, so users' workloads present diversity.

The heterogeneity of storage medium, diversity of users' workloads and throughput gap between servers are intertwined to bring more difficulties to the throughput allocation of multiple heterogeneous servers. There are many allocating ways for users in each server, how to find out one allocation way in each server for users' allocations satisfying the desirable fair properties in the whole system, that is difficult. When introducing the dynamic setting, this problem becomes more complicated, because at each step, after one user enters into the system, the current allocation need satisfy the fair properties for users that are present in the system, the reserved resources need to ensure that the following allocation can meet the desirable fair properties in the multi-server environment for users, who are going to enter the system.

Currently, most studies are static allocation mechanisms based on the proportional fairness [9]–[14] or the disproportional fairness [1], [15]–[17]. Although very few of these methods [1] can cope with the throughput allocation of multiple heterogeneous servers, they cannot adapt to the dynamic setting.

Fortunately, some researchers have proposed the throughput allocation methods of heterogeneous server in the dynamic setting [18], [19]. Although these dynamic methods have been somewhat effective, they cannot deal with the throughput allocation of the multi-server environment. So there are some challenges for the current existing dynamic methods.

The first challenge is how to make one efficient allocation in the environment of multiple servers. The existing dynamic allocation method makes all servers as one large resource pool, there are infinite allocation ways on each server in the process of dynamic allocating, but we do not know which one is efficient.

The second challenge is how to ensure users' allocations enjoying the fair properties in the whole system. The existing dynamic methods take no consideration of the

Manuscript received December 18, 2018.
Manuscript revised April 11, 2019.
Manuscript publicized May 27, 2019.

[†]The authors are with the State Key Laboratory of Software Development Environment, Beihang University, Beijing, 100191 China.
[††]The authors are with the School of Computer Science and Engineering, Beihang University, Beijing, 100191 China.
[†††]The author is with the College of Information Science and Technology, Hebei Agricultural University, Hebei Baoding, 071001 China.
[††††]The author is with the School of Communication Science, Beijing Language and Culture University, Beijing, 100083 China.
a) E-mail: huozhisheng1122@126.com
b) E-mail: xiaolm@buaa.edu.cn
c) E-mail: hezhenxue_2011@163.com
d) E-mail: rong.xiaoling@163.com
e) E-mail: weibing@buaa.edu.cn
DOI: 10.1587/transinf.2018EDP7431

throughput gap between heterogeneous servers, so it is infeasible to treat multiple storage servers as a resource pool. It is difficult to find out one allocation way in each server, that can ensure users' allocations satisfying the desirable fairness properties in the dynamic setting.

To address the above challenges, we design one dynamic throughput allocation method named DAM. Specifically, DAM has two key characteristics as follows.

(1) DAM can handle the throughput allocation of multiple heterogeneous servers to find out an efficient throughput allocation way in each server in the dynamic setting.

(2) DAM can achieve one allocation enjoying the fair properties in the whole system. In the process of dynamic allocating, DAM has taken consideration of the throughput gap between heterogeneous servers, and it jointly conducts the throughput allocation across all heterogeneous servers, so DAM can make users' allocations satisfying the fair properties in the whole system.

The remainder of this paper is organized as follows. In Sect. 2 we discuss the existing allocation methods. We propose DAM and define a set of desirable allocation policies in Sect. 3. In Sect. 4, we introduce the IO requests scheduling framework of DAM. In Sect. 5, we evaluate the performance of DAM in both the simulation and the HDFS cluster. We conclude the paper in Sect. 6.

2. Related Work and Motivation

In this section, we briefly introduce the existing allocation methods, which can be divided into two categories: the static allocation method and the dynamic allocation method.

2.1 Static Allocation Method

2.1.1 Proportional Fair Allocation Method

The resource allocation methods for CPU and bandwidth have been studied [20]–[22], these methods are implemented in the IO scheduling and Qos of storage system [10], [11], [23]–[27]. mClock [28] partitions the resources according to its shares. VirtualFence [29] provides predictable VM performance through space-partitioning of both the SSD cache and the HDD. Pisces [30] can achieve the datacenter-wide per-tenant performance isolation and fairness in shared key-value storage. Extensions of WFQ [31] provide reservations for constant capacity servers. All these models provide strict proportional allocation for a single resource subject to reservation and limit constraints [32]–[35]. However, the above allocation methods is ineffective for multiple kinds of resources.

DRF [12] provided a fair allocation for multiple resources. The extensions of DRF were implemented to the packet networks and the case of indivisible tasks [36], [37]. Dolev et al. [38] proposed an alternative to DRF based on fairly dividing a global system bottleneck resource. Gutman and Nisan [39] considered generalizations of DRF in a more general utility model. Jarett [40] proposed extensions of DRF about the classical bin packing problem. The IO scheduling for SSDs is examined in [41], [42], the data placement and scheduling trade offs of hybrid storage systems were studied in [9]. Reward scheduling making allocations in the ratio of the throughput received by a client is proposed in the multi-tiered storage systems [43]–[45]. The above allocation methods is effective for multiple resources, but due to the constraint of proportional fairness, their resource utilizations are low [12].

DRFH [46] extended DRF from one single server to multiple servers, and provided an allocation of multiple resources with enjoying all fairness properties of DRF [12]. The above method can determine the allocation of client on each server, but its efficiency is low due to the constraint of proportional fairness.

2.1.2 Disproportional Fair Allocation Method

BAA [17] provided a disproportional-fairness allocation for the multi-tiered storage system, and its allocation enjoys the fairness properties of DRF [12]. The above method relaxes the constraint of proportional fairness to improve the resource utilization, but it makes multiple resources as a resource pool, which cannot determine users' local allocations in each server for the environment of multiple servers. UAF [1] is one allocation method based on the disproportional fairness for multiple servers, it can determine users' local allocations in each server.

2.2 Dynamic Allocation Method

The above researches mainly focus on the static allocation, they cannot cope with the dynamic setting that users are not all present in the system simultaneously. In view of this problem, Kash et al. have proposed a dynamic model of fair division, which enjoys the desirable axiomatic properties [18]. Friedman et al. have proposed the optimal recursive mechanism to compute the allocations and provide tight analytic bounds, which have considered arrivals and departures of users [19]. Liu et al. have studied the generalized version of the dynamic multi-resource fair allocation problem, one user will be satisfied when its all tasks submitted can be processed [47].

2.3 Motivation

Existing methods described in the above are somewhat effective for the throughput allocation of the heterogeneous storage system in the environment of multiple servers. However, most of them are the static allocating method, and they assume that all the users are present in the system simultaneously, so these methods cannot tackle the dynamic throughput allocation of heterogeneous storage system in the setting that users enter the system dynamically.

