Per-Server Dominant-Share Fairness (PS-DSF): A Multi-Resource Fair Allocation Mechanism for Heterogeneous Servers

Jalal Khamse-Ashari*, Ioannis Lambadaris*, George Kesidis†, Bhuvan Urgaonkar† and Yiqiang Zhao‡

*Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada
†School of EECS, Pennsylvania State University, State College, PA, USA
‡School of Math and Statistics, Carleton University, Ottawa, Canada

Emails: *{jalalkhamseashari,ioannis}@sce.carleton.ca, †{gik2, buu1}@psu.edu ‡zhao@math.carleton.ca

Abstract—Users of cloud computing platforms pose different types of demands for multiple resources on servers (physical or virtual machines). Besides differences in their resource capacities, servers may be additionally heterogeneous in their ability to service users - certain users’ tasks may only be serviced by a subset of the servers. We identify important shortcomings in existing multi-resource fair allocation mechanisms - Dominant Resource Fairness (DRF) and its follow up work - when used in such environments. We develop a new fair allocation mechanism called Per-Server Dominant-Share Fairness (PS-DSF) which we show offers all desirable sharing properties that DRF is able to offer in the case of a single “resource pool” (i.e., if the resources of all servers were pooled together into one hypothetical server). We evaluate the performance of PS-DSF through simulations. Our evaluation shows the enhanced efficiency of PS-DSF compared to the existing allocation mechanisms. We argue how our proposed allocation mechanism is applicable in cloud computing networks and especially large scale data-centers.

I. INTRODUCTION

Cloud computing has become increasingly popular as it provides a cost-effective alternative to proprietary high performance computing systems. As the workloads to datacenters housing cloud computing platforms are intensively growing, developing an efficient and fair allocation mechanism which guarantees quality-of-service for different workloads has become increasingly important. Resource allocation and especially fair sharing in such shared computing system is particularly challenging because of the following reasons: a) heterogeneity of servers, b) placement constraints, c) dealing with multiple types of resources, and d) diversity of workloads and demands.

Real world data-centers are comprised of heterogeneous machines/servers with different configurations, where some machines might be incompatible for some processing purposes/tasks. Furthermore, each user may have specific requirements which further restrict the set of servers that the tasks of the user may run on. For example, a user may require a machine with a public IP address, particular kernel version, special hardware such as GPUs, or large amounts of memory, and might be unable to run on machines which lack such requirements. For instance, it has been observed that over 50% of tasks at Google clusters have strict constraints about the machines they can run on [1], [2].

Besides placement constraints, users present diversity over the amount of resources they need for executing one task. For instance, the tasks of some users might be CPU intensive while for others memory or I/O bandwidth might be a bottleneck.

Dominant Resource Fairness is the first allocation mechanism which describes a notion of fairness when allocating multiple types of resources [3]. With DRF users receive a fair share of their dominant resource. Of all the resources requested by the user (for every unit of work called a task), its dominant resource is the one with the highest demand when expressed as a fraction of the overall resource capacity spread across all available servers. There are several other works investigating DRF allocation in case that different resources are distributed over heterogeneous servers but there are no placement constraints [4], [5], [6], [7].

There are some recent works investigating max-min fair allocation/scheduling for one type of resource while respecting placement constraints [2], [8], [9], [10], [11], [12]. These schedulers could be useful in a multi-resource setting only when one of the resources serves as the bottleneck for all users, otherwise they might result in poor resource utilization [3], [2]. There are limited works in the literature investigating multi-resource fair allocation while respecting placement constraints [13], [14], [15]. In this case, it is unclear how to globally identify the dominant resource as well as the dominant share for different users, as each user may have access only to a subset of servers. [14], [15] present an elementary extension of DRF which identify the share of each user by ignoring the placement constraints and applying the same ideas as the un-constrained setting. We show that this approach does not achieve fairness even in the specific case that one of the resources serves as a bottleneck (Further discussions could be found in Section II-B).

Our Contributions: We propose a new allocation mechanism called Per-Server Dominant Share Fairness. We show that PS-DSF achieves all the desirable properties offered by DRF for a single resource pool: sharing incentive, strategy proofness, envy freeness, Pareto optimality, bottleneck fairness and single resource fairness (A detailed description of these properties can be found in Section II-A). In fact, PS-DSF reduces to DRF when one considers a single server system.

The intuition behind PS-DSF is to compare and weigh the allocated resources to each user from the perspective of each server. PS-DSF identifies a dominant resource and a virtual dominant share (VDS) for each user with respect to each server (as opposed to a single system-wide dominant share in DRF).
The VDS for user $n$ with respect to (w.r.t.) server $i$ describes the fraction of the dominant resource which should be allocated to user $n$ from server $i$ as if all user $n$’s tasks were allocated resources solely from server $i$. Each server may then use this localized metric to decide whether to increase or decrease the allocated tasks to each user without the need to identify a global dominant share. Besides its enhanced performance, PS-DSF is the first (to our knowledge) principled allocation mechanism which could be intrinsically implemented in a distributed manner.

