MLF-DRS: A Multi-level Fair Resource Allocation Algorithm in Heterogeneous Cloud Computing Systems

Hamed Hamzeh
Smart Technology Research Group (STRG)
Bournemouth University
Bournemouth, United Kingdom
e-mail: hamzehh@bournemouth.ac.uk

Sofia Meacham
Smart Technology Research Group
Bournemouth University
Bournemouth, United Kingdom
e-mail: smeacham@bournemouth.ac.uk

Botond Virginas
British Telecom
Ipswich, United Kingdom
e-mail: Botond.virginas@bt.com

Kashaf Khan
British Telecom
Ipswich, United Kingdom
e-mail: Kashaf.khan@bt.com

Keith Phalp
Smart Technology Research Group (STRG)
Bournemouth University
Bournemouth, United Kingdom
e-mail: kphalp@bournemouth.ac.uk

***Abstract*—Cloud computing is a novel paradigm which provides on demand, scalable and pay-as-you-use computing resources in a virtualized form. With cloud computing, users are able to access large pools of resources anywhere without any limitation. In order to use the provided facilities by the cloud in an efficient way, the management of resources is an undeniable fact that should be considered in different aspects. Among all those aspects, resource allocation has received much attentions. Given the fact that the cloud is heterogeneous, the allocation of resources has to become more sophisticated. As a first promising work to deal with that problem, Dominant Resource Fairness (DRF) has been proposed which takes into account dominant shares of users. Although DRF has a sort of desirable fairness properties, it has some limitations that have already been identified in the literature. Unfortunately, DRF and its recent developments are not intuitively fair with respect to various resource demands. In this paper, we propose a Multi-level Fair Dominant Resource Scheduling (MLF-DRS) algorithm as a new allocation model inspired by Max-Min fairness and proportionality. Unlike other works that they equalize dominant shares of different resource types which leads to starvation in the maximization of allocation for some users, our algorithm guarantees that each user receives the resources they desire for based on dominant shares. As can be deducted from the mathematical proofs, MLF-DRS provides a full utilization of resources and meets some of the desirable fair allocation properties and it is applicable to be used in a naïve extension form in the presence of multiple servers as well.**

Keywords-cloud computing; dominant resource; DRF; fairness; resource allocation

I. INTRODUCTION

Cloud computing [1] is a new technology which provides a wide range -and unlimited computing resources on-demand to users. Consequently, many organizations are adopting this technology in order to enhance their productivity and reduce costs in different sectors. This new technological trend abstracts the hardware and simplifies task management ranging from simple storage tasks to complex operations. Cloud computing consists of three architectural layers, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Application as a Service (AaaS) where each specific layer delivers particular aspects of cloud services. Virtualization [2] of resources which is done by Virtual Machines (VMs) is the fundamental aspect of cloud computing where all the resources are integrated in a resource pool without dealing with physical equipment, and it makes possible for everyone to access complex applications and huge amount of computing resources through the Internet. By considering the rapid evolution of the technology, cloud architectures have undergone major changes. There are many research directions such as resource management, provisioning and scheduling, VM management techniques and so on.

Resource allocation [3] is one of the most important aspects of cloud computing and how to allocate the resources among different users with various demands has become a challenging issue in recent decade. In perspective of a general definition of resource allocation, when CPU is allocated, it doesn't matter how much other resources are allocated. However, resource distribution in a datacenter is different and all the resources should be allocated in an

orchestrated way as a multi-resource allocation scheme to reach a fair and efficient allocation model [4].

Fairness in resource allocation has become a challenging problem and it has been investigated in different fields such as computer networks, economics, operating systems, and etc. Max-min fairness as a well-established algorithm has been proposed in order to provide a fair allocation among users with various demands. Generally, fairness is investigated for single resource in distributed and parallel computing like Hadoop fair scheduler [5] where multiple types of resources can be required by tasks which is a contradiction by considering the type of the Hadoop system. Since, cloud is a heterogeneous environment, the fair allocation of resources is very important especially in a case that users have intensive demands over different resources. To overcome fairness problem in the cloud, DRF [6] as a generalization of Max-min fairness has been proposed. Although DRF has some good features in terms of fairness, it has some limitations especially when there are multiple servers. Also, DRF is not able to consume all the resources which contributes to the wastage of them. Additionally, since DRF uses progressive filling, some users may not achieve their desirable allocation. There are many research works in extension of DRF, however they are not intuitively fair.

