An Appropriate Parameterized Utility Technique On Heterogeneous Server Dependencies

NAVA RAVI KIRAN
Student of M.Tech (CSE), Department of Computer Science & Engineering, Kakinada Institute of Engineering & Technology, Korangi, AP

D . SRINUVAS
Asst. Prof, Dept. of Computer Science & Engineering, Kakinada Institute of Engineering & Technology, Korangi, AP

Abstract: **A new server-based approach incorporated in Heterogeneous Servers. Current cloudinfrastructures are mostly homogeneous composed of a large number of machines of the same type – centrally managed and made available to the end user.In a cloud computing pattern, multiple resources types were utilizing. Users may have diverse resource needs. Furthermore, diversity in server properties/capabilities may mean that only a subset of servers may be usable by a given user. In platforms with such heterogeneity, we identify important limitations in existing multi-resource fair allocation mechanisms, notably Dominant Resource Fairness and its follow-up work. To overcome such limitations, we propose a new server-based approach; each server allocates resources by maximizing a per-server utility function. We propose a specific class of utility functions which, when appropriately parameterized, adjusts the trade-off between efficiency and fairness, and captures a variety of fairness measures. We establish conditions for the proposed mechanism to satisfy certain properties that are generally deemed desirable, e.g., envy-freeness, sharing incentive, bottleneck fairness, and Pareto optimality. To implement resource parameterized mechanism, we develop an iterative algorithm which is shown to be globally convergent on Heterogeneous server dependencies.**

Key words: **Heterogeneous Server; Cloud computing pattern; Utility function;**

INTRODUCTION

A. **Cloud computing**

It is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Large clouds, predominant today, often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close, it may be designated an edge server.

Clouds may be limited to a single organization, or be available to multiple organizations (public cloud). Cloud computing relies on sharing of resources to achieve coherence and economies of scale. Advocates of public and hybrid clouds note that cloud computing allows companies to avoid or minimize up-front IT infrastructure costs. Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and that it enables IT teams to more rapidly adjust resources to meet fluctuating and unpredictable demand, providing the burst computing capability: high computing power at certain periods of peak demand.

Cloud computing has become increasingly popular as high-performance computing systems. As the workloads to data-centers 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. Efficient and fair resource allocation in such a shared computing system is particularly challenging because of (a) the presence of multiple types of resources, (b) diversity in the workloads' needs for these resources, (c) heterogeneity in the resource capacities of servers, and (d) placement constraints on which servers may be used by a workload. In the following four paragraphs we briefly elaborate on each of these complexities.

The multi-resource needs of cloud workloads imply thatconventional single-resource oriented notions of fairnessare inadequate [1]. Dominant Resource Fairness (DRF) isthe first allocation mechanism which describes a notion offairness for allocating multiple types of resources for a singleserver system. Using DRF users receive a fair share of theirdominant resource [1]. Of all the resources requested by theuser (for every unit of work called a task), its dominantresource is the one with the highest demand when demandsare expressed as fractions of the overall resource capacities. DRF is shown to achieve several properties that arecommonly considered desirable from a multi-resource fairallocation mechanism.

Heterogeneity of workloads' resource demands is anothercomplexity which results in a trade-off between efficiencyand fairness. Specifically, heterogeneity of users' demandsmay preclude some resources from being fully utilized.Hence, the DRF allocation may result in a poor resourceutilization even when there is only one server [2], [3], [4].To address this issue, [2] proposed to allocate resources byapplying the so-called α-proportional fairness (instead ofmax-min fairness [5]) on dominant shares. The proposedmechanism, when appropriately parameterized, adjusts thetrade-off between efficiency and fairness. However, it isapplicable only to a single server/resource-pool.In the case of multiple heterogeneous servers, there are several studies investigating/extending DRF allocation whenthere is no placement constraint [6], [7], [8]. In all of theseworks, fairness is defined in terms of a global metric, a scalarparameter defined in terms of different resources across allservers. E.g., [7] presents an extension to DRF where thedominant resource for each user is identified as if all resources were concatenated at one server, and subsequentlythe resources are allocated by applying max-min fairness onthe dominant shares. Since such a global metric may notperfectly capture the impact of server heterogeneity, suchapproaches may lead to an inefficient resource utilization. Moreover, such mechanisms maynot be readily implementable in a distributed fashion [9],as each server needs information on the available resourcesover all servers. Such information may not be available ateach server, especially in a cloud computing environmentwhere the resource capacities (and even activity of servers)might be churning.

There are limited works in the literature investigatingmulti-resource fair allocation in the presence of user placement constraints [10], [11]. In this case, it is yet unclear howto globally identify the dominant resource as well as thedominant share for different users, as each one may have access only to a subset of servers. Work in [11] presents an extension to DRF identifying the user share by ignoring placement constraints and applying a similar approach as in an unconstrained setting. We show that this approach may not achieve fairness in the specific case that one of the resources serve as a bottleneck (see Section 2.2). In [12] we proposed a multi-resource *fair* allocation mechanism, called Per-Server Dominant Share Fairness (PSDSF), which is applicable to heterogeneous servers in the presence of placement constraints. The intuition behind PSDSF is to capture the impact of server heterogeneity bymeasuring the total allocated resources to each user explicitly from the perspective of each server. Specifically, PS-DSF identifies 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 respectto server *i*is defined as the ratio of *xn*- the total number of tasks allocated to user *n* - over the number of tasks executable by user *n* when monopolizing server *i*. Then the resources at each server are allocated by applying max-minfairness on VDS.This approach is amenable to a distributed implementation. It results in an enhanced performance over the existing mechanisms, and satisfies certain properties essential for fairallocation of resources [12].

