MRFS: A Multi-Resource Fair Scheduling Algorithm in Heterogeneous Cloud Computing

Hamed Hamzeh[1], Sofia Meacham[1] Kashaf Khan[2] Keith Phalp[1] and Angelos Stefanidis[1]
[1]Faculty of Science and Technology, Bournemouth University, UK.
hamzehh@bournemouth.ac.uk, smeacham@bournemouth.ac.uk, kphalp@bournemouth.ac.uk, astefanidis@bournemouth.ac.uk
[2]British Telecom, Ipswich, UK, kashaf.khan@bt.com

***Abstract*—Task scheduling in cloud computing is considered as a significant issue that has attracted much attention over the last decade. In cloud environments, users expose considerable interest in submitting tasks on multiple Resource types. Subsequently, finding an optimal and most efficient server to host users' tasks seems a fundamental concern. Several attempts have suggested various algorithms, employing Swarm optimization and heuristics methods to solve the scheduling issues associated with cloud in a multi-resource perspective. However, these approaches have not considered the equalization of the number of dominant resources on each specific resource type. This substantial gap leads to unfair allocation, SLA degradation and resource contention. To deal with this problem, in this paper we propose a novel task scheduling mechanism called MRFS. MRFS employs Lagrangian multipliers to locate tasks in suitable servers with respect to the number of dominant resources and maximum resource availability. To evaluate MRFS, we conduct time-series experiments in the cloudsim driven by randomly generated workloads. The results show that MRFS maximizes per-user utility function by %15-20 in FFMRA compared to FFMRA in absence of MRFS. Furthermore, the mathematical proofs confirm that the sharing-incentive, and Pareto-efficiency properties are improved under MRFS.**

***Index Terms*—Allocation, Cloud, Dominant, fairness, Lagrangian, resource, server, scheduling, task, utility.**

I. Introduction

Cloud computing has emerged as a fundamental computing paradigm that facilitates IT operations by providing on-demand and easy to scale resources through the internet. With use of virtualisation technology, cloud delivers resources to end-users in the form of Virtual Machines(VMs). Cloud computing services are offered by data-centers that are widely distributed in different places. These services are mainly categorised in Infrastructure-as-a-Service(IaaS), Platform-as-a-Service(PaaS), and Software-as-a-Service(SaaS). IaaS is the most substantial layer, including a large quantity of virtual and physical machines [1].

Due to the huge expansion of demands in the cloud data centers, task management has become more challenging. In fact, optimal management and scheduling of tasks has a major importance in fulfilling Service Level Agreement(SLA). On the one hand, users should be satisfied with services delivered to them, and in another hand providers expect to utilize their resources efficiently to minimize potential costs [2].

Despite traditional computing systems where a single type of resource is considered, scheduling in the cloud is likely to be more complicated as resources are concentrated in a shared pool [3]. Accordingly, users may expose a high interest in submitting tasks, containing multiple resource types such as CPU, RAM, and storage disk. Apart from resource heterogeneity [4], data centers are composed of different servers with distinct configurations in terms of resources. This diversity in servers and computational resources represents a considerable complexity in cluster management [5]. Given that some jobs are intensive on a particular resource type, beyond the efficient utilization of resources, the concept of fairness is recognised as a major concern [6].

Scheduling in cloud computing with fairness was initially investigated by introducing Dominant Resource Fairness(DRF) [7]. DRF is a generalization of Max-Min fairness, aiming to schedule users' tasks by performing the progressive filling algorithm. Dominant resource is the most heavily demanded resource by a user in entire resource pool. Many extensions have proposed various ways to overcome shortcomings associated with DRF such as [8] [9] [10] [11] [12] [13] [14] [15] [16]. These approaches investigate alternative ways to schedule resources and tasks based on the main intuitive behind DRF. For example, PS-DSF algorithm [15] tries to schedule resources in most efficient servers in presence of placement constraints. The missing point regarding these developments is that they ignore how the number of dominant resources in a particular server may impact on fairness and utility maximization. In other words, balancing the number of dominant resources in each server leads to reducing the competition among those tasks dominated on a specific resource type. To address this issue, in this paper we propose a new policy-based fair scheduling algorithm called MRFS to balance submitted tasks with respect to dominant resources, aiming to maximize user utilities. We advocate that minimizing the competitions between users with dominant resources contributes in maximizing per-user utility function.

We apply Lagrangian multipliers on the main optimization problem to reach the maximum optimal point. We then conduct time-series experiments in the CloudSim driven by the randomly generated workloads. The results from the experiments confirm that the allocation under MRFS policy gains an optimal utility maximization while the fairness is also achieved. Furthermore, the resource allocation under MRFS scheduling policy alongside FFMRA algorithm [17]

shows that the utility is maximized by %15-20 compared to FFMRA without MRFS algorithm. FFMRA tries to equalize dominant and non-dominant resources to ensure that resources are distributed evenly among users. Actually, it introduces a new notion of the fairness to satisfy intuition-fairness property in the cloud.

This paper is organised as follows. Chapter II discusses the background and related work. In chapter III we describe the main motivation behind the work. Chapter IV presents the basic resource scheduling principles in the cloud. MRFS is introduced in Chapter V. In chapter VI the evaluation of MRFS is represented. Finally, in chapter VII a conclusion is drawn regarding MRFS.