In this paper the resource allocation issues in cloud environments and several related algorithms were initially explored with emphasis on the fairness property. The most prominent algorithm is DRF. In this process, we found several drawbacks and limitations of DRF as regards to fairness and we proposed a new algorithm to address them. The rest of the paper is organized as follows. Section two investigates DRF algorithm in detail with its shortcomings and other related research works. Section three discusses regarding MLF-DRS with mathematical formulations and allocation algorithm. Section four presents evaluations using mathematical formulations and fairness properties met by MLF-DRS. Section five presents the conclusion and future research plans.

II. Fairness in Resource Allocation

One of the most important properties of resource allocation for the diverse and dynamic cloud environments is the fairness property. Fairness [7] can be considered as one of the challenging dimensions of Quality of Service (QoS) and an important concept in resource distribution so that subscribers should be happy with allocated resources. Since the fairness in an intuitive concept, it is difficult to define it as far as it is different from person to person. From the economic perspective, fair resource allocation is similar to a cake-cutting problem [8] in which the objective is to allocate specific parts of a cake among people. When it comes to the cloud and data center environment, the definition of fairness will be more challenging. Given that cloud environment is heterogeneous, users may submit their tasks consist of demands for different resources such as (CPU, Memory, Disk space and etc). Providing a fair resource allocation in such conditions requires defining new allocation rules and policies. Max-min fairness [9] is a well-known and popular algorithm which allocates available resources to users who are competing over resources. Based on max-min allocation policy, different resource allocation algorithms have been proposed to deal with the resource allocation problem. Initially, most of the research works have been done for single-resource environments such as works in [10][11] where there are only one type of resource such as CPU. However, considering that cloud environment is heterogeneous, multiple resources should be considered to provide a fair allocation [12]. Any fair resource allocation mechanism is subjected to have at least some important fairness properties [13] as followings:

- *Sharing Incentive:* Every user subjects to better off resource pool so that a user cannot get more tasks in a resource pool which is 1/n of total resources.
- *Pareto optimality:* A user will not be able to increase her/his allocation without decreasing other user's allocations. This is very promising feature which leads to maximum resource utilization in the system.
- *Envy-freeness:* A user should not be jealous about other user's allocation. In another word, a user's allocation is not tradable with another user. In terms of fairness definition, this allocation feature is counted as equity.
- *Strategy-proof:* A user cannot lie about his/her demands. Misreporting the demands may violate this important property.

The rest of the paper is organized as follows. Section II represents the related concepts in the area of fairness in resource allocation. Section III discusses about the recent conducted works in extension of DRF. Section IV describes the proposed model. Section V indicates the problem formulation of MLF-DRS and section VI provides the evaluations of MLF-DRS.

III. Related Works

DRF which is the generalization of max-min allocation algorithm is the first fair allocation method in a cloud environment with multiple resources with emphasis on resource intensive tasks. In DRF, the allocation is determined by equalizing subscribers' dominant shares so that the aim is to maximize the minimum dominant shares. As an example, if a system has (9 CPU, 18 GB) and user 1 requests (1 CPU, 4 GB) and User 2 requests (3 CPU and 1 GB), dominant resources will be GB for user 1 and CPU for user 2. Dominant shares for two users can be calculated as: 1/9 VS 4/18 (since 4/18>1/9 so GB is dominant), 3/9 VS 1/18 (since 3/9>1/18 so the CPU is dominant). Therefore, the method will try to equalize dominant shares of two users, subject to maximize their allocation. So, the final allocation according to DRF will be (3 CPU, 12 GB) for user 1 and (6 CPU and 4 GB) for user 2.