B.Contributions

In this paper, we build upon and generalize our proposed PS-DSF allocation mechanism [12] to *capture the trade-off between efficiency and fairness*. We concisely summarize ourcontributions.

- We propose a new *server-based formulation* (which includes PS-DSF as a special case) to allocate resources while capturing server heterogeneity. The new formulation can be viewed as a *concave game* among different servers, where each server allocates resources by maximizing a *per-server* utility function.
- We study a specific class of utility functions which results in an extension of *α-proportional fairness on VDS*. We show how the resulting allocation, which we call *αPF-VDS*, captures the trade-off between efficiency and fairnessby adjusting the parameter *α*. We show that *α*PF-VDS satisfies bottleneck fairness, envy-freeness and sharing incentive properties (as defined in Section 2.1) for $\alpha \geq 1$,and Pareto optimality for$\alpha = 1$.
- We develop a (centralized) convergent algorithm to implement our proposed mechanism. Towards this, we introduce an equivalent formulation for which we derive an iterative solution (Section 4 and 5.1).
- We propose a simple heuristic to develop a distributed implementation for our resource allocation mechanism (Section 5.2).
- We carry-out extensive simulations, driven by real-worldtraces, to show the enhanced performance of our proposed mechanism (Section 6).

C. Game – Theoretic Approach:

There are several works in the literature which study the resource allocation problem in a cloud computing environment with a game-theoretic approach. Among these, [13], [14], [15] are limited to a *single-resource*setting, while [16], [17], [18] consider a *multi-resource* environment. In these studies, the multi-resource allocation problem is formulated as a *game*, where each server strives to maximize a per-server utility function. The utility function at each server is defined as the summation of resource utilization, minus the variance of a fairness-related metric for different users.

The work in [16] aims at minimizing the variance of global dominant shares for different users. However, since it chooses DRF as the underlying notion of fairness, it has the same limitations as DRF for heterogeneous servers (see Section 2.2 for a discussion of such limitations). In [17], [18], a two-stage mechanism is proposed wherein each user is initially assigned to a *server/coalitionof-servers*. Then, each server/coalition-of-servers strives to minimize the variance of the local dominant shares for the
assigned users, while maximizing the resource utilization. Such a local implementation of DRF, however, may not satisfy bottleneck fairness in the

whole system. Moreover,these works need to solve an extensive form game with a huge strategy set space [16], [18], which may not be implementable in a distributed fashion.

D. Single-resource fair scheduling

There are some recentworks investigating max-min fair scheduling for one type of resource while respecting placement constraints [19], [20], [21], [22], [23]. A deadline-aware scheduler is also proposed in [24] which assign CPU cores to different users in a waythat their deadlines are met in a fair manner. These single resource schedulers could be useful in a multi-resource setting when one of the resources (e.g. CPU) is dominantly requested by all users.

Implementation:
Consider a set *K* of *K* heterogeneous servers/resourcepools1 each containing *M* types of resources. We denoteby *ci,r*≥ 0, the capacity (i.e., amount) of resource *r*(1, 2, · · · , *M*) on server *i*. We make thereasonable assumption that all resources on each server are arbitrarily divisibleamong the users running on it. Let *N* denote the set of *N*active users. Let *φn*>0 denote the weight associated withuser *n*. The eights reflect the priority of users with respectto each other. Let **d***n*= [*dn,r*] denote the per task *demandvector* for user *n* ∈ *N*, i.e., the amount of each resourcerequired for executing one task for user *n*. Let *xn,i*∈R+denote the number of tasks that are allocated to user *n* fromserver *i*. Assuming linearly proportionate resource-needs2,*xn,i***d***n* = [*xn,idn,r*] gives the amounts of different resourcesdemanded by user *n* from server *i*.
Due to heterogeneity of users and servers, each user maybe restricted to get service *only from a subset of servers*.

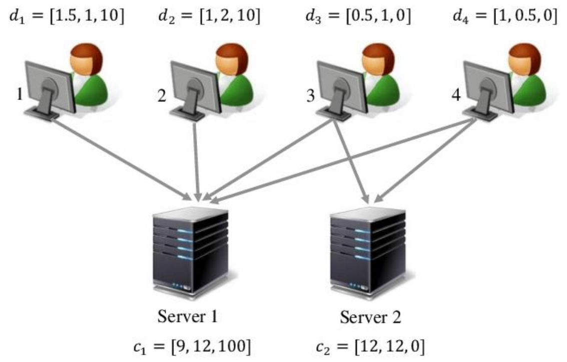

Fig. 1: A heterogeneous multi-resource system with two servers and four equally weighted users.