II. BACKGROUND AND RELATED WORK

Resource allocation and scheduling in the cloud has attracted much attention over the last two decades. It is an NP-complete problem that aims to fulfil the Quality of Service(QoS) objectives such as resource utilization, execution time, minimizing potential costs, and energy consumption. Traditionally, the static scheduling algorithms known as heuristic methods such as First In First Out(FIFO), Round Robin(RR), Max-Min, Min-Max are still being used in cloud computing [18]. However, due to the fluctuation in workloads and dynamic behavior of the cloud, these approaches are likely straightforward as they are not capable of handling dynamic settings. Accordingly, a dozens of dynamic scheduling algorithm were proposed to deal with such variations in terms of workloads and system behavior. Ant Colony Optimization(ACO), Particle Swarm Optimization(PSO), and Weighted Least Connection(WLC) are the well-known examples of dynamic scheduling approaches [19].

Due to the resource heterogeneity and diversity in server configurations in the cloud, in addition to QoS parameters, fairness seems to be a significant contribution in cloud environments. The mentioned static and dynamic scheduling approach mainly to seek an optimal solution to solve different QoS specifications such as completion time, energy efficiency, and resource utilization. In distributed computing frameworks, the concept of fairness is well-established in Hadoop fair scheduler [20]. The fair scheduler in Hadoop places users jobs in different queues and allocate resources in the form of slots. Nonetheless, the fair scheduler in Hadoop considers only one resource type that leads a poor efficiency. Hadoop fair scheduler and similar approaches like Dryad [21], and Quincy [22] suffer by poor efficiency due to the resource fragmentation. To address this drawback, Dominant Resource Fairness (DRF) was proposed as the first promising fair resource allocation policy. DRF applies the progressive filling algorithm to schedule jobs depending on dominant resources. To perform scheduling, DRF usually picks a job with the minimum dominant resource. Since the progressive filling is handled in a single server, a sort of problem may occur such as poor efficiency in a case where DRF is implemented in Multiple servers. DRFH suggested an alternative approach to schedule and allocate resources in heterogeneous server profiles as it performs a best-fit heuristic method to schedule tasks in multiple servers. Subsequently, authors considered a global dominant share as the contribution of a user in entire resource pool. The intuition behind DRFH is to place all incoming tasks in a specific server. Under the DRFH scheduling policy, the sharing incentive is not fully satisfied. The major problem is that DRFH schedules all tasks of any user exclusively in a particular server. However, this may increase the quantity of dominated tasks on a specific resource type in a particular server. So, DRFH falls short in satisfying sharing-incentive property that may result in degrading per-user utility function.

ACO was recently proposed to solve the fair scheduling issue in heterogeneous cloud settings called ACO-TS [23]. The authors combine the identical features in Tabu search beside ACO to cover the gap between heuristic algorithms and optimal approximations. In addition to multiple servers, authors in [24] proposed a new mechanism called TSF that considers resource allocation and scheduling in the presence of placement constraints. Based on TSF policy a user is eligible to submit tasks solely over a certain subset of servers. However, under TSF mechanism it is not obvious how to specify a globally-wide dominant resource. PS-DSF [15] proposed a new approach to schedule tasks in servers by locally determining a virtual dominant resource for each server. Also, authors in [25] introduced a new approach to schedule tasks in a most appropriate server to minimize operational costs.

III. MOTIVATION

Existing approaches have not considered an equal assigning of tasks in the server with regards to the number of dominant resources. In fact, equalizing the number of dominant resources in each server, and maintaining the minimum aggregate resource demands considering multiple resource types lead to maximizing users' utilities.

Let's draw an example to clarify the problem. Assume there are five users A, B, C, D, and E with demand vectors $< 3CPU, 1GB >, < 5CPU, 3GB >, < 1CPU, 5GB >, < 2CPU, 7GB >$, and $< 4CPU, 9GB >$ respectively. The main purpose is to schedule users' tasks in server S. This server is eligible to host a maximum of four users. Fig. 2 presents a scheduling example of those tasks in a server from two perspectives. As can be seen one of $< 5CPU, 3GB >, < 4CPU, 9GB >$ is eligible to be scheduled in the server according to Fig. 2(a)(b). Otherwise, they could be scheduled in other servers if existed. Fig. 2(a) refers to perfect scheduling as the number of dominant resources on CPU, and RAM(GB) is equal. However, the scenario in Fig. 2(b) does not reach an optimal resource sharing as the number of tasks dominated on GB is more than CPU. Applying FFMRA, users C, and D receive a total of 26.6 ratios of the entire resource pool. Consequently, dividing it equally among them gives 13.4 units of GB. On the other hand, in a scenario represented in Fig. 2(b), by increasing the population of tasks dominated on GB, the utilization is decreased as 30.66 ratio of resources is allocated to users C, D, and E. Furthermore, the aggregate demands dominated on GB is 21 in scenario (b) which is

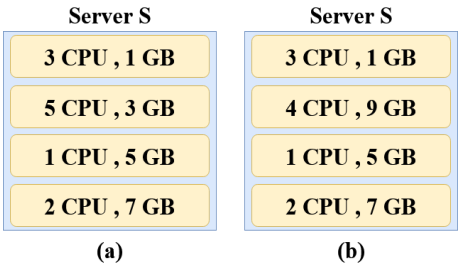

Fig. 1: An example of scheduling scenarios

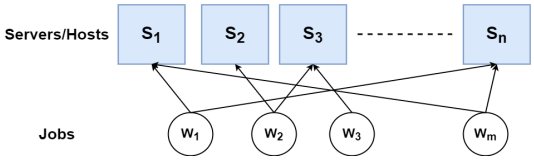

Fig. 2: Task scheduling in different servers

greater than the case in scenario (a) with 12 GB units. Therefore, imbalanced scheduling could adversely affect users' expectations in terms of utility maximization.