The rest of this paper is organized as follows: In Section II, after describing the model, we give the necessary background and discuss insufficiency of the existing multi-resource allocation mechanisms, especially in case of heterogenous servers with placement constraints. After presenting our proposed allocation mechanism in Section III, we investigate different sharing properties that it satisfies and present a distributed algorithm to realize it. We present some numerical experiments in Section IV, and finally we draw conclusions in Section V.

II. BACKGROUND AND MODEL

Consider a set $K = |K|$ heterogeneous servers (resource pools) each containing $M$ types of resources. We denote by $c_{i,r}$ the capacity of resource $r$ on server $i$, where $c_{i,r} \geq 0$. Let $N$ denote the set of active users, where $N = |N|$. Let $d_{n} = [d_{n,r}]$ denote the per task demand vector for user $n \in N$, that is the amount of each resource required for executing one task for user $n$. Let $\phi_{n} > 0$ denote the weight associated with user $n$. The weights reflect the priority of users with respect to each other.

Due to heterogeneity of users and servers, each user may be restricted to get service only from a subset of servers. For example, each user may have some special hardware/software requirements (e.g., public IP address, a particular kernel version, GPU, etc.) which restrict the set of servers that the tasks of the user may run on. Besides such explicit placement constraints, users may not run their tasks on servers which lack some required resources.

For instance, consider the example in Figure 1, where three types of resources, CPU, memory, and network bandwidth are available over two servers in the amounts of $c_1 = [9, 12, 100]$ and $c_2 = [12, 2, 12]$ Mb/s, where no communication bandwidth is available over the second server. Consider three users with the weights $\phi_1 = 1$, $\phi_2 = 2$, whose demand vectors are $d_1 = [1, 2, 10]$, $d_2 = [1, 2, 1]$ and $d_3 = [1, 2, 0]$. Accordingly, users 1 and 2 are restricted to get service only from the first server, while user 3 may get service from both servers. In summary, let $\delta_{n,i} = 1$ if the tasks of user $n$ can run on server $i$, and otherwise $\delta_{n,i} = 0$.

A. Dominant Resource Fairness

The problem of multi-resource fair allocation was originally studied in [3] under the assumption that all resources are aggregated at one resource-pool. Specifically, let $c_r$ denote the total capacity of resource $r$. Let $a_{n} = [a_{n,r}]$ denote the amounts of different resources allocated to user $n$ under some allocation mechanism $A$. The utilization of user $n$ of its allocated resources, $U_{n}(a_{n})$, is defined as the number of tasks, $x_{n}$, which could be executed using $a_{n}$, that is:

\[
U_{n}(a_{n}) := x_{n} = \min_{r} \frac{a_{n,r}}{c_{r}},
\]

where, $x_{n}$ is a non-negative real number. [3] argues that the following important properties must be satisfied by a multi-resource allocation mechanism:

- **Sharing Incentive:** Consider a generic uniform allocation where every user $n$ is allocated $\phi_{n} / \sum_{m} \phi_{m}$ portion of each resource. An allocation is said to provide sharing incentive, when each user is able to run more tasks compared to the uniform allocation.
- **Envy freeness:** A user should not prefer the allocation of another user when adjusted according to their weights, i.e., $U_{n}(a_{n}) > U_{n}(\frac{\phi_{n}}{\sum_{m} \phi_{m}} a_{m})$ for all $m$.
- **Pareto Optimality:** It should not be possible to increase $x_{n}$ for any user $n$, without decreasing $x_{m}$ for some user $m$.
- **Strategy Proofness:** Users should not be able to increase their utilization by lying about their resource demands.

Sharing incentive provides performance isolation, as it guarantees a minimum utilization for each user irrespective of the demands of the other users. Envy freeness embodies the notion of fairness. Pareto optimality results in maximizing system utilization. Finally, strategy proofness prevents users from gaming the allocation mechanism. The reader is referred to [3] or [16] for further details.

DRF is the first multi-resource allocation mechanism satisfying all the above properties. Specifically, for every user $n$, the Dominant Resource (DR) is defined as [3]:

\[
\rho(n) := \arg \max_{r} d_{n,r} / c_{r},
\]

that is, the resource whose greatest portion is required for execution of one task for user $n$. The fraction of the DR that is allocated to user $n$ is defined as dominant share:

\[
s_{n} := \frac{a_{n,\rho(n)}}{c_{\rho(n)}}.
\]

Without loss of generality, we may restrict ourselves to non-wasteful allocations, i.e., $a_{n} = x_{n}d_{n}$, $\forall n$. In this case, an allocation \{\(x_{n}\)\} is feasible when:

\[
\sum_{n} x_{n}d_{n,r} \leq c_{r}, \forall r.
\]
Definition 1. It is said that \( \{x_n\} \) satisfies DRF, if it is feasible and the normalized dominant share for each user, \( s_n/\phi_n \) cannot be increased while maintaining feasibility without decreasing \( s_m \) for some user \( u \) with \( s_m/\phi_m \leq s_n/\phi_n \) [3].

DRF is a restatement of max-min fairness in terms of dominant shares. What makes it appealing are desirable sharing properties which are satisfied under this allocation mechanism. Besides the above-mentioned essential properties, DRF also satisfies the following simple but essential properties [3].