For example, users may not run tasks on servers which lack some required resources. Furthermore, each user may have some special hardware/software requirements (e.g., public IP address, a particular kernel version, GPU, etc.) which further restrict the set of servers that the user's tasks may run on. Let *Ni* = ∅denote the set of *eligible users* for server *i*. The placement constraints imply that *xn,i*= 0, *n* / ∈ *Ni*, ∀*i*. Such constraints are referred to as *hard placement-constraints*.

Soft-constraints, such as data-locality, are another type of constraints which describe preferences of each user over different servers [19]. For instance, consider the example in Fig. 1, where three types of resources, CPU, RAM, and network bandwidth are available over two servers in the amounts of c1 =[12 cores, 4GB, 75Mb/s] and c2 =[8 cores, 16GB, 0Mb/s], where no communication bandwidth is available over the second server; four users with their own demand vectors are also shown in the figure. In this example, the first two users require network bandwidth for execution of their tasks, so they are not eligible to run tasks on the second server. However, the last two users may run tasks on both servers.

DOMINANT RESOURCE FAIRNESS

Multi-resource fair allocation was originally studied in [1]under the assumption that all resources are aggregatedat one resource-pool. Specifically, let *cr*denote the totalcapacity of resource *r*. Let **a***n* = [*an,r*] denote the amountsof different resources allocated to user *n* under some allocation mechanism. The utilization of user *n* of its allocatedresources, *Un*(**a***n*), is defined as the number of tasks, *xn*,which could be executed using **a***n*, that is:*Un*(**a***n*) *xn*= min*ran,rdn,r*. (1)
In [1] the following properties are deemed desirable for amulti-resource allocation mechanism.
• *Sharing incentive:* Each user is able to run more taskscompared to a *uniform allocation* where each user *n* isallocated a *φn/ m φm*fraction of each resource.
• *Envy freeness:* A user should not prefer the allocationvector of another user when adjusted according to theirweights, i.e., it should hold that *Un*(**a***n*) ≥ *Un*(*φφmn***a***m*) forall *n*, *m*.
• *Bottleneck fairness:* If there is one resource which is *dominantly requested by every user*, then the allocation satisfies*max-min fairness* for that resource.
• *Pareto optimality:* It should not be possible to increase thenumber of tasks *xn*for any user *n*, without decreasing *xm*for some other user(s).
• *Strategy proofness:* Users should not be able to increasetheir utilization by erroneously declaring their resourcedemands.
The reader is referred to [1] or [25] for further details.Sharing incentive provides some sort of performance isolation, as it guarantees a minimum utilization for eachuser irrespective of the demands of the other users. Envyfreeness embodies the notion of fairness. Bottleneck fairnessdescribes anecessary condition which applies to a specificcase that one resource is dominantly requested by every user, so that a *single-resource* notion of fairness is applicable.These three properties are essential to achieve fairness. So,we refer to them as *essential fairness-related* properties. Paretooptimality is a

benchmark for maximizing system utilization. Finally, strategy proofness prevents users from gaming the allocation mechanism. In our view these properties areapplicable mainly for private settings. In public settings,users pay explicit costs for their usage or allocations andthe provider's goal is to maximize its profits subject to allocation guarantees for users. Even for private clouds, strategy proofness would only be necessary in settings where usersact selfishly. In many private settings, users are cooperativeand here strategy proofness is not needed. In view of this,we will not consider strategy proofness.

DRF is the first multi-resource allocation mechanismsatisfying all the above properties.Specifically, for everyuser *n*, the *Dominant Resource* (DR) is defined as [1]:$\rho(n)$:= argmax*rdn,r/cr*, (2) that is, the resource whose greatest portion is required forexecution of one task for user *n*. The fraction of the DR thatis allocated to user *n* is defined as its *dominant share*:*sn*:=*an,ρ*(*n*)*cρ*(*n*). (3)

Without loss of generality, we may restrict ourselves tonon-wasteful allocations, i.e., **a***n* = *xn***d***n*, ∀*n*.

Hence, anallocation *{xn}* is feasible when: *nxndn,r*≤ *cr*, ∀*r*. (4)

Definition 1. *An allocation {xn} satisfies* DRF*, if it is feasibleand the weighted dominant share for each user, sn/φncannot beincreased while maintaining feasibility without decreasing smforsome user m with sm/φm≤ sn/φn[1].*

DRF is a restatement of max-min fairness in terms of dominant shares. What make it appealing are the desirable properties which are satisfied by this allocation mechanism.

A. Existing challenges with heterogeneous serversand placement constraints
In case of heterogeneous servers (whether there are anyplacement constraints or not), a naturalapproach to extendDRF is to identify a system-wide dominant resource for each user, *as if* all resources were concatenated within a *single virtual server*. Specifically, let *cr*:= *ici,r*denote the total capacity of resource *r* within such a virtual server. Then, one may identify the dominant resource for each user *n* according to (2). Furthermore, the *global dominant share* for user *n* is given by: *sn*= *xn*max*rdn,rcr*, (5) where *xn*is the total number of tasks that are allocated to user *n* from different servers, that is *xn*:= *ixn,i*. As in Definition 1, one may find an allocation *{xn,i}* which satisfies max-min fairness in terms of the global dominant shares [7]. Such an allocation, referred to as DRFH (**DRF** for **H**eterogeneous servers [7]), is shown to achieve *Pareto optimality* and *envy freeness*.