Unfortunately, the recently proposed approaches do not consider the competition based on the number of intensive tasks and also the minimum aggregate demands with dominant resources. This motivates us to design and develop a new policy-based scheduling algorithm to address these issues.

IV. Basic resource scheduling setup

Resource scheduling is a primary management process in the IaaS delivery model. VMs in the cloud data-center are recognized as scheduling units that are provisioned to heterogeneous resources. The scheduler regularly checks for the current status in the system to get available resources to check which server is suitable to host users' tasks.

After resource provisioning, incoming jobs are placed in different queues. Generally, the main purpose of the resource scheduler is to find the most efficient node/server to host a set of cloudlets/tasks. The main intuition behind resource scheduling in the cloud is to map a set of tasks $W =< W_1, W_2, ..., W_m >$ to available k types of resources $K =< 1, 2, ..., k >$. The resource scheduling problem is illustrated in Figure 1.

According to the basic settings in the cloud, assume that there is a system with heterogeneous servers and n number of users $U =< 1, 2, ..., n >$ with demand profiles $R =< r_1, r_2, ..., r_n >$. Also, assume that a user is eligible to submit any task including multiple types of resources. It is said that a resource is dominantly requested by a user if the following condition is satisfied:

$$d_{i,s}^k = max \frac{r_{i,s}^k}{C_k^{max}}, r > 0. \tag{1}$$

In which $d_{i,s}^k$ denotes a dominant resource that a user may request over any server s, and $r_{i,s}^k$ stands for the requested resource form a user on a resource type k over server s. Presumably, a submitted task by a user is determined locally as a task dominated by a particular resource type in a corresponding server. Respectively, the scheduler is responsible to map a task considering $d_{i,k}$ in a server to gain a maximum utility for each user.

$$U_i^{max}(\Pi(d_{i,s}^k)) \implies f : (f_{1,s}, f_{2,s}, ..., f_{n,s}) \tag{2}$$

The general utility maximization in (2) is subjected to find an optimal solution based on mapping function f, each binds cloudlets/tasks to the most efficient host. Typically, utility functions are the best indicators in evaluating the efficiency of any scheduling mechanism.

V. MRFS

In this section, a new policy-based fair task scheduling mechanism is proposed in the heterogeneous cloud called MRFS. MRFS considers dominant resources and performs a two-factor validation process to map a task to the most efficient server subjected to available resources. The first and the main factor is the frequency of dominant resources in each server, and the second one is the minimum aggregate dominant resources. If the last condition is not satisfied, the first factor is enough to schedule tasks. MRFS tries to maintain an equal distribution of tasks based on dominant resources on different resource types. As can be seen in Fig. 3, after submitting tasks, each task dominated on a specific resource type is placed in separate queues and then scheduled in the most appropriate server according to the specified rules. Technically, MRFS calculates dominant resources for each incoming task in all servers. If a server meets requirements and policies, then the task is scheduled in the corresponding server. It is important to note that the dominant resource for each task could be different from server to the server due to the diversity in configurations. So, MRFS treats for each task with respect to its dominant resource type in each server. On the other hand, if the configuration of all servers is identical, then MRFS considers a global dominant resource. In some conditions, if MRFS cannot find a suitable server to locate a task, the corresponding task is put into the non-dominant queue. However, as MRFS employs the FFMRA allocation policy, the corresponding dominant resource gets the fair share of the server's pool as it has the highest proportion of the resource quota belongs to non-dominant resources. This is to make sure that those tasks get at-least a fair share of resources to satisfy the sharing-incentive property. However, in large-scale scenarios with thousands of servers, the placing of dominant resources in non-dominant queues happens very rarely. This is important to note that MRFS could be applied in all fair allocation policies such as DRFH, and TSF.

Definition 1. *$N_{d,s}^k$ refers to the number of dominant resources in each specific resource type k in each server.*

$$N_{d,s}^k(t) = \Psi d_{i,s}^k \tag{3}$$

Based on (3) If t indicates the time in a series $< t_0, t_1, ..., t_i - 1 >$, the number of dominant resources on each particular resource type is updated in each time iteration t.

Definition 2. *The available resources after occupying a server with tasks is presented by* $A_{s,k}$.

$$A_{s,k}(t) = C_k^{max} - \sum_{i=1}^{k} r_{i,s}^k \tag{4}$$

Where C_k^{max} is the maximum capacity of a resource type in each server, and $r_{i,s}^k$ is the requested resource by user i in server s. Moreover, based on (4), and taking into account that in each iteration t, only one task is scheduled in each server, the available resources are updated relatively.

Definition 3. *As two sorted arrays, A and N represent two vectors each of which indicates available resources, and the number of dominant resources respectively in each server at time t. Correspondingly, the first, and the second items in the arrays are the minimum values of vectors. After each iteration, these vectors are sorted by increasing the number of tasks in each server.*

A server is indexed with a minimum aggregate dominant resource which is normalized to 1, so that $(\sum r_{i,S_1}^k \leq \sum r_{i,S_2}^k \leq ... \leq \sum r_{i,S_m}^k)$. If $S = \{r_{i,S_1}^k\}$, and $S^* = \{r_{i,S_2}^k, ..., r_{i,S_m}^k\}$, then $\zeta(t) = \sum S^*/n - s$ denotes the average utilization of all servers except S_1 that has the minimum summation of requested resources at time t.