A few researches focus on the dynamic allocation of heterogeneous storage, nevertheless, they conduct the dynamic allocation based on users' dominant resource equal-

ity, which will reduce the utilization of throughput resource. Meanwhile, they cannot cope with the dynamic allocation of multiple servers to determine users' local allocations in each server, due to that they do not consider a multi-service environment.

To address these issues, we propose a dynamic allocation mechanism called DAM in this paper. DAM can make a dynamic allocation for heterogeneous storage based on users' local bottleneck resource equality, and which can improve the utilization of throughput resource in contrast with the above existing methods. Further, DAM studies the dynamic allocation of a multi-server environment, and its allocation policies can cope with the environment of multiple servers, so it can get users' local allocations in each server. Next, we will give a detailed introduction of DAM.

3. DAM Allocation and Its Properties

In this section, we first model the throughput resources of heterogeneous storage system. We next formalize a number of desirable properties for DAM. We last introduce the DAM allocation model in detail.

3.1 Throughput Resource Model

Assume that users arrive at different times and do not depart, user 1 arrives first, then user 2, and in general at step k, user k arrives after users $1, \cdots, k-1$. Let $S = \{1, \cdots, s\}$ be the set of heterogeneous storage servers. Let $C_l = \{C_{l1}, C_{l2}\}$ be HDD and SSD throughput capacities of storage server $l \in S$. Let $U = \{1, \cdots, n\}$ be the user set in the system. Let $R = \{1, 2\}$ be HDD and SSD throughput resources respectively. Let $D_i = \{m_i, h_i\}$ be hit ratios of user $i \in U$ to HDD and SSD, $m_i + h_i = 1$. Let A^k_{ilr} be user i's SSD or HDD throughput allocation in server $l \in S$ at any step k, $r \in R$.

3.2 Desirable Properties

3.2.1 Preliminaries

The fair properties are widely recognized as the most measures in both computer systems and economics, then, we briefly review the preliminaries of sharing incentive (SI), envy freeness (EF), and pareto optimality (PO).

Sharing Incentive. It ensures that every user's throughput allocation is not worse than that obtained by evenly dividing the throughput resource.

Envy Freeness. No user prefers the other's throughput allocation to its own. This property ensures the fairness of the throughput allocation.

Pareto Optimality. A user's throughput allocation cannot be further improved without decreasing the other's throughput allocation, this property is critical to achieve a high resource utilization.

3.2.2 Dynamic Desirable Properties

Some researchers have modified the relevant fair properties in order to adapt to the dynamic setting, for example, DYNAMIC DRF [18] makes the allocation being compatible with EF and PO by relaxing the restriction of EF. However, these dynamic fair properties take no consideration of the multi-server environment, so we generalize SI, EF, and PO from a single server to the environment of multiple servers in the dynamic setting.

(1) Dynamic sharing incentive in the environment of multiple heterogeneous servers. A dynamic allocation mechanism is SI in the environment of multiple servers if $\sum_{l=1}^{s} A^k_{ilr_i^*} \geq \frac{1}{n} \times \sum_{l=1}^{s} C_{lr_i^*}$ ($r_i^* \in R$ represents user i's dominant throughput resource in the remainder of this paper) for all steps k and all users $i \leq k$.

The allocation method based on dominant resources cannot satisfy EF and PO simultaneously in the dynamic setting [18], because in order to satisfy PO property, the current users at step k (step k means that users $1, \ldots, k$ have been present in the system) occupy too much throughput resources, which will decrease the following users $k+1$'s, ..., n's allocations to violate EF fair property. So inspired by the idea [18], we extend EF from one single server to multiple heterogeneous servers in the dynamic setting.

(2) Dynamic envy freeness in the environment of multiple heterogeneous servers. A dynamic allocation mechanism is EF in the multi-server environment if $\frac{\sum_{l=1}^{s} A^k_{ilr_i^*}}{\sum_{l=1}^{s} C_{lr_i^*}} \geq \frac{\sum_{l=1}^{s} A^k_{jlr_j^*}}{\sum_{l=1}^{s} C_{lr_j^*}}$, then user i enters into the system before j, when the inequality is established, $\sum_{l=1}^{s} A^k_{ilr_i^*} = \sum_{l=1}^{s} A^{j-1}_{ilr_i^*}$ for all steps k and all users $i, j \leq k$.

(3) Dynamic pareto optimality in the environment of multiple heterogeneous servers. A dynamic allocation mechanism is PO in the multi-server environment if there exists one allocation A'^k at any step k, for at least one user i, its dominant resource allocation $\sum_{l=1}^{s} A'^k_{ilr_i^*} > \sum_{l=1}^{s} A^k_{ilr_i^*}$, and for any one user j ($j \neq i$) $\sum_{l=1}^{s} A'^k_{jlr_j^*} < \sum_{l=1}^{s} A^k_{jlr_j^*}$, that allocates up to a $(k/n)-$fraction of at least one throughput resource. $A^k_{ilr_i^*}$ is returned by the dynamic allocation mechanism, $A'^k_{ilr_i^*}$ is any other allocation.

3.3 Allocation Policies

In the dynamic setting, the allocation method proposed in this paper does not withdraw the allocated resources for users to reallocate, at any step k, it takes out a $1/n$ share of each throughput resource, and takes back the throughput resources that users $1, \ldots, k-1$ cannot use at step $k-1$. DAM starts from the current allocation of step $k-1$ to allocate the resource taken out and back for users $1, \ldots, k$ that have the minimum dominant share at the same rate, just as the static allocation method DRF does [12]. So we propose the following allocation policies, which are based on the above dynamic SI, EF and PO properties generalized from one single server to multiple heterogeneous servers.

(1) For any one user i at any step k, user i's dominant

resource allocation $\sum_{l=1}^{s} A_{ilr_i^*}^k \geq \frac{1}{n} \times \sum_{l=1}^{s} C_{lr_i^*}$.

(2) For any one user i at any step k, $\sum_{l=1}^{s} A_{ilr_i^*}^k \geq \sum_{l=1}^{s} A_{ilr_i^*}^{k-1}$.

(3) For any two users i and j at any step k, if $j > i$, then $\frac{\sum_{l=1}^{s} A_{ilr_i^*}^k}{\sum_{l=1}^{s} C_{lr_i^*}} \geq \frac{\sum_{l=1}^{s} A_{jlr_j^*}^k}{\sum_{l=1}^{s} C_{lr_j^*}}$.