However, it fails to provide sharing incentive [7]. We believe that the definition of bottleneck fairness employed by DRFH (with respect to a single virtual server that aggregates all resources) is also controversial. Specifically, if all users have the same dominant resource (with respect to the above mentioned virtual server), then DRFH satisfies max-min fairness with respect to such a resource [7].

In case of heterogeneous servers with placement constraints, however, one may consider other conditions under which a resource serves as a bottleneck.

Definition 2. *A resource ρ is said to be a* bottleneck *if for everyserveri:dn,ρci,ρ≥dn,rci,r, r, n, Ni. (6)*

If there exists a bottleneck resource, then the allocation should satisfy max-min fairness with respect to that resource. Unfortunately, DRFH does not satisfy bottleneck fairness in the sense of Definition 2. To appreciate this shortcoming of the DRFH mechanism, consider the example in Fig. 1, where the second resource (RAM) is dominantly requested by eligible users at each server. According to Definition 2, RAM is identified as the bottleneck resource in this example.

To allocate the RAM resources in a fair manner, each user should be allocated x1 = x1,1 = 2 (i.e., two tasks from the first server), x2 = x2,1 = 6, x3 = x3,2 = 8 and x4 = x4,2 = 8 tasks, respectively (This allocation results from our proposed PS-DSF allocation mechanism [12]).

On the otherhand, the DRFH mechanism would instead identify network bandwidth as the dominant resource for the first two *users and RAM as the dominant resource for the last twousers. To achieve max-min fairness in terms of dominant shares, the DRFH mechanism allocates x1 = x2 = 3 and x3 = x4 = 8 tasks to each user. Under such an allocation, the RAM resources are not allocated in a fair manner to the first two users.*

Yet another extension of DRF, which applies to heterogeneous servers in the presence of placement constraints, is TSF [11].

*As in [11], we letγn,i*Let *γn*:= *iγn,i*be defined as the number of tasks executable for user n when monopolizing all servers as if there were no placement constraints. An allocation is said to satisfy Task Share Fairness (TSF), when *xn/γn*satisfies max-min fairness [11]. When there is onlyone server, then *xn/γn*results in the dominant share for each user n. In such a case, TSF reduces to DRF. In case of heterogeneous servers with placement constraints, TSF is shown to satisfy Pareto optimality, envy freeness and sharing incentive properties [11]. However, we show by example that this mechanism may not satisfy bottleneck fairness (neither in the sense of Definition 2, nor in the conventional sense based on considering a single virtual server introduced above [12]).

For instance, consider again the example in Fig. 1, where the second resource is identified as a bottleneck according to Definition 2. The number of tasks that each user may run in the whole cluster is$\gamma 1 = 4$, $\gamma 2 = 12$, and $\gamma 3 = \gamma 4 = 4+ 16 = 20$ tasks, respectively. Hence, each user is allocated$x1 = x1,1 = 5/3$, $x2 = x2,1 = 5$, $x3 = x3,1+x3,2 = 8+1/3 = 25/3$ and $x4 = x4,1 + x4,2 = 8+1/3 = 25/3$ tasks, accordingto the TSF mechanism, which differs from the fair allocationin this example.

B. Per-server dominant share fairness (PS-DSF)

In this subsection, we describe PS-DSF which we introduced in [12]. PS-DSF is an extension to DRF which is applicable for heterogeneous servers in the presence of placement constraints. The core idea of this mechanism is to introduce a "virtual dominant share" for every user, with respect to each server. Towards this, we first identify the dominant resource for every user n with respect to each server*i*, $\rho(n,i)$:=argmax*rdn,rci,r*. (7)

It is assumed that $\gamma n,i>0$ for all *n / Ni*. We set $\gamma n,i= 0$ if*n / N*

$$\gamma_{n,i} := \min_r \frac{c_{i,r}}{d_{n,r}} = \frac{c_{i,\rho(n,i)}}{d_{n,\rho(n,i)}},\ n \in \mathcal{N}_i.$$

Definition 3. *The Virtual Dominant Share (VDS) for user n Niwith respect to server i, sn,i, is defined as:*

$$s_{n,i} := \frac{x_n}{\gamma_{n,i}} = \frac{x_n d_{n,\rho(n,i)}}{c_{i,\rho(n,i)}}, \quad (9)$$

where xn= j∈Kxn,jis the total number of tasks *that areallocated to user n from every server j ∈ K.*We have the following conditions on anallocation, **x**:=

$$\sum_{n\in\mathcal{N}_i} x_{n,i} d_{n,r} \le c_{i,r},\ \forall i, r.$$
$$x_{n,i} = 0,\ n \notin \mathcal{N}_i,\ \forall i.$$

Definition 4. *An allocation* **x** *satisfies PS-DSF, if it is feasibleand the allocated tasks to each user, xncannot be increased (whilemaintaining feasibility) without decreasing xm,ifor some user mand server i with sm,i/φm≤ sn,i/φn.*Intuitively, *sn,i*gives a measure of the *total* allocatedresources to user *n* from the perspective of server *i*. Inparticular, *sn,i*gives the normalized share of the dominant resource for user *n* with respect to server *i*which should be allocated to it as if *xn*tasks were allocated resources solely from server *i*(see the right hand side of (9)). The reader may note that *sn,i*could be possibly greater than 1, as some tasks might be allocated to user *n* from other servers. According to PS-DSF, the available resources at each server *i* are allocated by applying (weighted) max-min fairness on *{sn,i}*. It can be seen that PS-DSF reduces to DRF when there is only one server.