Definition 4. *The parameter α is defined as an auxiliary penalty variable which maintains a correlation between the number of dominant resources with the current utilization at time t in server s. This is basically a penalty to show how the scheduler keeps the system in a desired state. To achieve α, it is assumed that $\Delta = \{\delta_1 = \frac{\sum d_{i,s}^1}{\Psi d_{i,s}^1}, ..., \delta_k = \frac{\sum d_{i,s}^k}{\Psi d_{i,s}^k}\}$ indicates aggregate dominant resources over the number of dominant resource types k at time t. Then to capture α, the values of Δ are normalised with the maximum capacity of each particular resource type which is calculated as follows:*

$$\alpha = |\frac{\delta_{k_1}}{C_{k_1}^{max}} - ... - \frac{\delta_{k_1}}{C_{k_r}^{max}}|, 0 < \alpha \leq 1 \tag{5}$$

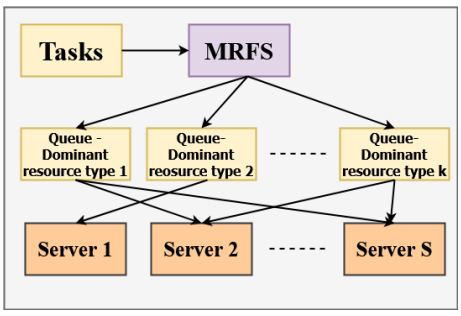

Fig. 3: MRFS scheduler structure

Generally speaking, the value of α represents the maximum penalty if the number of dominant resources on a specific resource type is more than the other dominant resource types. Even, if the number of dominant resources is the same, the value of α would be significantly high. In a perfect condition, where $\alpha = 0$, there is no penalty at all. Therefore, the following conditions are satisfied to maximize per-user utility in a set of servers.

$$\begin{cases} \Psi(d_{i,s}^1) = \Psi(d_{i,s}^2) = ... = \Psi(d_{i,s}^k) \\ min \sum_{i=1}^{k} d_{i,s}^k(t) \end{cases}$$

Where for all resource types in server S, $min \sum_{i=1}^{k} d_{i,s}^k(t)$ denotes the minimum aggregate requested tasks dominated on k types of resources. This is worth mentioning that the above conditions capture a perfect mapping of tasks to servers. However, there are cases that at-least one of these conditions is not fulfilled. For example, there is no server so that the number of dominant resources of all types of resources is not equal.

According to definitions 1,2,3 and 4, and given that $\Pi_{i,s}^k$ is the allocated resources to user i in server S, we are ready to formulate the optimization problem as follows:

$$\begin{aligned} \max \quad & U_i(\Pi_{i,s}^k) \\ \text{subject to} \quad & \sum_{i=1}^{n} r_{i,s}^k \leq C_k^{max} \\ & S \leq \zeta \end{aligned} \tag{6}$$

The maximization problem in (8) has two constraints. In order to achieve a perfect optimal mapping, the second constraint is required to be satisfied. To solve the maximization problem in (8) and find the best server to map a task onto it, we use *Lagrangian multipliers*. Lagrangian multiplier is an approach to find the local maximum of a function $f(x_1, x_2, ..., x_n)$ in an optimization problem, subject to a set of equality, or unequally constraints let's say $g(x_1, x_2, ..., x_n)$. The main intuition behind this method is to transform constraints to a set of *partial derivatives*. The derivatives of a function is applied in order to determine critical points of a function. This is very useful to find the local maximum point.

As an example, a simple utility maximization problem is considered as follows:

$$\begin{aligned} \max \quad & x = f(x) \\ \text{subject to} \quad & \mathrm{g}_i(x) \leq 0, \forall i = 1, ..., m \end{aligned} \tag{7}$$

So, it is possible to put the function f along with its constraints in a single maximization problem, using λ as a multiplier. Therefore, the problem in (7) can be written as follows:

$$x = max\mathcal{L}(x, \lambda) = max f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \tag{8}$$

Solving (8) gives a local maximum value for a maximization problem in (7).

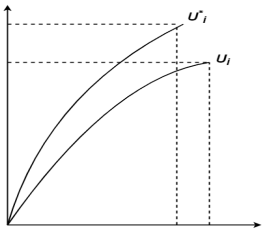

Fig. 4: The optimal utility over the original utility

Accordingly, the optimization problem in (6) could be formulated with constraint in an integrated optimization problem using Lagrangian multipliers. As there are more than one constraints, the optimization problem can be written as follows:

$$\mathcal{L}(U,\lambda,\mu) = (U_i(\Pi^k_{i,s}) + \lambda[\,C^{max}_k - \sum r^k_{i,s}] + \mu[\,\zeta - S]\,) - \alpha \quad (9)$$

Based on (9),the parameter U denotes the utility of a user i in any server s considering resource type k. Therefore, the optimization problem aims to maximize the allocation $\Pi^k_{i,s}$ in a most efficient host. The second constraint is accompanied by α to keep the correlation between $\Psi(d^k_{i,s})$, and $\sum d^k_{i,s}$.