- Single Resource Fairness: When there is only one resource type, the allocation satisfies max-min fairness.
- Bottleneck Fairness: If there is one resource which is dominantly requested by each user, then the allocation satisfies max-min fairness for that resource.

B. Challenges with Heterogeneous Resource-Pools and Placement Constraints

The notion of DRF has been extended to the case of heterogeneous servers, when all types of resources are available within each server and there are no placement constraints [7]. In this case, DRF for user \( u \) is readily identified as the resource whose greatest portion is required for execution of one task if all resources were integrated at resource pool. That is, DRF for user \( u \) could be identified according to (2), where \( c_r := \sum_i c_{i,r} \) is the total capacity of resource \( r \). Furthermore, the global dominant share for user \( n \) is given by:

\[
s_n = x_n \max_r \frac{d_{n,r}}{c_r},
\]

(5)

where \( x_n \) here is the total number of tasks which are allocated to user \( n \) from different servers, that is \( x_n := \sum_i x_{n,i} \). In [7] it is proposed to find \( \{x_{n,i}\} \) such that max-min fairness is achieved in terms of global dominant shares. This mechanism, which is referred to as DRFH, has been shown to achieve Pareto optimality, strategy proofness, envy freeness and bottleneck fairness. However, it fails to provide sharing incentive [7].

When there are placement constraints, it is unclear how to define a single system-wide DR for a user similar to that in [7]. A natural first thought may be to identify the DR over the set of eligible servers for each user. However, in this case users may have an incentive to misreport the set of eligible servers [14]. A strategy-proof approach is to identify the DR for each user as if there were no placement constraints and all resources were integrated at one resource pool. We argue that this approach, which we refer to as C-DRFH, does not result in a fair allocation as it does not satisfy bottleneck fairness.

To appreciate this shortcoming of C-DRFH, consider the example in Figure 1, where the second resource (RAM) is dominantly requested by each user from its eligible servers. If we allocate the available RAM proportionate to the weights, 6GB is allocated to the first two users and 12 GB is allocated to the third user. Accordingly, each user is allocated \( x_1 = x_{1,1} = 3 \), \( x_2 = x_{2,1} = 3 \) \( x_3 = x_{3,1} + x_{3,2} = 6 \) tasks (this allocation follows from our proposed allocation mechanism - see Section III). However, C-DRFH would instead identify bandwidth as the dominant resources for the first user and identifies RAM as the dominant resource for the second and third users. Hence, if we allocate global dominant shares in a weighted fair manner, each user is allocated \( x_1 = x_{1,1} = 2.609 \), \( x_2 = x_{2,1} = 3.130 \), and \( x_3 = x_{3,1} + x_{3,2} = 6 + 0.261 = 6.261 \) tasks respectively, which obviously violates fairness on the bottleneck resource.

Yet another extension of DRF that also considers heterogeneous servers, all containing all types of resources without any placement constraints, is CDRF [4]. Specifically, let \( \gamma_n := \sum_i \gamma_{n,i} \) be defined as the number of tasks which are allocated to user \( n \) when monopolizing the whole cluster (i.e., if \( n \) were the only user running on the cluster). An allocation is said to satisfy CDRF1, when \( x_n/\gamma_n \) satisfies max-min fairness. In case of one server, \( x_n/\gamma_n \) gives dominant share for each user \( n \). As a result, CDRF reduces to DRF in case of one server. In case of multiple heterogeneous servers with no placement constraints, CDRF is shown to satisfy Pareto optimality, strategy proofness, envy freeness and sharing incentive properties [4].

In [14] CDRF has been extended to address the placement constraints. Specifically, let \( \gamma_n := \sum_i \gamma_{n,i} \) be (re)defined as the number of tasks which are allocated to user \( n \) from different servers when monopolizing all servers as if there were no placement constraints [14]. An allocation is said to satisfy Task Share Fairness (TSF), when \( x_n/\gamma_n \) satisfies max-min fairness. TSF is shown to satisfy Pareto optimality, strategy proofness, envy freeness and sharing incentive properties in case of heterogeneous servers with placement constraints [14]. However, we argue that this mechanism is not essentially fair as it does not satisfy bottleneck fairness.

For instance, consider again the example in Figure 1. The number of tasks that each user may run in the whole cluster is \( \gamma_1 = \gamma_2 = 6 \), and \( \gamma_3 = 12 \) tasks, respectively. Hence, each user is allocated \( x_1 = x_{1,1} = 2 \), \( x_2 = x_{2,1} = 2 \) and \( x_3 = x_{3,1} + x_{3,2} = 6 + 2 \) = 8 tasks according to TSF mechanism, which is completely different and far from the fair allocation.

Table I summarizes different sharing properties which could be satisfied under different allocation mechanisms. Shortcomings of the existing allocation mechanisms in case of heterogeneous servers with placement constraints motivates us to develop a new allocation mechanism.