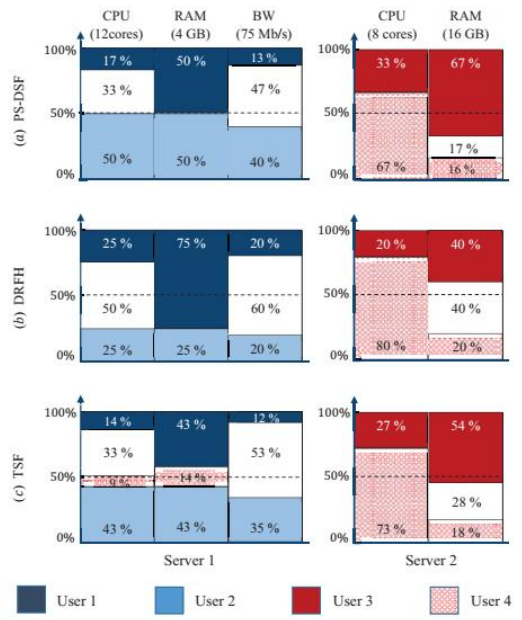

Fig. 2: Comparing the PS-DSF allocation with the DRFH and TSF allocations. The PS-DSF allocation mechanism is more efficient in utilizing different resources.

To gain more intuition, consider again the example inFig. 1, but this time let *d*4 = [1, 0.5, 0]. In this case, each usermay run $\gamma 1,1 = 4$, $\gamma 2,1 = 12$, $\gamma 3,1 = 4$, $\gamma 4,1 = 8$ tasks whenmonopolizing server 1. The third and the fourth users eachmay run $\gamma 3,2 = 16$ and $\gamma 4,2 = 8$ tasks when monopolizingserver 2. In order to satisfy PS-DSF, each user should beallocated $x1 = x1,1 = 2$, $x2 = x2,1 = 6$, $x3 = x3,2 = 32/3$and $x4 = x4,2 = 16/3$ tasks, respectively. Therefore, theVDS (c.f. Definition 3) for each user with respect to the firstserver is $s1,1 = s2,1 = 0.5$, $s3,1 = 8/3$ and $s4,1 = 2/3$. Also,the VDS for user 3 and 4 with respect to the second serveris $s3,2 = s4,2 = 2/3$. The reader can verify that for eachserver *i*the allocated tasks to any user may not be increased without decreasing the allocated tasks to another user witha less or equal VDS. Intuitively, the RAM resources are dominantly requested by eligible users of the first server in this example.

To achieve per-server dominant share fairness, the first server strives to allocate the RAM resources in a fair manner. Towards this, the last two users are not allocated resources from the first server, since there exists enough RAM for them over the second server. On the other hand, the second server identifies RAM as the dominant resource for User 3 and CPU as the dominant resource for User 4. To achieve PS-DSF, the second server needs to balance the respective dominant shares for User 3 and User 4. The resulting PS-DSF allocation is shown in Fig. 2. The DRFH and TSF allocations for this example are also illustrated in Fig. 2. Besides its desirable performance in terms of fairness, the PS-DSF allocation is observed to be more efficient in utilizing different resources compared to the DRFH and TSF mechanisms.

Table 1: Properties of different allocation mechanisms in case ofheterogeneous servers with placement constraints: sharing incentive(SI), envy freeness (EF), Pareto optimality (PO), and bottleneck fairness (BF).

Property	DRFH	TSF	PS-DSF
SI		Yes	Yes
EF	Yes	Yes	Yes
PO	Yes	Yes	
BF			Yes

Table 1:Compares essential sharing properties

which are satisfied under different allocation mechanisms [12]. The reader may note that PS-DSF does not satisfy Pareto optimality in general. It is worth noting that Pareto optimality may not also be satisfied in other works, e.g., [17], [18], which aim at developing a distributed implementation for DRFH. PS-DSF not only is amenable to distributed implementation (as we show in [12]), but also may lead to moreefficient utilization of resources compared to the DRFH and TSF mechanisms [12] (as also can be observed in Fig. 2, or in the trace-driven simulations in Section 6). The intuitive reason for this is that each of the DRFH and TSF allocation mechanisms allocates resources based on a global metric.

Since a global metric throws away information about the actual distribution of resources across servers, approaches based on it may not perfectly capture the impact of server heterogeneity, and therefore may lead to an inefficient resource utilization in heterogeneous settings.

In summary, PS-DSF has been shown to satisfy the essential fairness-related properties, i.e., envy-freeness, sharing incentive and bottleneck fairness, has been observed to offer highly efficient utilization of resources, and is amenable to distributed implementation [12].