Different works have been proposed to tackle the existing problems in DRF in which they have considered different

extensions to use it in heterogeneous servers. Here we provide a brief description of the research works have been done until now based on DRF. DRFH is proposed in [14] as a generalization of DRF in order to apply in an environment with multiple servers by exploring an allocation to equalize each user's global dominant share, which is defined as the highest ratio of any resources the user has been allocated in the resource pool. DRFH improves the resource utilization in terms of multiple servers. In [15] heterogeneous DRF (HDRF) which is a generalization of DRF is proposed as a multi-resource scheduling algorithm to avoid job starvation in different data centers. The experiments show that HDRF presents better resource utilization and throughput compared to DRF. In [16] Long-Term DRF (H-MRF) is proposed which tackles the memory-less feature of DRF and other similar proposed models. The mentioned method is appropriate for pay-as-you-use services which are not considered in previous works. In [17] Per Server DRF (PS-DRF) is introduced to allocate resources more efficiently by considering server heterogeneity in presence of placement constraints. The most important feature of this algorithm is that it identifies dominant resource of each user based on Virtual Dominant Share (VDS). The proposed model considers important properties of fair resource allocation. In [18] Bottleneck-Based Fairness (BBF) is proposed which guarantees each user gets at least its entitlement resources and it also applies polynomial-time algorithm to calculate fair resource allocation. In [19] authors proposed a fair allocation model by taking into account the heterogeneous system and focusing on minimizing resource allocation costs.

IV. Proposed Model

According to the previous section, we highlighted the most important conducted research works regarding fair allocation so that much of their focus was on extension of DRF in different aspects such as cost optimization and commonly its application in heterogeneous servers. There are several problems associated with DRF and its application in cloud environments which can be summarized as follows: 1) In DRF allocation, full resource utilization is a problem so that the system is not able to consume all of its resources. Therefore, it can be a violation of Service Level Agreement as an obligation between users and providers. So, resource wastage will be occurred in DRF. Additionally, the most important deficiency of DRF and other proposed work is starvation so that when they try to maximize the allocation of the resources, some users may not be able to maximize their allocation due to the equalization of dominant shares. We will show in continue when a user tries to maximize own allocation, non-dominant resources of other users gets the considerable proportion of resources.

2) DRF is specifically designed for a single server. So, in terms of multiple servers DRF is not efficient so that a user may have dominant resources in different servers so that applying DRF in a naive extension may lead to an inefficient allocation.

By taking into account the above-mentioned problems, we propose a Multi-Level Fair Resource Allocation and Scheduling Algorithm (MLF-DRS) as a generalization of Max-Min fairness algorithm in multi-resource cloud environment considering dominant shares of each user's task. The allocation of resources in DRF and MLF-DRS has been shown in the previous section in presence of two users and in Table 2 with four users that are competing over two resources. According to the example in previous section, in the final allocation user 2 with dominant share on CPU has been allocated exactly what he/she asked for. In another word, the user is not able to maximize his/her allocation under DRF policy. As can be seen from table 1, in DRF, memory has not been fully utilized so that approximately 72% of the memory has been utilized and the remaining 24% has been wasted. It may be tangible in a system with high number of users and large-scaled system configuration. According to this, we think that the equalization of different resource types may cause starvation and resource wastage in each resource type. Hence, we are looking to the concept of equalization from another perspective in a way that equalization is done based on each specific resource type. After allocating the resources based on a user's initial demand vector, MLF-DRS tries to share the available resources equally among the dominant shares. In MLF-DRS, at the first level each user gets own initial demands and at the next stage since the objective is to maximize dominant shares of each user, dominant shares will be grouped in one set and no-dominants will be grouped in another one. At this level, the available resources will be divided and allocated equally among dominant shares subject to conditions mentioned in algorithm 1. For the final step, every set of dominant and non-dominants will get the equal amount of available resources. Therefore, the algorithm tries to share the resources in the same rate for intensive tasks until a resource becomes saturated. Additionally, MLF-DRS uses upper-bound available resources for dominants and lower-bound of available resources for non-dominants in a way that the allocation is in the same rate for both sets of resources. It is important to note that MLF-DRS applies dynamic allocation policy based on the server specifications such as total requested resources from the users and the capacity of resource pool.