To solve the Lagrangian function in (9), the First Order Necessary Condition(FOC) [26] is applied. As can be seen in Fig. 4 If U_i is the original user utility, U^* indicates the optimal utility by solving the Lagrangian function. Accordingly, it is necessary to substitute U with U^* in FOC.

$$\frac{\mathcal{L}(U^*,\lambda^*,\mu^*)}{\partial U} = \left(\frac{\partial f}{\partial U}(U^*) - \lambda^* \frac{\partial \sum r^k_{i,s}(U^*)}{\partial U} - \mu^* \frac{\partial S}{\partial U}(U^*)\right) - \alpha = 0 \quad (10)$$

Consequently, the equality in (8) could be divided in the following inequalities:

$$\begin{cases} C^{max}_k - \sum r^k_{i,s} \geq 0, \lambda^* \geq 0, \lambda^*[\,C^{max}_k - \sum r^k_{i,s}] \\ \zeta - S(U^*) \geq 0, \mu^* \geq 0, \mu^*[\,\zeta - S(U^*)] \end{cases}$$

The maximization problem in (6) could be relaxed to one constraint in a case, where the second constraint is not required due to the requirements. In this occasion, the problem can be written using a barrier function Φ as follows:

$$max U_i(\Pi^k_{i,s}) + \sum \Phi(d^k_{i,S_1}) \quad (11)$$

The value of Φ could be selected as a logarithmic barrier function as follows:

$$\Phi(U_i(\Pi^k_{i,s})) = -\sum log(d^k_{i,s}(U_i(\Pi^k_{i,s}))) \quad (12)$$

As MRFS performs FFMRA to calculate allocations, all fair allocation principals in FFMRA are exactly applied in MRFS. Among all fair allocation properties, we show that the sharing incentive and Pareto-efficiency properties are improved under MRFS scheduling mechanism.

Algorithm 1 MRFS scheduling algorithm

$K \leftarrow (k_1, k_2, ..., k_r)$ ▷ Resource vector
$S \leftarrow (s_1, s_2, ..., s_m)$ ▷ The vector contains all servers
$U \leftarrow (1, 2, ..., n)$ ▷ total users in the system
$D \leftarrow (r^k_{1,s}, ..., r^k_{i,s})$ ▷ demand vector
$Q^k_{d,s}$ ▷ Queue for dominant resource k in server s
$Q^k_{\tilde{d},s}$ ▷ Queue for non-dominant resource k in server s
$d^k_{i,s} = max \frac{r^k_{i,s}}{C^{max}_{s,k}}$ ▷ Dominant resource type k in server s
$t := 0$ ▷ time interval starts at 0
$\Psi(d^k_{i,s})(t)$ ▷ number of dominant resources in each server at time t
$A \leftarrow C^{max}_{s,k} - \sum r^k_{i,s}$ ▷ Available resource in each iteration at t
for each s in S **do**
  if $(\Psi(d^{k_1}_{i,s}) = \Psi(d^{k_2}_{i,s}) = ... = \Psi(d^{k_r}_{i,s}))$ & $s_m = min(\sum d^k_{i,s})$ **then**
    $Q^k_{d,s} \leftarrow r^k_{i,s}$ ▷ Placing requested task onto dominants queue
  else if $(\Psi(d^{k_1}_{i,s}) = \Psi(d^{k_2}_{i,s}) = ... = \Psi(d^{k_r}_{i,s}))$ **then**
    $Q^k_{d,s} \leftarrow r^k_{i,s}$ ▷ Placing requested task onto dominants queue
  else if $(\Psi(d^{k_1}_{i,s}) = ... = \Psi(d^{k_r}_{i,s})) + 1$ **then**
    $Q^k_{\tilde{d},s} \leftarrow r^k_{i,s}$
  else if $(\Psi(d^{k_1}_{i,s}) > \Psi(d^{k_r}_{i,s})) + 1$ **then**
    $Q^k_{\tilde{d},s} \leftarrow r^k_{i,s}$
  $UpdateA$
  $t := t + 1$

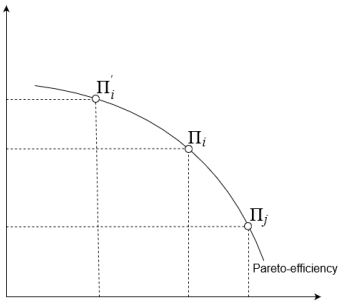

Fig. 5: The curve represents Pareto-efficiency where $\Pi^{'}_i$ represents the improved Pareto-efficiency

Theorem 1. *There is Pareto-efficiency improvement under MRFS mechanism*

Proof. If there is a positive change in user i's allocation $\Pi_{i,k}$ denoted by $\Pi^{'}_{i,k}$ so that $\Pi^{'}_{i,k} > \Pi_{i,k}$. Accordingly, user i cannot worse-off user j'allocation under a perfect complementary utility function, e.g. $U_i(\Pi^{'}_i) \geq U_j(\Pi_j)$.

As it is shown in Fig. 5, all allocations are in line with Pareto-efficiency. So, $\Pi^{'}_i$ is increased without affecting $\Pi^{'}_j$ allocation. The further improvement in terms of Pareto-efficiency is not feasible if an allocation reaches to a desired state. Therefore, we need to show that under MRFS,

decreasing the population of tasks dominated on a specific resource type reduces the competition among users that leads to improve Pareto-efficiency.

To show whether the Pareto-efficiency is improved under MRFS, we assume that a user i is given by a weight ω_i. Accordingly, for each allocation Π_i there is a social-welfare indicator let's say S_w based on weighted aggregate utility as follows:

$$s_\omega(\Pi) = \sum_{i=1}^{n} \omega_i \cdot U_i(\Pi_i) \tag{13}$$

So, the allocation Π_ω maximizes the social-welfare over other allocations, e.g:

$$\Pi_\omega \in argmax s_\omega(\Pi) \tag{14}$$

Considering (13) and (14), under MRFS scheduling policy, minimizing the number of dominant resources leads to maximizing the utility of each user in a specific server S as follows:

$$min\Psi(d^k_{i,s}) \implies max U_i(\Pi) \tag{15}$$

Therefore, the social welfare is improved according to (15) as:

$$S_\omega(\Pi^*) > S_\omega(\Pi)$$

□

Theorem 2. *MRFS improves sharing-incentive property*

Proof. Primarily, to proof the sharing incentive property, it is essential to show that

$$U_i(\Pi^k_{i,s}) \geq U_j(\Pi^k_{j,s}) \tag{16}$$

So, the utility of user i is greater than or equal to user j's utility. This condition is assumed to be satisfied under FFMRA mechanism, if $U_i(\Pi) = \Pi^k_{i,s} / \sum_j (\Pi_{j,s})$.