PARAMETERIZED UTILITY TECHNIQUE ON HETEROGENEOUS SERVER DEPENDENCY

As already discussed, in most of the existing multi-resource allocation mechanisms, fairness is defined in terms of a global metric, a scalar parameter defined for each user in terms of different resources across all servers. Such mechanisms may not succeed in satisfying all the essential fairness-related properties (c.f. Section 2.2), may not readily be implementable in a distributed fashion, and may lead to inefficient resource utilization. In this section, we propose a new formulation for multi-resource allocation problem which is based on a per-server metric (as opposed to a global metric) for different users, so that server heterogeneity is captured. The proposed allocation mechanism is built upon our proposed PS-DSF allocation mechanism [12], which was briefly described in the previous section. It generalizes PSDSF in order to address the trade-off between efficiency and fairness. Furthermore, it inherits all the properties that are satisfied by PS-DSF.

The properties of the αPF-VDS allocation mechanism In this section, we investigate different properties which are satisfied by the αPF-VDS mechanism6. In case of heterogeneous servers with placement constraints, we need to extend the notion of sharing incentive property. The notion of bottleneck fairness has been extended by Definition 2. Other properties, Pareto optimality and envy freeness follow the same definitions as described in Section 2. To generalize the sharing incentive property, consider a uniform allocation, where a fraction φn/ m φm of the available resources over each server (whether this server is eligible or not) is allocated to each user n. An allocation is said to satisfy sharing incentive, when each user is able to run more tasks compared to such a uniform allocation.

A. Adjusting the resource utilization

As discussed in Section 3.2 and shown in the example ofFig. 3, the resource utilization improves as the parameter αin the αPF-VDS mechanism gets smaller. In this subsectionwe further investigate this effect when applying this mechanism to real-world workloads.

In the Bitbrain workload, users become active/incactivewith a relatively low rate. So, the resources could be allocated to different users/virtual-machines in a semi-staticmanner. In this case, we do several experiments for differentsets of active users chosen at random instants of time. Inparticular, for each set of active users we find the αPF-VDSallocation for α = 1, α = 3 and α = ∞. In case of α = 1and α = 3 we employ the distributed iterative algorithm proposed.

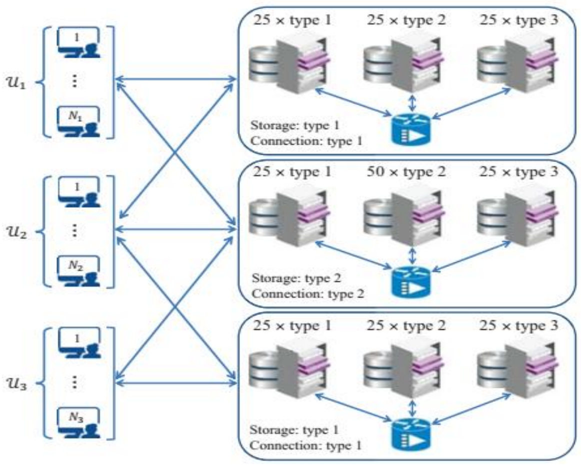

Fig. 2: A data-center distributed over three different locations.

There are three types of servers, where the configuration of resources (CPU and memory respectively) for each type of server is as follows: (4 GHz, 12 GBytes) for type 1, (8 GHz, 8 GBytes) for type 2, (16 GHz, 4 GBytes) for type 3. The storage devices are of two different types, where the read/write bandwidth for type 1, used at the first and the last cluster locations, is 32 MB/s, and for type 2, used at the second location, is 100 MB/s. Finally, there are two types of broadband connections, where the first type provides a bandwidth of 100 Mb/s (and 1 Gb/s respectively) to send (receive) data, while the second type provides a bandwidth of 1 Gb/s (and 2 Gb/s respectively) to send (receive) data.

For α = ∞, we use the customized algorithm proposed in [12] to implement PS-DSF. The latter is also available open-source at [34], where it is prototyped for cluster scheduling with Apache-Mesos. Fig. 5 shows the average processing time to find the PSDSF allocation and the αPF-VDS allocation for the computing cluster of Fig. 4, and for an expanded cluster where thenumber of users and servers are doubled. It can be observedthat the convergence time for the distributed iterative algorithm of Section 5.2 increases as α gets larger. Such anoverhead (which is less than 1 second in a cluster withthousands of users) is acceptable for modest values of α,especially in a semi-static setting where the same allocationcan be used for at least a few minutes. For the case α = ∞,the customized PS-DSF allocation algorithm offers muchless processing time (around 0.03 second), which remainsin the same range even for the expanded cluster.

In Fig. 2 we report the overall resource utilization thatis achieved on average over different servers and over100 runs, for different variants of αPF-VDS. As expected,the αPF-VDS results in a greater utilization of differentresources for smaller values of α. In this experiment, theimprovement in utilization could be significant as α rangesfrom ∞ to 1.

For the Google workload, we allocate resources in asemi-dynamic manner. In particular, consider the computing cluster described in Table 3, where 2% of users from theGoogle traces are randomly chosen as the input workload.

In such a setting, we decide to (re)allocate resources from the servers to demanding jobs (at least) every 5 minutes.