3.4 DAM Allocation Model

DAM resides inside the heterogeneous storage system, and make dynamic allocations based on users' workloads, with determining users' local allocations in each server. Meanwhile, when new users enter the system, DAM can get a fair and efficient allocation for users, who has been present in the system, and it reserves the throughput resource for subsequent users who will enter the system. Finally, DAM can achieve one fair and efficient allocation when all users have entered into the system. Specifically, at step k, users $1, \ldots, k$ ($k \leq n$) are present in the system, the current HDD and SSD throughput capacities are set to the (k/n) share of each throughput resource that can be provided by the system. According to the above allocation policies, we propose the following allocation model.

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{k} \frac{\sum_{l=1}^{s} A_{ilr_i^*}^k}{\sum_{l=1}^{s} C_{lr_i^*}} \\
\text{s.t.} \quad & \frac{\sum_{l=1}^{s} A_{ilr_i^*}^k}{\sum_{l=1}^{s} C_{lr_i^*}} \geq \frac{\sum_{l=1}^{s} A_{jlr_j^*}^k}{\sum_{l=1}^{s} C_{lr_j^*}}, i < j \leq k \\
& \sum_{i=1}^{k} \sum_{l=1}^{s} A_{ilr}^k \leq \frac{k}{n} \times \sum_{l=1}^{s} C_{lr}, r \in R \\
& \sum_{l=1}^{s} A_{ilr_i^*}^k \geq \frac{1}{n} \times \sum_{l=1}^{s} C_{lr_i^*}, i \leq k \\
& \sum_{l=1}^{s} A_{ilr_i^*}^k \geq \sum_{l=1}^{s} A_{ilr_i^*}^{k-1}, i \leq k-1
\end{aligned}
$$

Example We make use of one example to demonstrate the allocating process of DAM. Suppose that there are two servers in the heterogeneous storage system, and their throughput capacities are S_1(SD: 300 IOPS, HD: 200 IOPS) and S_2(SD: 1800 IOPS, HD: 1300 IOPS) respectively. Meanwhile, there exists three users accessing the system. In the following formula, A_{ij} represents user i's allocations in server j. At step 1, user 1 enters into the system with its hit ratio $h_1 = 0.5$, according to the allocation model presented by the above formula, we can get user 1's allocations: A_{11}(SD: 71.4 IOPS, HD: 66.7 IOPS), A_{12}(SD: 428.6 IOPS, HD: 433.7 IOPS); at step two, with user 2 entering the system with hit ratio $h_2 = 0.9$, DAM can get the allocations of users 1 and 2 through the above model: A_{11}(SD: 134.9 IOPS, HD: 116.5 IOPS), A_{12}(SD: 809.1 IOPS, HD: 757.5 IOPS), A_{21}(SD: 116.6 IOPS, HD: 12 IOPS), A_{22}(SD: 699.4 IOPS, HD: 78 IOPS); at step 3, user 3 goes into the system with hit ratio $h_3 = 0.8$, we can get three users' final allocation by making use of the above model: A_{11}(SD: 134.9 IOPS, HD: 116.5 IOPS), A_{12}(SD: 809.1 IOPS, HD: 757.5 IOPS), A_{21}(SD: 116.6 IOPS, HD: 12 IOPS), A_{22}(SD: 699.4 IOPS, HD: 78 IOPS), A_{31}(SD: 100 IOPS, HD: 23.3 IOPS), A_{32}(SD: 600 IOPS, HD: 151.7 IOPS). From the above allocating process, we can know that DAM can cope with the dynamic allocation with users entering the system dynamically, and can deal with the environment of multiple heterogeneous servers.

4. DAM Implementing Framework

As shown in Fig. 1, we use one two-tier scheduling framework to dispatch IO requests to multiple servers. The first tier scheduler sends the IO requests to the target of storage server, DAM resides in the first tier scheduler, and recomputes the throughput allocations of users in every server periodically.

The second tier scheduler locates in each server, DAM can get users' throughput allocations in each server in the first scheduling tier, these allocations make up users' weights to a proportional-share scheduler like WFQ in the second tier. The second tier scheduler takes over the IO request scheduling function of linux in each server, and dispatches IO requests according to users' weights. Algorithm 1 describes the IO request dispatching process of DAM.

With users entering the system, the two-tier IO scheduler will recompute users' throughput allocations; meanwhile, as users' workloads change, it recomputes users' allocations periodically to maintain a higher resource utilization. This two-tier scheduler framework includes one module, which monitors hit ratios of users over a moving window of requests in the system.

As the system runs, when one or some users leave, the number of users has changed; however, DAM will perform a throughput allocation based on users' workloads in a fixed time window, in which, the number of users is fixed. With users' departures, the system enters a new state, and the number of users will be fixed again. DAM will recalculate the throughput allocation after the number of users is fixed again.

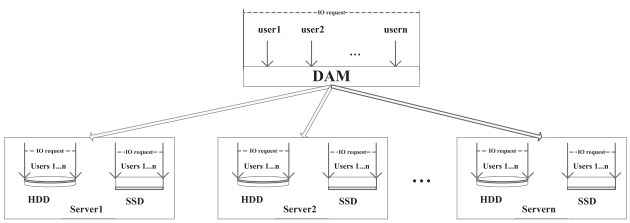


Fig. 1 DAM implementing framework

Algorithm 1 IO request scheduling using DAM

1: Users statistic their hit ratios periodically.
2: DAM recomputes users' local allocations in each server.
3: In each server, DAM dispatches IO requests according to the weights made up from users' allocations.
4: Repeat the above steps periodically.

5. Experimental Result

We evaluate DAM in both PFSsim and the HDFS cluster, and implement DAM into the IO path, so the experiment includes two parts: simulation experiment, evaluation in the real environment. The raw IO is performed to eliminate the the influence of OS buffer caching.

5.1 PFSsim Evaluation

PFSsim is a simulator of file system, it contains SSD and HDD modules. We ran PFSsim in one Linux machine with AMD Quad-core processors, 8GB RAM, 1TB Seagate 7200RPM hard drive. We configure three servers in PFSsim, and the throughput capacities of these servers are described in Table 1. The synthetic workload set [48] is a file system and storage benchmark that allows to generate a variety of workloads, we choose the webserver as it is the intensive workload, we distribute the trace among three servers to produce the corresponding workloads in advance.

In this experimental part, we experimentally answer the following questions: (1) Can DAM cope with the throughput allocation of the multi-server environment in the dynamic setting? (2) Does DAM satisfy SI, PO and EF properties for the multi-server environment in the dynamic setting?