We consider Z_o as a solution under FFMRA in absence of MRFS, and also Z^*_o in presence of MRFS. Typically, the number of dominant resource of a specific resource type under solution Z_o is more than that in Z^*_o; $(\Psi d^k_{i,s})(Z_o) \geq \Psi d^k_{i,s}(Z^*_o)$. Therefore, considering λ, and μ, the utility of user i is increased as follows:

$$(U_i(\Pi^k_{i,s}) + \lambda[C^{max}_k - \sum r^k_{i,s}] + \mu[\zeta - d^k_{i,s_1}]U_i(\Pi^k_{i,s})) - \alpha \tag{17}$$

Consequently, any allocation Π under solution Z^*_o captures a strong sharing-incentive property. So, the inequality in (11) is applied to the optimal solution Z^*_o.

□

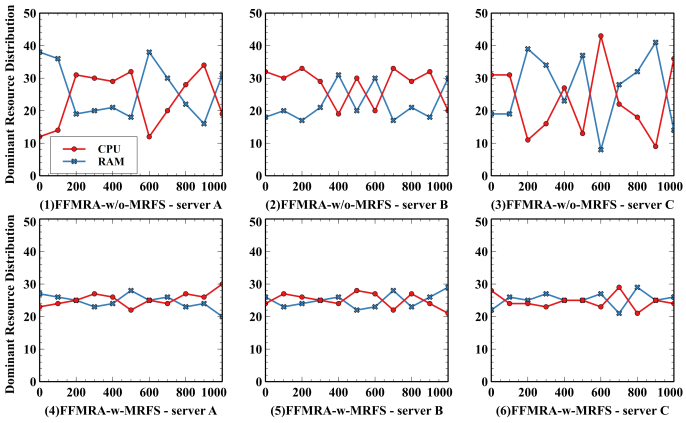

Fig. 6: The distribution of dominant resources in FFMRA and MRFS

VI. Evaluation

In order to evaluate the performance of MRFS, we use a system with Intel core-i5, 8250U, 1.6GHz CPU, and 8GB RAM. To conduct the experiments, we use the CloudSim simulation framework driven by the randomly-generated workloads in 1000 time-series experiments. we take into account different parameters that are: the distribution of dominant resources in servers; the proportion of resource pool among dominant and non-dominant resources; the proportion of dominant resources for each specific resource type; and finally the total allocated resource to users. Except for the last one which we mainly use a relatively large-scale evaluation. For other parameters, and in order for the simplicity we simulate the system with three users over three servers, each has the same number of VM requests with diverse demand profiles. Furthermore, each server is configured with $< 3000CPU, 6000RAM >$ since the total capacity of the data center is $< 9000CPU, 18000RAM >$.

Fig.6 illustrates the number of dominant resources in each server in 1000 iterations over three hosts, comparing FFMRA with and without MRFS. As can be seen in Fig.6 (1)(2)(3) the population of dominant resources in CPU is significantly higher than dominated tasks on RAM. However, under MRFS scheduling as it is illustrated in Fig. 6(4)(5)(6), the difference between the number of dominant resource types is considerably low. This is due to that according to the queuing mechanism in MRFS, the scheduler tries to equalize the number of dominant resources in each server in the entire data center. For example, based on Fig. 6(4)(5) after iteration 900, the difference of dominant resources in servers A, and B is increasing. However, MRFS tries to minimize it in server C as it is shown in Fig. 6(6).

Equalizing the number of dominant resources in each specific resource type may contribute to maximizing the proportion of the entire resource pool in the data-center to all dominant resources. Based on the FFMRA allocation policy(see[17]), Fig.7(a)(b) clearly states that the proportion of resource pool for dominant resources under the MRFS policy

is considerably better than the VM Allocation policy in which FFMRA operates accordingly. Assume that CPU and RAM are considered as two types of resources, the total proportion for dominant CPU and RAM in MRFS scheduling policy is strictly better than FFMRA in absense of MRFS in the CloudSim. However, Fig.7(b) presents that the proportion for non-dominant tasks on CPU, and RAM in FFMRA is higher than scheduling under MRFS.

Assume that FFMRA shares a certain proportion of data-center resources to each specific group of users dominated on a particular resource type. Since, under MRFS the number of dominant resources on CPU and RAM is well-balanced, based on Fig. 8, the corresponding proportional value is considerably high compared to the same scenario in FFMRA. In Fig.8(a) though the proportion for CPU is higher than the value for RAM under MRFS. This is due to that the sum of requested demands for CPU is greater than RAM.