Specifically, given the resource usage for different tasks of each job by the Google traces, we may find the demand vector for each job (at the beginning of each 5 minutes interval) as the summation of the resource usage for different tasks (different tasks of the same job usually have proportional demands [32]). Given the total demand for each job, dn = [dn,r], we define an execution quantum for job n as a block of resources in the amount of d˜n := dn/ maxrdm,r

that is allocated to job n for 1 second. Accordingly, job n demands qn := 300 maxrdm,r execution quanta for the next 5 minutes interval. We use the normalized demand vectors, {d˜n}, as the input to the αPF-VDS mechanism in order to find the number of tasks that are allocated to each job under this mechanism. Given the allocated tasks to each job, the completion time for job n is given by qn/xn. If a job leaves the system during the 5 minutes period, the released resources are reallocated among the remaining jobs.

The number of execution quanta demanded by different jobs, and also their activity duration, span a quite wide range. Our observations over an interval of 24 hours show that around 38% of jobs are completed within a 5 min period, while 16% of them are active more than 24 hours.

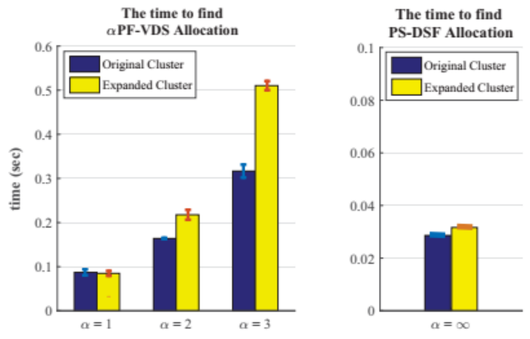

Fig. 3: The average processing time to find the PS-DSF allocation and the αPF-VDS allocation in the computing cluster of Fig. 2, and in an expanded cluster where the number of users and servers are doubled. The average is calculated over 10 different runs. The 95% confidence interval is shown at the top of each bar.

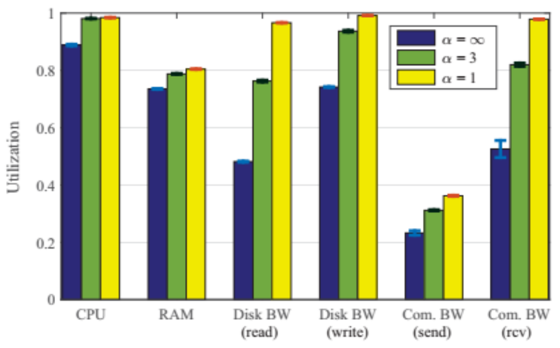

Fig. 4: The overall resource utilization, averaged over different servers of the computing cluster of Fig. 4 (serving the Bitbrain workload) and over 100 runs, for different variants of αPF-VDS: α = 1 (proportional fairness), α = 3, α = ∞ (PS-DSF). The 95% confidence interval is shown at the top of each bar.

COMPARISON WITH EXISTING MECHANISMS

In this subsection, we compare our proposed mechanismin terms of resource utilization against the known-proposedmulti-resource fair allocation mechanisms, which briefly described in Section 2.2.

Specifically, we compare the PS-DSFmechanism (which is the least efficient variant of αPF-VDS),against the DRFH and TSF allocation mechanisms (whichall are applicable to heterogeneous servers in the presence ofplacement constraints). First, consider the computing clusterof Fig. 3feeded by the Bitbrain workload. We employ eachof the aforementioned mechanisms to allocate resources ofthe servers in Fig. 3 to different sets of active users chosenat random instants of time. The overall utilization thatis achieved by of each of these mechanisms for differentresources, is shown in Fig. 4, when averaged over differentservers and over 100 runs. It can be observed that thePS-DSF allocation mechanism outperforms the two othermechanisms in terms of the achieved utilization for differentresources. In particular, the resource utilization is enhancedby the PS-DSF mechanism for up to 20% for some resources.

We make similar observations with the Google traces.Specifically, consider again the computing cluster describedin Table 3, where 2% of jobs in the Google traces are randomly chosen as the input workload. We employ each of thePS-DSF, DRFH and TSF allocation mechanisms to allocate

resources of the specified servers in Table 3 to demandingjobs over an interval of 24 hours. Fig. 4 compares theoverall resource utilization (averaged over different servers)that is achieved by different allocation mechanisms. It canbe observed that PS-DSF is again more efficient in utilizingdifferent resources, compared to the DRFH and TSF allocation mechanisms, while the achieved resource utilizationby DRFH and TSF mechanisms is almost the same11. Theoverall resource utilization that is achieved on average overthe 24 hour period is shown in Fig. 4 for different allocation mechanisms. The resource utilization over the last twoclasses of servers is also shown in Fig. 5. It can be observedthat the PS-DSF allocation improves the resource utilizationover the last two classes of servers more significantly.

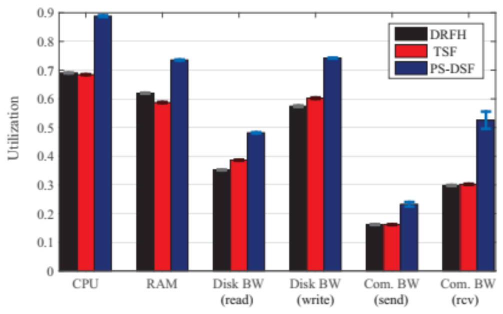

Fig.5 The overall resource utilization, averaged over different servers of the computing cluster of Fig. 6 (serving the Bitbrain workload) and over 100 runs, for different allocation mechanisms. The 95% confidence interval is shown at the top of each bar.