First, user 1 enters the system with hit ratio $h_1 = 0.5$. Figure 2 (a) shows user 1's allocation among three servers. Next, user 2 comes into the system, its hit ratio is $h_2 = 0.9$, users 1's and 2's allocations are as shown in Fig. 2 (b). At last, user 3 gets into the system with hit ratio $h_3 = 0.8$. Figure 2 (c) shows three users' allocations among servers. As shown in Fig. 2 (a)–(c), as users enter into the system at each step, DAM can determine users' local allocations in each server, so DAM can cope with the throughput allocation of multiple heterogeneous servers in the dynamic setting.

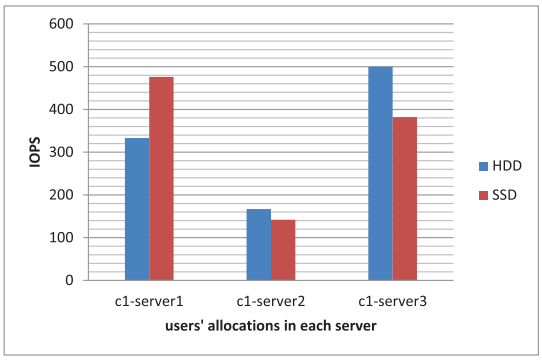
From Fig. 2 (a)–(c), we can get that at step 1, user 1's dominant throughput allocation is $\sum_{l=1}^{3} A_{1l1}^{1} = \frac{\sum_{l=1}^{3} C_{l2}}{3} = 500$ IOPS. At step 2, the dominant throughput allocations of users 1, 2 satisfy $\sum_{l=1}^{3} A_{1l1}^{2} = 1166 > \frac{\sum_{l=1}^{3} C_{l1}}{3} = 500$ IOPS, $\sum_{l=1}^{3} A_{2l2}^{2} = 1633 > \frac{\sum_{l=1}^{3} C_{l2}}{3} = 1400$ IOPS respectively. At step 3, the dominant throughput allocations of three users are $\sum_{l=1}^{3} A_{1l1}^{3} = 1166 > \frac{\sum_{l=1}^{3} C_{l1}}{3} = 500$ IOPS, $\sum_{l=1}^{3} A_{2l2}^{3} = 1633 > \frac{\sum_{l=1}^{3} C_{l2}}{3} = 1400$ IOPS, and $\sum_{l=1}^{3} A_{3l2}^{3} = \frac{\sum_{l=1}^{3} C_{l2}}{3} = 1400$ IOPS respectively. So we can get one conclusion that DAM satisfies SI at each step in the dynamic setting.

Figure 2 (a)–(c) show that for any two users i and j, if $i < j$, then $\frac{\sum_{l=1}^{3} A_{ilr_i^*}^{k}}{\sum_{l=1}^{3} C_{lr_i^*}} \geq \frac{\sum_{l=1}^{3} A_{jlr_j^*}^{k}}{\sum_{l=1}^{3} C_{lr_j^*}}$ at any step k, we can get that DAM satisfies EF for the environment of multiple heterogeneous servers in the dynamic setting.

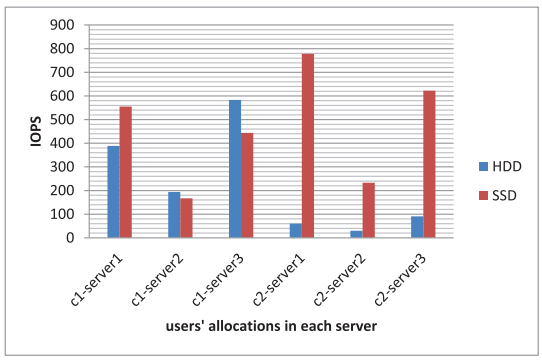
As shown in Fig. 2 (a)–(c), at any step k, DAM allocates up to a $(k/n)-$fraction of at least one throughput resource, so DAM satisfies PO.

Table 1 Configurations of servers

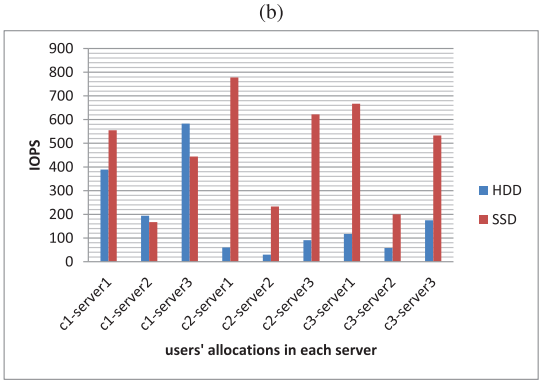
Server	HDD (IOPS)	SSD (IOPS)
$Server_1$	1000	2000
$Server_2$	500	600
$Server_3$	1500	1600



(a)



(b)



(c)

Fig. 2 Users' allocations using DAM

5.2 HDFS Experiment

We evaluate DAM in HDFS, and compare its behavior with DYNAMIC DRF and DRFH. This part of experiments run on Linux servers with AMD Quad-core processors, 8GB RAM, 1TB Seagate 7200RPM hard drive, and 120GB SAMSUNG 840 Pro Series SSDs, SSD is used as the main storage in the system. In order to simulate the throughput gap between server, the number of SSD and HDD is

different between servers. The evaluation workloads are from Umass Trace [49] and Microsoft Exchange server [50]. These traces are for a homogeneous server, and do not distinguish between HDD and SSD in the heterogeneous storage server, so we reform them to distribute on both SSD and HDD in different storage servers to produce the corresponding workloads.

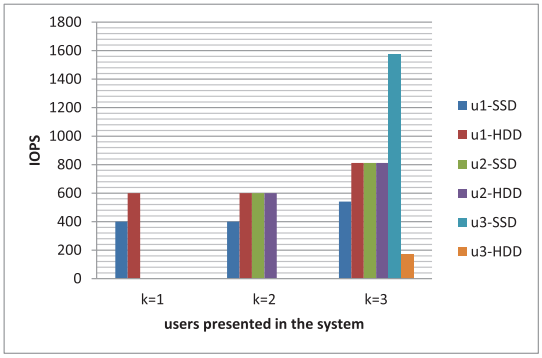
In reality, the storage system contains a huge number of servers, and there are a large number of users accessing the system. In our experiment, we set up three servers and three users with the following considerations. First, our experimental goal is that whether DAM can cope with the environment of multiple heterogeneous servers in the process of dynamic allocation, this has nothing to do with the number of servers and users. Three servers and three uses are enough to to verify the effectiveness of our method, namely, DAM can handle the dynamic allocation in the environment of multiple servers. Second, our experimental goal also test and verify that whether DAM can handle the allocation in the dynamic setting, where users enter the system dynamically, this is not related to the number of users and servers.