A. Using MLF-DRS in Naïve Extension Form

In order to enable MLF-DRS to be used in multiple servers, according to [6] we assume that every user has a weight on demanded resources. Given that $w_{i,j}$ indicates the weight of users i on resource j. In that case, dominant resource d_i can be defined as follow:

$$d_i = \max\left\{\frac{s_{i,j}}{\omega_{i,j}}\right\}$$

Where $s_{i,j}$ represents the share of resource j of user i.

V. Problem Formulation

Given that there is a system with k resource types represented by $R = \{1,2,3, \dots, k\}$ with the capacity indicated

by C and there are n users compete over k types of resources with requested tasks indicated by x_i. Accordingly, dominant and non-dominant shares indicated by d_i^k *and* nd_i^k respectively, are calculated as follows:

$$d_i^k = argmax_k \ \frac{x_i}{C_k} \quad (1)$$

$$nd_i^k = argmin_k \ \frac{x_i}{C_k} \quad (2)$$

Based on (1) and (2), dominant and non-dominant shares vectors are specified by $D = \{d_1^k,\ d_2^k, \dots, d_n^k\}$ *and* $D = \{nd_1^k,\ nd_2^k, \dots, nd_n^k\}$ respectively. In that case, to calculate the allocation for each user which is indicated by a_i^k, the following optimization problem can be applied:

$$\max\left(d_i^k \ and \ nd_i^k\right) \quad (3)$$

$$subject\ to \ \sum_{i=1}^{n} x_i \ \leq C_k$$

$$x_i \leq \ f_s^k$$

$$\frac{a_1^k}{C_k} = \frac{a_2^k}{C_k} = \cdots = \frac{a_n^k}{C_k}$$

The optimization problem in (3) is subjected to different conditions of requested resources by the users which have been specified in algorithm 1. This is note that the allocation is not always equal for dominant shares and based on the capacity of the resource pool and user requests, it can be different. Hence, the maximization problem in (3) is generally applicable when requested dominant share of each user is less or equal to fair share of a specific resource type represented by f_s^k.

Note that in Table I, if the overall system capacity consists of <18 CPU , 36 RAM> are distributed equally between users, considering the example in table 1, user 2 gets 5 units in which smaller than the initial demand of that user. In that case based on the algorithm1, the allocation will be a bit different for dominant shares. Sometimes all the demands of users on a specific resource can be dominant. In that case, the algorithm will be relaxed to the traditional Max-Min fairness algorithm.

TABLE I. RESOURCE ALLOCATION IN DRF AND MLF-DRS

Users	**User 1**	**User 2**	**User 3**	**User 4**
Resources	CPU,GB	CPU, GB	CPU,GB	CPU,GB
Demands	3 , 1	5 , 3	1 , 5	2 , 7
DRF	6 , 2	5 , 3	3 , 12	4 , 14
MLF-DRS	6.5 , 2.3	7.2 , 4.3	1.4 , 14.7	2.9 , 14.7