### Table I: Properties of different allocation mechanisms in case of heterogeneous servers with placement constraints: sharing incentive (SI), envy freeness (EF), strategy proofness (SP), Pareto optimality (PO), and bottleneck fairness (BF).

| Property | C-DRFH | TSF | PS-DSF |
|----------|--------|-----|--------|
| SI       | ✓      | ✓   | ✓      |
| EF       | ✓      | ✓   | ✓      |
| SP       | ✓      | ✓   | *      |
| PO       | ✓      | ✓   | *      |
| BF       | ✓      | ✓   | ✓      |

Yet another extension of DRF that also considers heterogeneous servers, all containing all types of resources without any placement constraints, is CDRF [4]. Specifically, let \( \gamma_n := \sum_i \gamma_{n,i} \) be defined as the number of tasks which are allocated to user \( n \) when monopolizing the whole cluster (i.e., if \( n \) were the only user running on the cluster). An allocation is said to satisfy CDRF1, when \( x_n/\gamma_n \) satisfies max-min fairness. In case of one server, \( x_n/\gamma_n \) gives dominant share for each user \( n \). As a result, CDRF reduces to DRF in case of one server. In case of multiple heterogeneous servers with no placement constraints, CDRF is shown to satisfy Pareto optimality, strategy proofness, envy freeness and sharing incentive properties [4].

In [14] CDRF has been extended to address the placement constraints. Specifically, let \( \gamma_n := \sum_i \gamma_{n,i} \) be (re)defined as the number of tasks which are allocated to user \( n \) from different servers when monopolizing all servers as if there were no placement constraints [14]. An allocation is said to satisfy Task Share Fairness (TSF), when \( x_n/\gamma_n \) satisfies max-min fairness. TSF is shown to satisfy Pareto optimality, strategy proofness, envy freeness and sharing incentive properties in case of heterogeneous servers with placement constraints [14]. However, we argue that this mechanism is not essentially fair as it does not satisfy bottleneck fairness.

For instance, consider again the example in Figure 1. The number of tasks that each user may run in the whole cluster is \( \gamma_1 = \gamma_2 = 6 \), and \( \gamma_3 = 12 \) tasks, respectively. Hence, each user is allocated \( x_1 = x_{1,1} = 2 \), \( x_2 = x_{2,1} = 2 \) and \( x_3 = x_{3,1} + x_{3,2} = 6 + 2 = 8 \) tasks according to TSF mechanism, which is completely different and far from the fair allocation. Table I summarizes different sharing properties which could be satisfied under different allocation mechanisms. Shortcomings of the existing allocation mechanisms in case of heterogeneous servers with placement constraints motivates us to develop a new allocation mechanism.

### III. Per-Server Dominant Share Fairness

In this section we describe PS-DSF, an extension of DRF that is applicable for heterogeneous resource-pools in the presence of placement constraints. As discussed in the previous section, in the case of heterogeneous servers and in the presence of placement constraints, it is unclear how to globally identify one DR and the corresponding dominant share for each user. The intuition behind PS-DSF is to define a virtual
dominant share for every user w.r.t. each server. Towards this, we first define the DR for every user \( n \) w.r.t. each server \( i \) as:

\[
\rho(n, i) := \arg \max_r \frac{d_{n,r}}{c_{i,r}}.
\]

(6)

Let \( \gamma_{n,i} \) denote the number of tasks which could be executed by user \( n \) when monopolizing server \( i \):

\[
\gamma_{n,i} := \delta_{n,i} \min_r \frac{c_{i,r}}{d_{n,r}} = \delta_{n,i} \frac{c_{i,\rho(n,i)}}{d_{n,\rho(n,i)}}.
\]

(7)

We say that server \( i \) is eligible to serve user \( n \) when \( \gamma_{n,i} > 0 \) or equivalently \( \delta_{n,i} = 1 \). Without loss of generality we restrict ourselves to non-wasteful allocations, that is \( \delta_{n,i} = x_{n,i}d_n \), where \( d_n = [a_{n,i,r}] \) is the vector of allocated resources to user \( n \) from server \( i \) and \( x_{n,i} \in \mathbb{R}^+ \) is the number of allocated tasks from the same server.

**Definition 2.** The Virtual Dominant Share (VDS) for user \( n \) w.r.t. server \( i \), \( s_{n,i} \), is defined as:

\[
s_{n,i} = \frac{x_n}{\gamma_{n,i}},
\]

(8)

where \( x_n = \sum_j x_{n,j} \) is the total number of tasks that are allocated to user \( n \) (whether or not these tasks are actually allocated using server \( i \)).

Intuitively, \( s_{n,i} \) gives the fraction\(^2\) of the dominant resource for user \( n \) w.r.t. server \( i \) which should be allocated to it as if \( x_n \) tasks were allocated to it entirely from server \( i \). When the available resources over each server are arbitrarily divisible, we have the following condition on \( \{x_{n,i}\} \) to be feasible.

**Definition 3.** An allocation, \( \{x_{n,i}\} \), is said to satisfy Resource Division Multiplexing (RDM) constraint, when:

\[
\sum_n x_{n,i}d_{n,r} \leq c_{i,r}, \forall i,r.
\]

(9)

For a data-center comprising of a plurality of servers, it is sometimes of more practical interest to assume that servers may not be divided to finer partitions [2]. Accordingly, the hypervisor may only time-share servers among different users. In this case, we have the following condition on \( \{x_{n,i}\} \) to be feasible.