5.2.1 Compare DAM with DYNAMIC DRF

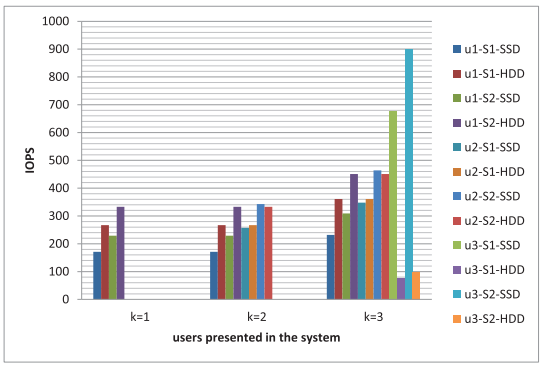
For the simplicity of the experiment, we configurate two data servers in the HDFS cluster, there are three users with their hit ratios $h_1 = 0.4$, $h_2 = 0.5$, $h_3 = 0.9$ respectively. Users 1 and 2 run two financial workloads from Umass Trace; user 3 runs one exchange workload from Microsoft Exchange server workload. These workloads have a mix of read and write requests and request sizes range from 512 bytes to 8MB.

When running the above workloads, the throughput capacity of HDD is around 700 ∼ 1000 IOPS, and the SSD is 1000 ∼ 2000 IOPS. As shown in Fig. 3 (a)–(b), both of DAM and DYNAMIC DRF can deal with the throughput allocation of the dynamic setting. Figure 3 (a) shows the allocation of DYNAMIC DRF, it makes three servers as one resource pool, and cannot cope with the throughput allocation of the environment of multiple heterogeneous servers. As shown in Fig. 3 (b), compared with DYNAMIC DRF, DAM can determine users' local allocations in each server, because it considers the throughput gap between heterogeneous servers, and jointly allocates the throughput across multiple servers.

As shown in Fig. 3 (a)–(b), at step 1, the HDD and SSD throughput utilizations of DYNAMIC DRF are $\frac{A^1_{11}}{\frac{1}{3}\times\sum_{l=1}^{2} C_{l1}} = \frac{600}{600} = 100\%$, $\frac{A^1_{12}}{\frac{1}{3}\times\sum_{l=1}^{2} C_{l2}} = \frac{400}{1167} = 34\%$ respectively, that of DAM are $\frac{\sum_{l=1}^{2} A^1_{1l1}}{\frac{1}{3}\times\sum_{l=1}^{2} C_{l1}} = \frac{600}{600} = 100\%$, $\frac{\sum_{l=1}^{2} A^1_{1l2}}{\frac{1}{3}\times\sum_{l=1}^{2} C_{l2}} = \frac{400}{1167} = 34\%$ respectively; at step 2, the resource utilizations of DYNAMIC DRF are $\frac{\sum_{i=1}^{2} A^2_{i1}}{\frac{2}{3}\times\sum_{l=1}^{2} C_{l1}} = \frac{1200}{1200} = 100\%$, $\frac{\sum_{i=1}^{2} A^2_{i2}}{\frac{2}{3}\times\sum_{l=1}^{2} C_{l2}} = \frac{1000}{2333} = 43\%$ respectively, that of DAM have the same resource utilizations: $\frac{\sum_{i=1}^{2}\sum_{l=1}^{2} A^2_{il1}}{\frac{2}{3}\times\sum_{l=1}^{2} C_{l1}} = \frac{267+333+267+333}{1200} = 100\%$,



(a)



(b)

Fig. 3 Evaluate DYNAMIC DRF and DAM

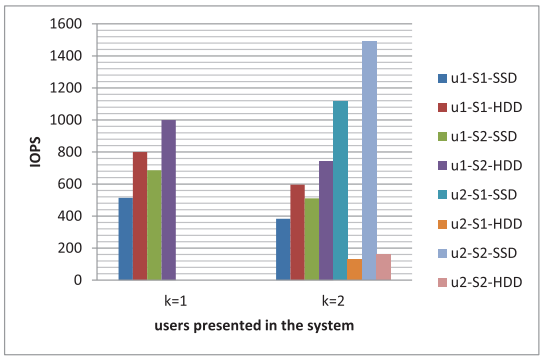
$\frac{\sum_{i=1}^{2}\sum_{l=1}^{2} A^2_{il2}}{\frac{2}{3}\times\sum_{l=1}^{2} C_{l2}} = \frac{171+229+257+343}{2333} = 43\%$; at step 3, the resource utilizations of DYNAMIC DRF are $\frac{\sum_{i=1}^{3} A^3_{i1}}{\sum_{l=1}^{2} C_{l1}} = \frac{812+812+176}{1800} = 100\%$, $\frac{\sum_{i=1}^{3} A^3_{i2}}{\sum_{l=1}^{2} C_{l2}} = \frac{541+812+1579}{3500} = 84\%$ respectively, that of DAM are $\frac{\sum_{i=1}^{3}\sum_{l=1}^{2} A^3_{il1}}{\sum_{l=1}^{2} C_{l1}} = \frac{361+451++361+451+78+98}{1800} = 100\%$, $\frac{\sum_{i=1}^{3}\sum_{l=1}^{2} A^3_{il2}}{\sum_{l=1}^{2} C_{l2}} = \frac{232+309+348+464+677+902}{3500} = 84\%$ respectively.

So we can get that DAM has the same resource utilization with DYNAMIC DRF, but DAM can handle the throughput allocation of multiple heterogeneous servers compared with DYNAMIC DRF that cannot.

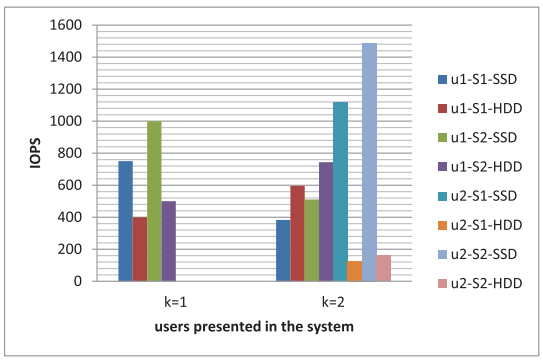
5.2.2 Compare DAM with DRFH

In this experimental part, we configurate two data servers in the system. There are two uses with their hit ratios $h_1 = 0.4$ and $h_2 = 0.9$ respectively. User 1 runs one financial workload from Umass Trace, user 2 runs one exchange workload from Microsoft Exchange server. When running these workloads, these two servers can provide HDD and SSD throughput capacities of 760 ∼ 1000 IOPS and 1400 ∼ 2000 IOPS respectively.