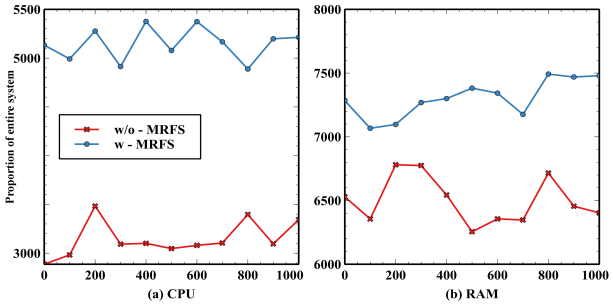

Fig. 7: The proportion of entire resource pool to dominant resources under FFMRA in presense of MRFS and without MRFS

Fig. 9 represents the allocated tasks to users under MRFS. Typically, as the competition among tasks dominated on each resource type is well-balanced in MRFS, the total allocated resources to users is slightly better than the policy used by FFMRA. However, based on Fig. 9(1) FFMRA allocates more resources to user A due to that user A has more non-dominant task submissions on CPU in contrast to users B, and C. On the other hand, users A and C regularly submit non-dominant tasks on RAM. Therefore, the allocated tasks to those users in some iterations are lower than the allocation under MRFS. The penalty variable α is shown in Table II where the values are randomly selected from different servers in time-series experiments. Accordingly, the greatest value in the table refers to a higher penalty. As is has been already discussed, the scheduler applies α for each user's allocation based on (5). The calculations of α, δ-CPU, and δ-RAM are randomly drawn using β values for dominant, and non-dominant CPU and RAM in TABLE I. The values in both tables are strongly depended to each other.

To analyze the sharing incentive property under MRFS policy, we conduct an experiment using 100 servers each of which is eligible to host 40 demands in each time t. In order for simplicity, it is assume that the configuration of all servers is identical. Therefore, each task is recognized by a global

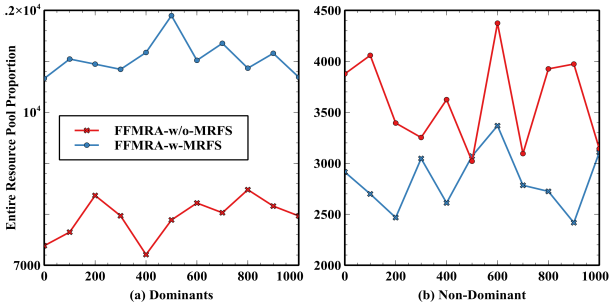

Fig. 8: The entire resource pool proportion for dominant, and non-dominant resources in FFMRA in presence and absence of MRFS

dominant resource on a particular resource type. Furthermore, we have selected eight demands in random from one of the servers that is also randomly selected. according to Fig. 10(a) all demands schedule more CPU-dominated tasks than the *fairshare*. On the other hand, and based Fig. 10(b) demands 5 cannot schedule more tasks above the *fairshare* in some iterations. This is due to that the scheduler is not able to find a suitable host to locate that task. Consequently, demand 5 is placed in non-dominant queues so that it schedules less tasks than the *fairshare*. However, this is trivial and it could not be happened in scenarios with large number of servers. Nonetheless, in a general setting MRFS captures $\%95 - 97$ sharing incentive for demands dominated on a specific resource type. For example for a scenario with 10000 servers, the probability of locating a task with dominant resource in non-dominant queue is something like $1/10000$ which is a trivial value. Therefore, in this occasion the sharing-incentive property is almost fully achievable.

TABLE I: The values of Beta in which d and nd refer to dominant, and non-dominant resources respectively

Beta d-cpu	Beta d-RAM	Beta nd-CPU	Beta nd-RAM
1.066834069	0.728409041	0.108561842	0.250502592
1.035760848	0.719698324	0.10183153	0.244452501
1.012268025	0.711362962	0.103893504	0.246280228
0.999276224	0.700556703	0.107685474	0.262689845
0.998897746	0.820490563	0.141714168	0.312789566
1.066772968	0.728390924	0.108579392	0.250518144

VII. Conclusion and Future Work

In this paper, we propose a new multi-resource fair scheduling algorithm called MRFS where users impose a great interest

TABLE II: The calculation of Alpha based on Delta variable over CPU and RAM

Delta CPU	**Delta RAM**	**Alpha**
2457.352941	10203.4375	0.264436581
2600	10289.0625	0.254453125
2705.588235	9435.625	0.201222426
2720.882353	9909.375	0.223380515
2173.333333	8273.888889	0.196361111
2457.255882	10203.25	0.264436912

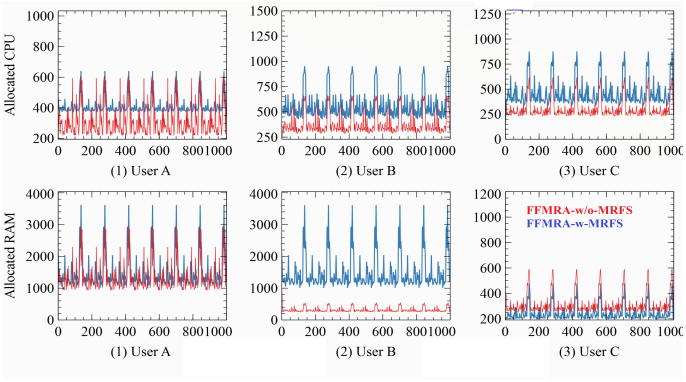

Fig. 9: The number of tasks allocated as dominant resources for three users

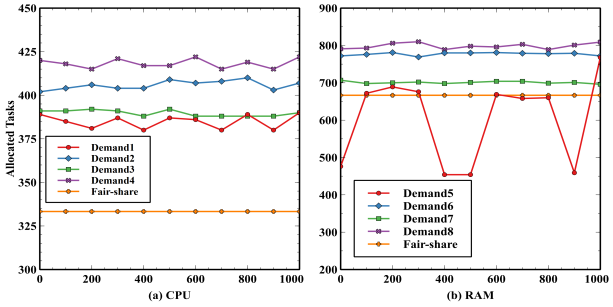

Fig. 10: The allocation of resources for eight demands under FFMRA with and without MRFS with respect to the fairshare

in submitting tasks on multiple types of resources over the servers with distinct configurations. The proposed algorithm tries to equalize the number of dominant resources to reduce the competition among tasks dominated by a specific resource type. The main purpose of MRFS is to maximize the per-user utility function of with dominant resource submissions. We employ Lagrangian multipliers to schedule tasks in a server with the most availability in terms of resources and the number of dominant resources. Then we conduct time-series experiments in the cloudsim driven by randomly-generated workloads to evaluate the performance of MRFS. The experiments show that FFMRA allocation under MRFS policy leads to a maximum allocation to users compared to FFMRA in absence of MRFS. We also show that the sharing-incentive and Pareto-efficiency are improved under MRFS. For future work we intend to extend MRFS in cutting-edge cloud frameworks such as Kubernetes as a novel cloud-native solution.