Algorithm 1 MLF-DRS algorithm

$R \leftarrow (1, 2, \dots, k)$ ▷ k resource types vector
C^k ▷ capacity of resource type k
D^k ▷ all dominant shares of each specific resource k
ND^k ▷ all non-dominant shares of each specific resource k
$f_s \leftarrow C_k/n$ ▷ n indicates number of users and f_s is fair share
for each d_i^k and nd_i^k **do**
  $A(d_i^k) \leftarrow f_s$ ▷ initial allocation for dominant shares
  $A(nd_i^k) \leftarrow C_k/n$ ▷ initial allocation for non-dominant shares
$S \leftarrow \sum A(d_i^k)$ ▷ sum of all dominant shares of the resource type k
$S' \leftarrow \sum A(nd_i^k)$ ▷ sum of all non-dominant shares of the resource type k
$U \leftarrow S + S'$ ▷ current utilization of resources
$Av_r^k \leftarrow C_k - U$ ▷ Available resources in the resource pool for each specific resource type
$req(d_i)$ ▷ requested dominant resource
$req(nd_i)$ ▷ requested non-dominant resource
for each D and ND **do**
  if $req(d_i) \leq f_s$ and $req(nd_i) \leq f_s$ **then**
    $FA(d_i) = f_s + ((f_s * A_v)/U)$
    $FA(nd_i) = req(nd_i) + ((req(nd_i) * A_v)/U)$
  else if $req(d_i) \leq f_s$ and $req(nd_i)$ ¿ f_s **then**
    $FA(d_i) = f_s + ((f_s * A_v)/U)$
    $FA(nd_i) = f_s + (f_s * A_v)/U)$
  else if $req(d_i) > f_s$ and $req(nd_i) \leq f_s$ **then**
    $FA(d_i) = req(d_i) + ((req(d_i) * A_v)/U)$
    $FA(nd_i) = req(nd_i) + ((req(nd_i) * A_v)/U)$

VI. EVALUATIONS

A. *Evaluations Based on Mathematical Formulations*

Since, MLF-DRS is in implementation phase, we evaluate using numerical analysis to compare it with DRF. The demanded resources are specified in table 1. According to formulas (1) and (2), dominant shares for user 1 and 2 are CPU and for user 3 and 4 is GB. Based on the results of Table 1, Fig.1 and Fig2, it is obvious that in MLF-DRS users get their desirable amount of resources whereas in DRF some users are not able to maximize their allocation. So, by looking at the table 1, one realizes that user 2 is not able to maximize her allocation so that user 3 with a non-dominant share which is in CPU is able to maximize her allocation. This happens in equalization process and since DRF using progressive filling algorithm and allocates resources in a constant rate, non-dominants get more resources. In another word, non-dominants may get satisfied with allocation at least based on their initial demands. Hence, it is logical to allocate the most proportion of resources to dominant shares. Also, the results in table 1, Figures 1, 2, 3 and 4 represent that resource utilization is improved in 100% for CPU and memory.

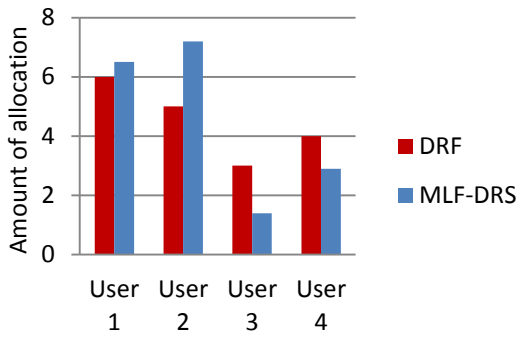

Figure 1. CPU allocation for MLF-DRS and DRF.

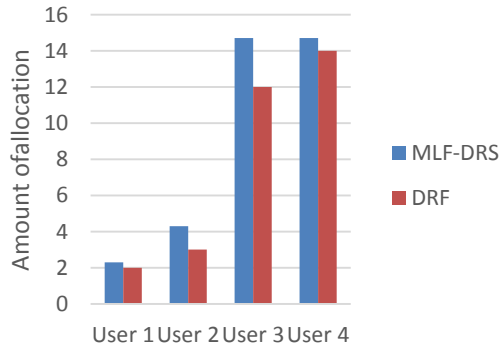

Figure 2. RAM allocation for MLF-DRS and DRF.

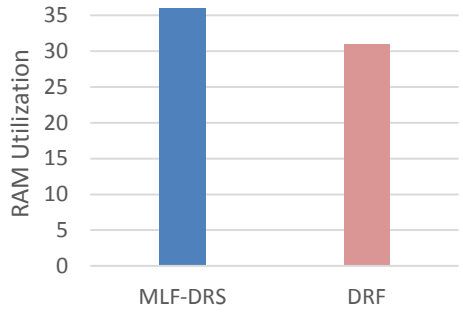

Figure 3. CPU utilization of MLF-DRS and DRF.