Figure 4 (a)–(b) show the allocations of users using DRFH and DAM. As shown in Fig. 4, we can know that although both of methods can cope with the throughput allocation of the environment of multiple heterogeneous servers, DRFH cannot adapt to the throughput allocation of the dynamic setting. Figure 4 (a) shows that when using



(a)



(b)

Fig. 4 Evaluate DAM and DRFH

DRFH, after user 1 enters the system, the HDD throughput achieves 100% utilization, when user 2 getting into the system, DRFH revokes user 1's allocation to reallocate the throughput resource for users 1 and 2, then, user 1's HDD allocation is reduced from 1800 IOPS to 1340 IOPS, its SSD is reduced from 1200 IOPS to 894 IOPS, and the reallocating process will suspend user 1 accessing the system until finishing the reallocation. However, Fig. 4 (b) shows that compared with DRFH, DAM can handle the throughput allocation of the dynamic setting, it first allocates some throughput to user 1, after user 2 entering the system, DAM will allocate the remaining throughput to users 1and 2 without interrupting user 1's operation.

Next, we compare the resource utilization of DAM with DRFH. Because DRFH is one kind of static allocation methods, when two users have entered into the system, we compare resource utilizations of these two methods. As shown in Fig. 4 (a)–(b), HDD and SSD throughput utilizations of DAM are same with that of DRFH: $\text{DAM}_{HD}=\text{DRFH}_{HD}=\frac{596+744+129+161}{1800}=91\%$, $\text{DAM}_{SD}=\text{DRFH}_{SD}=\frac{383+511+1117+1489}{3500}=100\%$. So we can get one conclusion that DAM not only can achieve the same utilization with DRFH, but also it can handle the throughput allocation of the dynamic setting compared with DRFH that cannot.

6. Conclusion

We have introduced one allocation model named DAM, which can handle the throughput allocation of the environment of multiple heterogeneous servers in dynamic setting. DAM satisfies SI, PO and EF properties. We have evaluated the performance of DAM using implementation in both the simulator and the HDFS cluster. The experimental results show that DAM is fair and efficient allocation method compared with the existing method.

DAM is based on users' dominant resources to conduct one dynamic allocation, which will prevent the throughput resource utilization from going any higher. Our future work studies the dynamic allocation method based on users' local bottleneck resources to improve the resource utilization.

Acknowledgments

This version has benefited greatly from the anonymous reviewers. The authors gratefully acknowledge these comments and suggestions. This work is supported by the National Key Research and Development Program of China under Grant (NO. 2018YFB0203901, NO. 2018YFB0203900), the China Postdoctoral Science Foundation under Grant No. 2018M641154, the Introducing Talent Research Project of Agricultural University of Hebei under Grant No. YJ201829, the National Natural Science Foundation of China (No.61370059, No.61772053), the fund of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2017ZX-10.

References

[1] Z. Huo, L. Xiao, Q. Zhong, S. Li, A. Li, R. Li, K. Liu, Y. Zang, P. Wang, and Z. Lu, "Hybrid storage throughput allocation among multiple clients in heterogeneous data center," IEEE International Conference on High PERFORMANCE Computing and Communications, 2015 IEEE International Symposium on Cyberspace Safety and Security, and 2015 IEEE International Conf on Embedded Software and Systems, pp.140–147, 2015.

[2] Z. Huo, L. Xiao, Q. Zhong, S. Li, A. Li, R. Li, S. Wang, and L. Fu, A Metadata Cooperative Caching Architecture Based on SSD and DRAM for File Systems, Springer International Publishing, 2015.

[3] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," Commun. ACM, vol.53, no.4, pp.50–58, April 2010.

[4] C. Reiss, A. Tumanov, G.R. Ganger, R.H. Katz, and M.A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," Proc. Third ACM Symposium on Cloud Computing, SoCC '12, New York, NY, USA, pp.7:1–7:13, ACM, 2012.

[5] C. Reiss, J. Wilkes, and J.L. Hellerstein, "Google cluster-usage traces: format+ schema," White Paper, Google Inc., 2011.

[6] B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K.D. Bowers, and M.M. Swift, "More for your money: exploiting performance heterogeneity in public clouds," Proc. Third ACM Symposium on Cloud Computing, p.20, ACM, 2012.

[7] Z. Ou, H. Zhuang, A. Lukyanenko, J. Nurminen, P. Hui, V. Mazalov, and A. Yla-Jaaski, "Is the same instance type created equal? Exploiting heterogeneity of public clouds," 2013.

[8] T. Harter, D. Borthakur, S. Dong, A.S. Aiyer, L. Tang, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau, "Analysis of HDFS under hbase: a facebook messages case study," FAST, pp.199–212, 2014.

[9] X. Wu and A.L.N. Reddy, "Exploiting concurrency to improve latency and throughput in a hybrid storage system," 2010 IEEE Inter-

national Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pp.14–23, IEEE, 2010.
[10] Y. Wang and A. Merchant, "Proportional-share scheduling for distributed storage systems," FAST, p.4, 2007.
[11] A. Gulati, A. Merchant, and P.J. Varman, "pclock: an arrival curve based approach for qos guarantees in shared storage systems," ACM SIGMETRICS Performance Evaluation Review, vol.35, no.1, pp.13–24, 2007.
[12] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," NSDI, p.24, 2011.
[13] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multiresource allocation: fairness-efficiency tradeoffs in a unifying framework," IEEE/ACM Trans. Netw., vol.21, no.6, pp.1785–1798, 2013.
[14] W. Wang, B. Liang, and B. Li, "Multi-resource fair allocation in heterogeneous cloud computing systems," IEEE Trans. Parallel Distrib. Syst., vol.26, no.10, pp.2822–2835, 2015.
[15] J. Li and J. Xue, "Egalitarian division under leontief preferences," Economic Theory, vol.54, no.3, pp.597–622, 2013.
[16] A.D. Procaccia, "Cake cutting: not just child's play," Commun. ACM, vol.56, no.7, pp.78–87, 2013.
[17] H. Wang and P.J. Varman, "Balancing fairness and efficiency in tiered storage systems with bottleneck-aware allocation," FAST, pp.229–242, 2014.
[18] I. Kash, A.D. Procaccia, and N. Shah, "No agent left behind: dynamic fair division of multiple resources," International Conference on Autonomous Agents and Multi-Agent Systems, pp.351–358, 2013.
[19] E. Friedman, C.-A. Psomas, and S. Vardi, "Dynamic fair division with minimal disruptions," Sixteenth ACM Conference on Economics and Computation, pp.697–713, 2015.
[20] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," ACM SIGCOMM Computer Communication Review, pp.1–12, ACM, 1989.
[21] P. Goyal, H.M. Vin, and H. Chen, "Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks," ACM SIGCOMM Computer Communication Review, pp.157–168, ACM, 1996.
[22] C.A. Waldspurger and W.E. Weihl, "Lottery scheduling: Flexible proportional-share resource management," Proc. 1st USENIX Conference on Operating Systems Design and Implementation, p.1, USENIX Association, 1994.
[23] A. Gulati, I. Ahmad, C.A. Waldspurger, *et al.*, "Parda: Proportional allocation of resources for distributed storage access," FAST, pp.85–98, 2009.
[24] A. Gulati, C. Kumar, I. Ahmad, and K. Kumar, "Basil: Automated io load balancing across storage devices," FAST, p.13, 2010.
[25] C.R. Lumb, A. Merchant, and G.A. Alvarez, "Façade: Virtual storage devices with performance guarantees," FAST, pp.131–144, 2003.
[26] W. Jin, J.S. Chase, and J. Kaur, "Interposed proportional sharing for a storage service utility," ACM SIGMETRICS Performance Evaluation Review, pp.37–48, ACM, 2004.
[27] C.R. Lumb, J. Schindler, G.R. Ganger, D.F. Nagle, and E. Riedel, "Towards higher disk head utilization: extracting free bandwidth from busy disk drives," Proc. 4th Conference on Symposium on Operating System Design & Implementation, vol.4, p.7, USENIX Association, 2000.
[28] A. Gulati, A. Merchant, and P.J. Varman, "mclock: Handling throughput variability for hypervisor io scheduling," Proc. 9th USENIX Conference on Operating Systems Design and Implementation, pp.1–7, 2013.
[29] C. Li, Í. Goiri, A. Bhattacharjee, R. Bianchini, and T.D. Nguyen, "Quantifying and improving i/o predictability in virtualized systems," Ieee/acm International Symposium on Quality of Service, pp.1–6, 2013.
[30] D. Shue, M.J. Freedman, and A. Shaikh, "Performance isolation and fairness for multi-tenant cloud storage," Usenix Conference on Operating Systems Design and Implementation, pp.349–362, 2012.
[31] I. Stoica, H. Abdel-Wahab, and K. Jeffay, "Duality between resource reservation and proportional share resource allocation," Electronic Imaging'97, pp.207–214, International Society for Optics and Photonics, 1997.
[32] M. Karlsson, C. Karamanolis, and X. Zhu, "Triage: Performance differentiation for storage systems using adaptive control," ACM Transactions on Storage (TOS), vol.1, no.4, pp.457–480, 2005.
[33] T.M. Wong, R.A. Golding, C. Lin, and R.A. Becker-Szendy, "Zygaria: Storage performance as a managed resource," Proc. 12th IEEE Real-Time and Embedded Technology and Applications Symposium, 2006, pp.125–134, IEEE, 2006.
[34] A. Gulati, A. Merchant, and P.J. Varman, "mclock: handling throughput variability for hypervisor io scheduling," Proc. 9th USENIX Conference on Operating Systems Design and Implementation, pp.1–7, USENIX Association, 2010.
[35] A. Gulati, G. Shanmuganathan, X. Zhang, and P.J. Varman, "Demand based hierarchical qos using storage resource pools," USENIX Annual Technical Conference, pp.1–13, 2012.
[36] A. Ghodsi, V. Sekar, M. Zaharia, and I. Stoica, "Multi-resource fair queueing for packet processing," ACM SIGCOMM Computer Communication Review, vol.42, no.4, pp.1–12, 2012.
[37] D.C. Parkes, A.D. Procaccia, and N. Shah, "Beyond dominant resource fairness: extensions, limitations, and indivisibilities," Proc. 13th ACM Conference on Electronic Commerce, pp.808–825, ACM, 2012.
[38] D. Dolev, D.G. Feitelson, J.Y. Halpern, R. Kupferman, and N. Linial, "No justified complaints: On fair sharing of multiple resources," Proc. 3rd Innovations in Theoretical Computer Science Conference, pp.68–75, ACM, 2012.
[39] A. Gutman and N. Nisan, "Fair allocation without trade," Proc. 11th International Conference on Autonomous Agents and Multiagent Systems, vol.2, pp.719–728, International Foundation for Autonomous Agents and Multiagent Systems, 2012.
[40] C.A. Psomas and J. Schwartz, "Beyond beyond dominant resource fairness: Indivisible resource allocation in clusters," Tech. Report Berkeley, 2013.
[41] S. Park and K. Shen, "Fios: A fair, efficient flash i/o scheduler," FAST, p.13, 2012.
[42] K. Shen and S. Park, "Flashfq: A fair queueing i/o scheduler for flash-based ssds," USENIX Annual Technical Conference, pp.67–78, 2013.
[43] A. Elnably, K. Du, and P. Varman, "Reward scheduling for qos in cloud applications," 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp.98–106, IEEE, 2012.
[44] A. Elnably and P. Varman, "Application-sensitive qos scheduling in storage servers," ACM Symposium on Parallelism in Algorithms and Architecture, 2012.
[45] A. Elnably, H. Wang, A. Gulati, and P. Varman, "Efficient qos for multi-tiered storage systems," 4th USENIX Workshop on Hot Topics in Storage and File Systems, 2012.
[46] W. Wang, B. Liang, and B. Li, "Multi-resource fair allocation in heterogeneous cloud computing systems,"
[47] X. Liu, X. Zhang, X. Zhang, and W. Li, "Dynamic fair division of multiple resources with satiable agents in cloud computing systems," IEEE Fifth International Conference on Big Data and Cloud Computing, pp.131–136, 2015.
[48] R. McDougall and J. Mauro, "Filebench," URL: http://www.nfsv4bat.org/Documents/nasconf/2004/filebench.pdf (cited on page 56), 2005.
[49] S.P. Council, "Spc i/o traces," 2009.
[50] S. Kavalanekar, B. Worthington, Q. Zhang, and V. Sharda, "Characterization of storage workload traces from production windows servers," IEEE International Symposium on Workload Characteri-

zation, 2008, IISWC 2008, pp.119–128, IEEE, 2008.

Appendix: Desirable properties proof

Now we proof that DAM satisfies SI, EF and PO properties for the environment of multiple heterogeneous servers in the dynamic setting.

Lemma 1: The DAM allocation is sharing incentive.