REFERENCES

[1] Zhongni Zheng, Rui Wang, Hai Zhong and Xuejie Zhang, "An approach for cloud resource scheduling based on Parallel Genetic Algorithm," 2011 3rd International Conference on Computer Research and Development, Shanghai, 2011, pp. 444-447.

[2] Z. Zhan, X. F. Liu, Y. j. Gong, J. Zhang, H. S. Chung, and Y. Li, "Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches, " 2015 ACM Comput. Surv. 47, 4, Article 63 (July 2015), 33 pages.

[3] D. Dong, H. Xiong, G.G. Castañé, P. Stack, J.P. Morrison, "Heterogeneous Resource Management and Orchestration in Cloud Environments" 2018 Cloud Computing and Service Science. Communications in Computer and Information Science, vol 864. Springer, Cham

[4] C. Reiss, A. Tumanov, G. Ganger, R. Katz, M. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis", Proc. ACM 3rd Symp. Cloud Comput., pp. 7, 2012.

[5] R. Boutaba, L. Cheng, Q. Zhang, "On cloud computational models and the heterogeneity challenge", J. Internet Services Appl., vol. 3, no. 1, pp. 77-86, May 2012.

[6] C. Joe-Wong, S. Sen, T. Lan, M. Chiang, "Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework", Proc. IEEE Conf. Comput. Commun., pp. 1206-1214, 2012.

[7] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in Proc. USENIX NSDI, 2011.

[8] A. Ghodsi, M. Zaharia, S. Shenker, I. Stoica, "Choosy: Max-min fair sharing for datacenter jobs with constraints", Proc. 8th ACM Eur. Conf. Comput. Syst. (EuroSys), pp. 365-378, Apr. 2013.

[9] D. Klusáček, H. Rudová, "Multi-resource aware fairsharing for heterogeneous systems", Proc. 18th Int. Workshop Job Scheduling Strategies Parallel Process. Revised Selected Papers (JSSPP), pp. 53-69, May 2015.

[10] W. Li, X. Liu, X. Zhang, X. Zhang, "Dynamic fair allocation of multiple resources with bounded number of tasks in cloud computing systems", Multiagent Grid Syst., vol. 11, no. 4, pp. 245-257, 2015.

[11] D. C. Parkes, A. D. Procaccia, N. Shah, "Beyond dominant resource fairness: Extensions limitations and indivisibilities", Proc. 13th ACM Conf. Electron. Commerce (EC), pp. 808-825, Jun. 2012.

[12] W. Wang, B. Li, B. Liang, "Dominant resource fairness in cloud computing systems with heterogeneous servers", Proc. 33rd Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM), pp. 583-591, Apr. 2014.

[13] J. Zhang et al., "ATFQ: A fair and efficient packet scheduling method in multi-resource environments", IEEE Trans. Netw. Service Manag., vol. 12, no. 4, pp. 605-617, Dec. 2015.

[14] Y. Tahir, S. Yang, A. Koliousis, J. McCann, "Udrf: Multi-resource fairness for complex jobs with placement constraints", GLOBECOM, pp. 1-7, Dec 2015.

[15] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar and Y. Zhao, "Per-Server Dominant-Share Fairness (PS-DSF): A multi-resource fair allocation mechanism for heterogeneous servers," 2017 IEEE International Conference on Communications (ICC), Paris, 2017, pp. 1-7.

[16] "Towards multi-resource fair allocation with placement constraints", Proc. ACM SIGMETRICS, 2016.

[17] H. Hamzeh, S. Meacham, K. Khan, K. Phalp and A. Stefanidis, "FFMRA: A Fully Fair Multi-Resource Allocation Algorithm in Cloud Environments", 2019 The 3th IEEE International Workshop on Software Engineering for Smart Systems (SESS).

[18] S. A. Ali and M. Alam, "A relative study of task scheduling algorithms in cloud computing environment," 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), 2016.

[19] S. Wang and G. Ji, "Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications, " 2015, Hindawi.

[20] $https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html$

[21] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly.Dryad: distributed data-parallel programs fromsequential building blocks. InEuroSys 07, 2007.

[22] M. Isard, V. Prabhakaran, J. Currey, U. Wieder,K. Talwar, and A. Goldberg. Quincy: Fair scheduling fordistributed computing clusters. InSOSP '09, 2009.

[23] X. Liu, X. Zhang, Q. Cui and W. Li, "Implementation of Ant Colony Optimization Combined with Tabu Search for Multi-resource Fair Allocation in Heterogeneous Cloud Computing," 2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity).

[24] W. Wang, B. Li, B. Liang and J. Li, "Multi-resource Fair Sharing for Datacenter Jobs with Placement Constraints," SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, 2016, pp. 1003-1014.

[25] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar and Y. Zhao, "A Cost-Efficient and Fair Multi-Resource Allocation Mechanism for Self-Organizing Servers," 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-7.

[26] Notes on Calculus and Optimization.