**Definition 4.** An allocation, \( \{x_{n,i}\} \), is said to satisfy Time Division Multiplexing (TDM) constraint, when\(^3\):

\[
\sum_n x_{n,i}/\gamma_{n,i} \leq 1, \forall i.
\]

(10)

We investigate our proposed allocation mechanism under both of these feasibility conditions.

**Definition 5.** An allocation \( \{x_{n,i}\} \) satisfies Per-Server Dominant-Share Fairness, if it is feasible and the allocated tasks to each user \( x_n \) cannot be increased while maintaining feasibility without decreasing \( x_{m,i} \) for some user \( m \) and server \( i \) with \( s_{m,i}/\phi_m \leq s_{n,i}/\phi_n \).

---

\(^2\)The reader may note that \( s_{n,i} \) could be possibly greater than 1, as some tasks could be allocated to user \( n \) from other servers.

\(^3\)Considering resources such as CPU, BW, \ldots, which are attributed a processing speed per time-unit, \( x_{n,i}/\gamma_{n,i} \) represent the percentage of time-unit that server \( i \) is allocated to user \( n \).

---

**A. An Example**

Consider again the heterogeneous servers from our earlier example this time serving four equally weighted users whose demand vectors are \( d_1 = [1.5, 1, 10] \), \( d_2 = [1, 2, 10] \), \( d_3 = [0.5, 1, 0] \), \( d_4 = [1, 0.5, 0] \). We show this in Figure 2. Note the placement constraints for users 1 and 2 whose tasks may only run on the first server. We show the PS-DSF allocation (based on RDM) in Figure 3. The allocated tasks to each user are \( x_1 = x_{1,1} = 3.6, x_2 = x_{2,1} = 3.6, x_3 = x_{3,2} = 8, x_4 = x_{4,2} = 8 \), respectively, where no tasks are allocated to users 3 and 4 from the first server. Specifically, the VDS for user 3 (and user 4 respectively) w.r.t. the first server is \( s_{3,1} = 8/12 \) (\( s_{4,1} = 12/12 = 1 \)), while the VDS of users 1 and 2 w.r.t. this server is \( s_{1,1} = s_{2,1} = 0.6 \). The VDS of users 3 and 4 w.r.t. the second server is \( s_{3,1} = s_{4,1} = 8/12 \). The reader may verify that for each server \( i \) the allocated tasks to each user may not be increased without decreasing the allocated tasks of some user with less VDS.

**Fig. 2:** A heterogeneous multi-resource system with four users and two servers.

**Fig. 3:** PS-DSF allocation for the example in Figure 2.

---

**B. The Properties of the PS-DSF Allocation Mechanism**

Before examining different sharing properties satisfied under PS-DSF allocation mechanism, we describe a necessary and sufficient condition to achieve PS-DSF.

**Definition 6.** Given a feasible allocation \( \{x_{n,i}\} \) based on RDM, we say that \( r \) is a bottleneck resource for user \( n \) w.r.t. an eligible server \( i \) if \( d_{n,r} > 0 \), \( \sum_m x_{m,i}d_{m,r} = c_{i,r} \) (i.e. \( r \) is saturated), and

\[
\frac{s_{n,i}}{\phi_n} \geq \frac{s_{m,i}}{\phi_m}, \forall m \text{ such that } x_{m,i}d_{m,r} > 0.
\]

(11)
Theorem 1. A feasible allocation \( \{x_{n,i}\} \) based on RDM satisfies PS-DSF if and only if there exists a bottleneck resource for every user w.r.t. every eligible server.

Theorem 2. A feasible allocation \( \{x_{n,i}\} \) based on TDM satisfies PS-DSF if and only if \( (10) \) holds with equality, and
\[
\frac{s_{n,i}}{\phi_n} \geq \frac{s_{m,i}}{\phi_m}, \forall n \text{ and } \forall m \text{ such that } x_{m,i} > 0. \tag{12}
\]

For the proof refer to our technical report [17]. These conditions will be useful in determining a PS-DSF allocation (see Section III-C). In the following we examine different sharing properties that are satisfied under PS-DSF. In case of a heterogeneous system with placement constraints, we will need to extend the notion of Sharing Incentive, Strategy Proofness and Bottleneck Fairness. Other properties, Envoy Freeness, Pareto Optimality and Single Resource Fairness, will follow the same definitions as described in Section II-A.

The generalization of sharing incentive property is straightforward. We consider a uniform allocation which allocates \( \phi_n/\sum s_{n,i} \phi_n \) portion of the resources on each server (whether this server is eligible or not) to each user \( n \). An allocation is said to satisfy sharing incentive, when each user is able to run more tasks compared to such uniform allocation.

For the strategy proofness property, we may note that we assume each user to declare its demand vector and also the set of eligible servers. We say that an allocation satisfies strategy proofness when users may not increase their utilization by lying about their resource demands or the set of eligible servers.