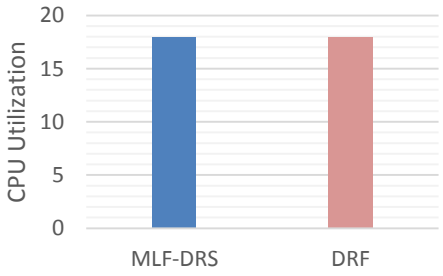

Figure 4. RAM utilization of MLF-DRS and DRF.

B. *Evaluations Based on Mathematical Formulations*

In this section we explore how MLF-DRS is able to meet some desirable fairness properties.

Theorem 1 MLF-DRS satisfies envy-freeness.

Proof: If we assume that *r,r** represent cpu and ram intensive tasks respectively, each task in each group *r,r** gets equal share of available resources. So, it will be automatically envy-free. As an example, if there are two tasks with dominant resources *d,d** where *d*>d*, and by taking into account that the algorithm increase dominant resources based on user demands, the allocation of *d** will be greater than *d* before saturation of the resource. In that case *d* will not be able to envy *d**.

Theorem 2 MLF-DRS meets the requirements for sharing incentive property.

Proof: Given that we have two groups of tasks denoted by *r,r**, and MLF-DRS increases the allocation of dominant shares of all users in each group based on the maximum share by proportionality, in that case, in final allocation we ensure that dominant resources get at least *Cr/n* of dominant share.

Theorem 3 MLF-DRS satisfies pareto-efficient.

Proof: Again, assuming that *r,r** denote ram and cpu intensive tasks respectively. Any resource of a task in *r,r** is able to increase its dominant resource without decreasing the allocation of other tasks. In another word, if there are two users *i,j* which are using a saturated resource *r*, then increasing the dominant share of user i would be decreasing the dominant share of user *j*. However, in every step of MLF-DRS algorithm by increasing the dominant resource of a user, another user's dominant share is increased as well. So, the algorithm is pareto-efficient.

Theorem 4 MLF-DRS meets strategy-proofeness in which users are not able to misreport their demands.

Proof: Assume that a user considers demand vectors *dr* and *d'r* in which *dr* and *d'r* denote true and misreported demands respectively. Given that MLF-DRS increases dominant shares based on available resources in each stage, if a user with *dr* tries to manipulate the server by *d'r* and considering that the capacity constraint is taken into account according to formula 3, in that case the constraint will be violated by misreporting the true demand by any user. So, it is not possible for a user to misreport her demand under MLF-DRS allocation policy.

VII. Conclusion and Future Works

In this paper we proposed MLF-DRS as a new fair allocation model in multi-resource cloud environments by considering DRF as a first fair resource allocation algorithm in multi-resource cloud environments. Although DRF has some good fairness features, it contains some drawbacks in terms of efficiency. In previous sections, we showed that in some situations, DRF is not able to allocate resources to some users in a desirable way specifically when there are more than two users in the system. Additionally, it is not possible to use DRF in naive extension way in presence of multiple servers and therefore leads to an inefficient resource

allocation in this case. Hence, our proposed MLF-DRS is applicable in such conditions with offering full utilization of resources. According to the results, MLF-DRS gives 100% utilization of resources and guarantees that each user gets his/her desirable resources. Note that we are currently in the implementation phase of this algorithm in a cloud environment for one of BT's industrial cloud application contexts. Also, the applicability of our proposed algorithm to self-adaptive and autonomic systems will be investigated as part of our research plans as autonomicity which is a part of most future industrial plans and fairness in autonomic systems hasn't been explored yet in the literature. Last but not least, we plan to extend our work to areas of user experience and socio-technical modelling and investigate the impact of fairness algorithms in societal contexts.