Proof 1: If DAM enjoys SI, for any one user i at any step k, its dominant resource allocation $\sum_{l=1}^{s} A_{ilr_i^*}^{k} \geq \frac{1}{n} \times \sum_{l=1}^{s} C_{lr_i^*}$. When $k = 1$, use 1's dominant resource allocation $\sum_{l=1}^{s} A_{1lr_1^*}^{1} = \frac{1}{n} \times \sum_{l=1}^{s} C_{lr_1^*}$. For the case $k = 2$, after user 2 enters the system, user 2 will be allocated the throughput resource until its dominant resource allocation $\frac{\sum_{l=1}^{s} A_{2lr_2^*}^{2}}{\sum_{l=1}^{s} C_{lr_2^*}} = \frac{\sum_{l=1}^{s} A_{1lr_1^*}^{1}}{\sum_{l=1}^{s} C_{lr_1^*}}$, then, if there are free SSD and HDD throughput, users 1 and 2 will be given the throughput resource at the same rate, until that up to a $(2/n)-$fraction of at least one resource, so their dominant resource allocations are higher than $\frac{1}{n} \times \sum_{l=1}^{s} C_{lr_1^*}$ and $\frac{1}{n} \times \sum_{l=1}^{s} C_{lr_2^*}$ respectively.

Assume that this is true at step $k-1$, for any one user i, $\sum_{l=1}^{s} A_{ilr_i^*}^{k-1} \geq \frac{1}{n} \times \sum_{l=1}^{s} C_{lr_i^*}$. At step k, the worst case is that when user k's dominant resource is given $\frac{1}{n} \times \sum_{l=1}^{s} C_{lr_k^*}$, the allocation of throughput resource r_k^* ($r_k^* \in R$) is up to $\frac{k}{n} \times \sum_{l=1}^{s} C_{lr_k^*}$, we can get one feasible solution, for any one user i ($i < k$), $\sum_{l=1}^{s} A_{ilr_i^*}^{k} = \sum_{l=1}^{s} A_{ilr_i^*}^{k-1} \geq \frac{1}{n} \times \sum_{l=1}^{s} C_{lr_i^*}$, for user k, $\sum_{l=1}^{s} A_{klr_k^*}^{k} = \frac{1}{n} \times \sum_{l=1}^{s} C_{lr_k^*}$. So we can get that for any one user i ($i \leq k$), its dominant resource allocation $\sum_{l=1}^{s} A_{ilr_i^*}^{k} \geq \frac{1}{n} \times \sum_{l=1}^{s} C_{lr_i^*}$ at any step k. DAM satisfies SI for the environment of multiple heterogeneous servers in the dynamic setting.

Lemma 2: The DAM allocation is pareto optimality.

Proof 2: In order to maximize the utilization of throughput resource, at each step k, for users $1, \ldots, k$, their allocations are up to a $(k/n)-$fraction of at least one throughput resource: $\frac{k}{n} \times \sum_{l=1}^{s} C_{lr}$ ($r \in R$); otherwise, they can improve their allocations, which contradicts the objective function of LP. So this means that DAM satisfies PO for the environment of multiple heterogeneous servers in the dynamic setting.

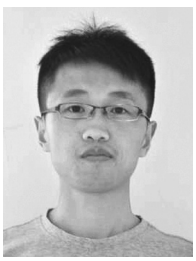
Lemma 3: The DAM allocation is envy freeness.

Proof 3: For any two users $i, j \in U$ such that $i < j$ at any step k. When $i \leq k < j$, user j's dominant resource allocation $\sum_{l=1}^{s} A_{jlr_j^*}^{k} = 0$, so $\sum_{l=1}^{s} A_{ilr_i^*}^{k} > \sum_{l=1}^{s} A_{jlr_j^*}^{k}$. For the case $i < j \leq k$, according to the first constraint of the LP, we can know that $\frac{\sum_{l=1}^{s} A_{ilr_i^*}^{k}}{\sum_{l=1}^{s} C_{lr_i^*}} > \frac{\sum_{l=1}^{s} A_{jlr_j^*}^{k}}{\sum_{l=1}^{s} C_{lr_j^*}}$.

Assume that if $\frac{\sum_{l=1}^{s} A_{ilr_i^*}^{k}}{\sum_{l=1}^{s} C_{lr_i^*}} > \frac{\sum_{l=1}^{s} A_{jlr_j^*}^{k}}{\sum_{l=1}^{s} C_{lr_j^*}}$, then $i < j$ and $\sum_{l=1}^{s} A_{ilr_i^*}^{k} = \sum_{l=1}^{s} A_{ilr_i^*}^{j-1}$. Suppose for contradiction $\sum_{l=1}^{s} A_{ilr_i^*}^{k} > \sum_{l=1}^{s} A_{ilr_i^*}^{j-1}$, this means that these exits one step t, $\sum_{l=1}^{s} A_{ilr_i^*}^{t} > \sum_{l=1}^{s} A_{ilr_i^*}^{t-1}$. When $t \leq j$, $\frac{\sum_{l=1}^{s} A_{ilr_i^*}^{t}}{\sum_{l=1}^{s} C_{lr_i^*}} = \frac{\sum_{l=1}^{s} A_{jlr_j^*}^{t}}{\sum_{l=1}^{s} C_{lr_j^*}}$. Extend step t to k, $\frac{\sum_{l=1}^{s} A_{ilr_i^*}^{k}}{\sum_{l=1}^{s} C_{lr_i^*}} = \frac{\sum_{l=1}^{s} A_{jlr_j^*}^{k}}{\sum_{l=1}^{s} C_{lr_j^*}}$, this contradicts our assumption. So DAM satisfies EF for the environment of multiple heterogeneous servers in the dynamic setting.

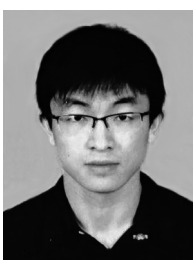
Zhisheng Huo is a Post-doctoral candidate of School of Computer Science and Technology, Beihang University. His research interests include big data storage and distributed storage system.

Limin Xiao is a professor of School of Computer Science and Engineering, Beihang University. His research interests include computer architecture, high performance computing and computer system software.



Zhenxue He is an associate professor of Hebei Agricultural University. His research interests include low power integrated circuit design and optimization, multiple-valued logic circuits and computer aided design. He is a member of ACM and China Computer Federation.

Xiaoling Rong is currently working toward the master's degree in School of Communication Science, Beijing Language and Culture University. Her research interests include children language processing and children language disorder.



Bing Wei is currently pursuing a Ph.D. degree in computer science at Beihang University. His main research interests include distributed file systems, high performance computing, software engineering, and clusters.