Finally, a resource is considered as a bottleneck in the whole system when it is dominantly requested by each user from every eligible server. If there is a bottleneck resource, then the allocation should satisfy max-min fairness w.r.t. such resource.

Theorem 3. PS-DSF allocation mechanism (whether based on RDM or TDM) satisfies single resource fairness, bottleneck fairness, envy freeness, and sharing incentive properties. It also satisfies Pareto optimality and strategy proofness in case of TDM.

For the proof refer to our technical report [17]. Unfortunately PS-DSF does not satisfy Pareto optimality in case of RDM. This is the reason why strategy proofness is not generally satisfied in case of RDM. The following lemma describes the behaviour of PS-DSF allocation mechanism from this respect.

Lemma 1. Assume that all users demand all type of resources, that is \( d_{n,r} > 0, \forall n, r \). Under the PS-DSF allocation mechanism with RDM, each user cannot decrease the utilization of other users by lying about its resource demands or the set of eligible servers, without decreasing its own utilization.

For the proof refer to our technical report [17].

C. PS-DSF Allocation Algorithm

In this subsection, we present an algorithm which realizes the PS-DSF allocation in the case of RDM\(^4\). According to Theorem 1, an allocation satisfies PS-DSF when every user has a bottleneck resource w.r.t. every eligible server. Let \( N_i \) denote the set of users for which \( \gamma_{n,i} > 0 \). The following corollary describes a condition to check whether a saturated resource serves as a bottleneck for user \( n \) w.r.t. server \( i \).

Corollary 1. If \( r \) is saturated at server \( i \) and
\[
n \in \arg \min_{m \in N_i} \left\{ \frac{s_{n,i}}{\phi_m} \mid d_{m,r} > 0 \right\}
\]
Then, \( r \) is a bottleneck resource for user \( n \) at server \( i \) when:
\[
\frac{s_{m,i}}{\phi_m} > \frac{s_{n,i}}{\phi_n}, \quad d_{m,r} > 0 \Rightarrow x_{m,i} = 0. \tag{14}
\]

To find a PS-DSF allocation, we may apply an iterative algorithm beginning with an initial allocation. Assume that servers are indexed from 1 to \( K \). Starting from the first server, the proposed algorithm sequentially updates the allocation for different servers, so that at the end a bottleneck resource is identified for every user w.r.t. every eligible server. In the following we describe the procedure for updating the allocation at each server.

Specifically, for each server \( i \) let \( N_i \) initially denote the set of users for which \( \gamma_{n,i} > 0 \). Given a feasible allocation, \( \{x_{n,i}\} \), find \( S_i^* \) as the minimum VDS at server \( i \):
\[
S_i^* := \min_{m \in N_i} \left\{ \frac{s_{n,i}}{\phi_m} \right\}. \tag{15}
\]
The set of users achieving the minimum in \( (15) \) is denoted by \( N_i^* \). Let \( R_i^+ \) denote the set of saturated resources at server \( i \) for which \( d_{n,r} > 0 \) for some user \( n \in N_i^* \). These resources are the potential bottleneck resources for users \( n \in N_i^* \). If the condition in Corollary 1 is satisfied for some resource \( r^* \in R_i^+ \), then this resource serves as the bottleneck for users \( n \in N_i \) with \( d_{n,r^*} > 0 \). In this case, we restrict our attention to the users for which no bottleneck resource is identified w.r.t. server \( i \). Specifically, \( N_i \) is updated to:
\[
N_i = N_i - \{ n \mid d_{n,r^*} > 0 \}. \tag{16}
\]

When the condition in Corollary 1 is not satisfied for any resource \( r \in R_i^+ \), the algorithm updates the allocation for server \( i \). Specifically, for every resource \( r \in R_i^+ \), a user \( n_r \) is chosen such that:
\[
n_r \in \arg \max_{n \in N_i} \left\{ \frac{s_{n,i}}{\phi_n} \mid x_{n,i} d_{n,r} > 0 \right\}. \tag{17}
\]

If we release the whole allocated resources to these users from server \( i \), the maximum potential increase in \( S_i^* \) is given by \( z^* \) (see the Update-Allocation subroutine in Algorithm II). To make sure that \( S_i^* \) is monotonically increasing, \( \beta \in (0, 1] \) is chosen such that \( S_i^* + \beta z^* \) remains less than or equal to the updated VDS w.r.t. server \( i \) for all users \( n_r, r \in R_i^+ \).

For each server \( i \), the above procedure is repeated until \( N_i \) becomes empty. At the end of this procedure, a bottleneck resource is identified for every user eligible to be served by server \( i \). However, the subsequent updates for the next servers, may violate this condition for server \( i \) and the previous servers. Hence, we repeat the whole process for all servers, until no more update is possible for any of the servers\(^5\). This process is described in Algorithm I.

\(^4\)A simplified version of this algorithm can be used in the case of TDM.

\(^5\)Convergence properties of this algorithm will be studied in our future work.
Algorithm I: PS-DSF Allocation Algorithm

Initialization
Initially allocate available resources by applying DRF individually to each server.