REFERENCES

[1] S. P. Ahuja and A. C. Rolli, "Survey of the State-of-the-Art of Cloud Computing," International Journal of Cloud Applications and Computing, vol. 1, no. 4, pp. 34–43, 2011.

[2] R. Buyya, C. Vecchiola, and S. T. Selvi, "Virtualization," Mastering Cloud Computing, pp. 71–109, 2013.

[3] P. Poullie, T. Bocek, and B. Stiller, "A Survey of the State-of-the-Art in Fair Multi-Resource Allocations for Data Centers," IEEE Transactions on Network and Service Management, vol. 15, no. 1, pp. 169–183, 2018.

[4] S. Kleban and S. Clearwater, "Fair share on high performance computing systems: what does fair really mean?," CCGrid 2003. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003. Proceedings., 2003.

[5] https://hadoop.apache.org/docs/r2.7.4/hadoop-yarn/hadoop-yarn-site/FairScheduler.html.

[6] H. Liu and B. He, "Reciprocal Resource Fairness: Towards Cooperative Multiple-Resource Fair Sharing in IaaS Clouds," SC14: International Conference for High Performance Computing, Networking, Storage and Analysis, 2014.

[7] D. Wischik, "Fairness, QoS, and buffer sizing," ACM SIGCOMM Computer Communication Review, vol. 36, no. 1, p. 93, Oct. 2006.

[8] J. Marenco and T. Tetzlaff, "Envy-free division of discrete cakes," Electronic Notes in Discrete Mathematics, vol. 37, pp. 231–236, 2011.

[9] A. Coluccia, A. D'Alconzo, and F. Ricciato, "On the optimality of max–min fairness in resource allocation," annals of telecommunications - annales des télécommunications, vol. 67, no. 1-2, pp. 15–26, 2011.

[10] J. Wang, Y. Yao, Y. Mao, B. Sheng, and N. Mi, "FRESH: Fair and Efficient Slot Configuration and Scheduling for Hadoop Clusters," 2014 IEEE 7th International Conference on Cloud Computing, 2014.

[11] K. Siriwong and R. Ammar, "QoS using delay-synchronized dynamic priority scheduling," Proceedings. Sixth IEEE Symposium on Computers and Communications.

[12] L. Wei, C. H. Foh, B. He, and J. Cai, "Towards Efficient Resource Allocation for Heterogeneous Workloads in IaaS Clouds," IEEE Transactions on Cloud Computing, vol. 6, no. 1, pp. 264–275, Jan. 2018.

[13] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar, and Y. Zhao, "An Efficient and Fair Multi-Resource Allocation Mechanism for Heterogeneous Servers," IEEE Transactions on Parallel and Distributed Systems, pp. 1–1, 2018.

[14] W. Wang, B. Li, and B. Liang, "Dominant resource fairness in cloud computing systems with heterogeneous servers," IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014.

[15] W. Wang, Y. Tan, Q. Wu, and Y. Zhang, "Multiple resources scheduling for diverse workloads in heterogeneous datacenter," 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), 2015.

[16] S. Tang, Z. Niu, B. He, B.-S. Lee, and C. Yu, "Long-Term Multi-Resource Fairness for Pay-as-you Use Computing Systems," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 5, pp. 1147–1160, Jan. 2018.

[17] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar, and Y. Zhao, "An Efficient and Fair Multi-Resource Allocation Mechanism for Heterogeneous Servers," IEEE Transactions on Parallel and Distributed Systems, pp. 1–1, 2018.

[18] D. Dolev, D. G. Feitelson, J. Y. Halpern, R. Kupferman, and N. Linial, "No justified complaints, A Bottleneck-Based Fair Resource Allocation" Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on - ITCS '12, 2012.

[19] K. Mukherjee, P. Dutta, G. Raravi, T. Rajasubramaniam, K. Dasgupta, and A. Singh, "Fair Resource Allocation for Heterogeneous Tasks," 2015 IEEE International Parallel and Distributed Processing Symposium, 2015.