The main subroutine
while (1)

Last-round-flag := 1
for (i = 1; i ≤ K; i + +)
    \(N_i := \{n \in N \mid |\gamma_n,i| > 0\}\).
while (\(N_i \neq 0\))
    \(S_i^* \) according to (15).
    Identify \(N_i^*\) as the set of users achieving the minimum in (15).
    Identify \(R_i^*\) as the set of saturated resources at server \(i\) for which \(d_{n,r} > 0\) for some \(n \in N_i^*\).
    If \(S_i^* = \max_{n \in N_i} \{\sum_{r \in R_i^*} x_{n,r}d_{n,r} > 0\}\), for \(r^* \in R_i^*\)
    Update \(N_i = N_i - \{n \mid d_{n,r^*} > 0\}\)
    else
        Last-round-flag = 0
        Call Update-Allocation(x, i).
        If (Last-round-flag = 1)
            break.

Update-Allocation(x, i) subroutine
Identify \(f_i = [x_i]\) as the amount of unallocated resources under \(x\).
for (\(r \in R_i^*\))
    Choose \(n_r \in \arg \max_{n \in N_i} \{\sum_{r \in R_i^*} x_{n,r}d_{n,r} > 0\}\).
    Update \(f_i = f_i + x_{n_r}d_{n_r}\).
    Find \(D_r^* := \sum_{n \in N_i^*} \phi_n \gamma_n,r d_{n_r}\).
    Find \(z^* := \min_{n \in N_i^*} \phi_n \gamma_n,r d_{n_r}\).
    Choose \(\beta \in (0, 1]\) such that: \(S_i^* + \beta z^* \leq \sum_{n \in N_i^*} \phi_n \gamma_n,r d_{n_r}, \forall r \in R_i^*\).
    Update \(x_{n_r} = x_{n_r} + \beta \phi_n \gamma_n,r z^*, \forall n \in N_i^*\).
    Set \(x_{n_r,i} = (1 - \beta)x_{n_r,i}, \forall n \in R_i^*\).

D. Distributed Implementation

One of the advantages of the PS-DSF allocation mechanism is that it locally identifies the dominant resource for each user w.r.t. each server, without any knowledge of the available resources on the other servers, as opposed to existing allocation mechanisms which need to globally identify dominant resource and/or dominant share for each user. This is of great importance from a practical point of view, as we may develop a distributed algorithm to find the PS-DSF allocation.

Specifically, consider the inner while-loop in the main subroutine of Algorithm I which we refer to as "server procedure". According to this procedure the allocated tasks to different users from each server \(i\) are updated only based on the knowledge of the available resources on server \(i\) and the total allocated tasks to each user. Accordingly, we may come up with a distributed version of Algorithm I where each server individually (and even asynchronously) executes the server procedure every \(T\) seconds. When \(T\) is chosen sufficiently smaller than period of changes in a cluster (like changes in the set of active users and/or servers), such distributed algorithm may dynamically achieve the PS-DSF allocation. We implement this algorithm in our experiments in Section IV.

IV. Numerical Results

In this section we evaluate performance of the PS-DSF allocation mechanism through some numerical experiments. In our simulations, we consider a cluster with four different classes of servers (120 servers in total), where the configuration of servers are drawn from the distribution of Google cluster servers [18]. It is assumed that the available resources over each server can be partitioned in any arbitrary way. We consider four users where the last two users may run their tasks only by the last two classes of servers (see Figure 4).

The number of tasks that each user may run when monopolizing each class of servers are given in Table III. Assume that all users are active. The PS-DSF (based on RDM) and the TSF allocations in this case are given in Table IV. Under both allocations the servers of the first two classes (the second two classes respectively) are allocated to the first (the last) two users. According to the PS-DSF allocation, the servers of the third class (the fourth class respectively) are entirely allocated to the third user (the fourth user), which results in maximizing the minimum VDS w.r.t. the servers of the first class.

Intuitively, PS-DSF tries to allocate each server to the most efficient users. Therefore, we expect that PS-DSF results in greater utilization for different resources of a server compared to other allocation mechanisms such as TSF and C-DRFH. To observe this, we have executed these algorithms over the interval \((0, 300)\) sec for the cluster in Figure 4. For the PS-DSF, we start with an initial allocation and update the allocation every second according to the servers’ procedure (see our discussions in Section III-D on distributed implementation). For TSF and C-DRFH mechanisms we precisely find these allocations every second.

It is assumed that all users except User 4 are continuously active during the simulation interval. User 4 is inactive during interval \((100, 250)\) sec, and is active elsewhere. The utilization that is achieved under any of these allocation mechanisms for the CPU at the third and the fourth classes of servers are shown respectively in Figure 5 (The CPU on the first two classes of servers and also the memory on all servers are fully utilized under any of the allocation mechanisms). It can be observed that the PS-DSF allocation mechanism results in
Fig. 5: The utilization that is achieved for the CPU at the third and the fourth classes of servers under PS-DSF, TSF and C-DRFH allocation mechanisms.

TABLE III: The total number of tasks that each user may run when monopolizing each class of servers.

| User | Class A | Class B | Class C | Class D |
|------|---------|---------|---------|---------|
| User 1 | 80      | 340     | 82.5    | 55      |
| User 2 | 40      | 170     | 41.25   | 41.25   |
| User 3 | 0       | 0       | 82.5    | 27.5    |
| User 4 | 0       | 0       | 0       | 27.5    |

TABLE IV: The total number of tasks allocated to each user from each class of servers under PS-DSF and TSF allocations.

| PS-DSF | Class A | Class B | Class C | Class D |
|--------|---------|---------|---------|---------|
| User 1 | 40      | 170     | 0       | 0       |
| User 2 | 20      | 85      | 0       | 0       |
| User 3 | 0       | 0       | 82.5    | 0       |
| User 4 | 0       | 0       | 0       | 27.5    |

| TSF    | Class A | Class B | Class C | Class D |
|--------|---------|---------|---------|---------|
| User 1 | 35      | 170     | 0       | 0       |
| User 2 | 22.5    | 85      | 0       | 0       |
| User 3 | 0       | 0       | 58.33   | 0       |
| User 4 | 0       | 0       | 8.05    | 27.5    |

greater utilization compared to the two other mechanisms in this example. Furthermore, it can be observed that the distributed version of the PS-DSF allocation algorithm promptly converges when changes occur in the set of active users.

V. CONCLUSION

In summary, we studied the problem of multi-resource fair allocation for heterogeneous servers while respecting placement constraints. We identified important shortcomings in existing multi-resource fair allocation mechanisms when used in such environments. Hence, we proposed a new allocation mechanism, called PS-DSF. We discussed how our proposed allocation mechanism achieves different sharing properties which are satisfied under DRF in the case of one resource-pool/server. Furthermore, we discussed how PS-DSF could be implemented in a distributed manner. The performance of the PS-DSF allocation mechanism was compared against the existing allocation mechanisms and its enhanced performance was demonstrated through the numerical experiments. Further studies are under way and they will appear in future work.

REFERENCES

[1] V. Chudnovsky, R. Rifaat, J. Hellerstein, B. Sharma, and C. Das, “Modeling and synthesizing task placement constraints in google compute clusters,” in Symposium on Cloud Computing, 2011.

[2] A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica, “Choosy: Max-min fair sharing for datacenter jobs with constraints,” in Proc. ACM EuroSys, 2013, pp. 365–378.

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

[4] E. Friedman, A. Ghodsi, and C.-A. Pomes, “Strategynproof allocation of discrete jobs on multiple machines,” in Proceedings of the ACM conference on Economics and computation, 2014.

[5] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, “Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework,” IEEE/ACM Trans. Networking, vol. 21, no. 6, Dec. 2013.

[6] M. Chowdhury, Z. Liu, A. Ghodsi, and I. Stoica, “Hug: Multi-resource fairness for correlated and elastic demands,” in Proc. NSDI, Mar 2016.

[7] W. Wang, B. Liang, and B. Li, “Multi-resource fair allocation in heterogeneous cloud computing systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 10, pp. 2822–2835, Oct 2015.

[8] K. Yap, T. Huang, Y. Yiakominis, S. Chinchali, N. McKeown, and S. Katti, “Scheduling packets over multiple interfaces while respecting user preferences,” in Proc. ACM coNEXT, Dec. 2013.

[9] J. Khamse-Ashari, I. Lambadaris, and Y. Q. Zhao, “Constrained multi-user multi-server max-min fair queuing,” http://arxiv.org/abs/1601.04749, Jan. 2016.

[10] J. Khamse-Ashari, G. Kesidis, I. Lambadaris, B. Urgaonkar, and Y. Zhao, “Max-min fair scheduling of variable-length packet-flows to multiple servers by deficit round-robin,” in Proc. CISS, Princeton, Mar 2016.

[11] ——, “Constrained max-min fair scheduling of variable-length packet-flows to multiple servers,” in Proc. IEEE Globecom, Dec 2016.

[12] ——, “Efficient and fair scheduling of placement constrained threads on heterogeneous multi-processors,” in Proc. DConPerf, Atlanta, USA, May 2017.

[13] Y. Tahir, S. Yang, A. Koliosius, and J. McCann, “Udrf: Multi-resource fairness for complex jobs with placement constraints,” in Proc. ACM EC, Valencia, Spain, June 2012.

[14] W. Wang, B. Liang, and J. Li, “Multi-resource fair sharing for datacenter jobs with placement constraints,” SC 2016.

[15] ——, “Towards multi-resource fair allocation with placement constraints,” in Proc. ACM SIGMETRICS, Antibes, France, 2016.

[16] D. Parkes, A. Procaccia, and N. Shah, “Beyond dominant resource fairness: Extensions, limitations, and indivisibilities,” in Proc. ACM EC, Valencia, Spain, June 2012.

[17] 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 resource pools,” Tech. Rep., Nov., 2016, Available at http://arxiv.org/abs/1611.00404.

[18] C. Reiss, J. Wilkes, and J. L. Hellerstein, “Google cluster-usage traces,” 2011, http://code.google.com/p/googleclusterdata/.