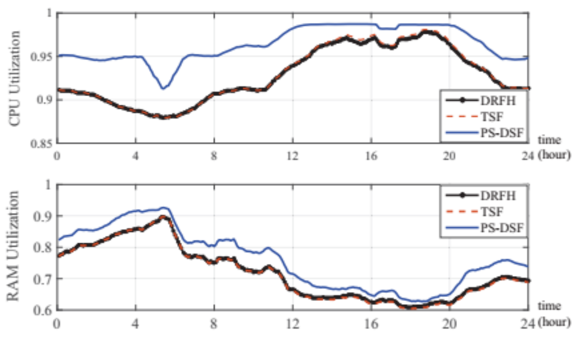

Fig. 6: The overall resource utilization (averaged over different servers of the Google cluster) that is achieved by different allocation

mechanisms during an interval of 24 hours. To get a better view, a moving average with a window size of 1 hour is applied to all plots.

Intuitively, the PS-DSF allocation mechanism allocatesresources at each server based on the per-server virtualdominant shares. So, at each server it gives more priority tousers which may run more tasks (c.f. (9)). Hence, one mayexpect that the PS-DSF allocation mechanism results in agreater resource utilization compared to the DRFH and TSFmechanisms, especially when the resources are heterogeneously distributed over different servers. That is the reasonwhy a more significant increase in utilization is achieved bythe PS-DSF allocation over the last two classes of servers,where the resources are more heterogeneously distributed(the available resources over the first two classes of servers.

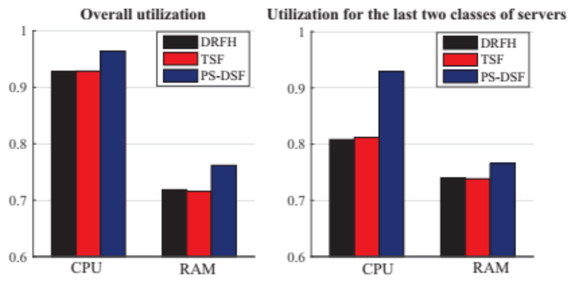

Fig. 7: The overall resource utilization that is achieved on average inthe Google cluster over an interval of 24 hours.

in Table 3 are almost proportional to the overall resourcecapacities). This is also consistent with our observation inthe first experiment (with the Bitbrain workload), wherethe variety of resources along with the heterogeneity ofservers results in a significant outperformance by the PSDSF mechanism.

CONCLUSION

We studied an Appropriate Parameterized Utility Technique on Heterogeneous Server Dependencies in thepresence of placement constraints. We identified potential limitations in the existing multi-resource fair allocation and parameterized mechanisms, DRF and its follow up work, when used in suchenvironments. In certain occasions, they may not succeed insatisfying all of the essential fairness-related parameterized properties, maynot be readily implementable in a distributed fashion, andmay lead to inefficient resource utilization. We proposeda new server-based approach to efficiently allocate resourceswhile capturing server heterogeneity. We showed how ourproposed αPF-VDS mechanism could be parameterized (byα) to adjust the trade-off between efficiency and fairness.

Distributed parameterized implementation usually comes at the price ofdegrading the performance. Our proposed mechanism notonly is amenable to distributed implementation, but alsoresults in an enhanced resource utilization compared to theexisting mechanisms. We carried out extensive simulations,driven by real-world traces, to demonstrate and implementable in a distributed fashion.

REFERENCES

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

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

[3] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, andA. Akella, “Multi-resource packing for cluster schedulers,” SIGCOMM Rev., vol. 44, no. 4, pp. 455–466, Aug. 2014.

[4] T. Bonald and J. Roberts, “Enhanced cluster computing performance through proportional fairness,” Performance Evaluation, vol. 79, pp. 134–145, 2014.

[5] D. Bertsekas and R. Gallager, Data networks. Prentice Hall, 1992.

[6] E. Friedman, A. Ghodsi, and C.-A. Psomas, “Strategyproof allocation of discrete jobs on multiple machines,” in Proc. ACM Conf. On Economics and Computation, June 2014.

[7] W. Wang, B. Liang, and B. Li, “Multi-resource fair allocation inheterogeneous cloud computing systems,” IEEE TPDS, vol. 26, no. 10, pp. 2822–2835, Oct 2015.

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

[9] Q. Zhu and J. C. Oh, “An approach to dominant resource fairnessin distributed environment,” in Proc. IEA-AIE, May 2015.

[10] Y. Tahir, S. Yang, A. Koliousis, and J. McCann, “Udrf: Multiresource fairness for complex jobs with placement constraints,” in GLOBECOM, Dec 2015, pp. 1–7.

[11] W. Wang, B. Li, B. Liang, and J. Li, “Multi-resource fair sharing fordatacenter jobs with placement constraints,” in Proc. IEEE/ACM Supercomputing, Nov. 2016.

[12] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar, andY. Zhao, “Per-server dominant-share fairness (ps-dsf): A multiresource fair allocation mechanism for heterogeneous servers,” inProc. IEEE ICC, May, 2017.

[13] J. Bredin, R. T. Maheswaran, C. Imer, T. Bas¸ar, D. Kotz, andD. Rus, “A game-theoretic formulation of multi-agent resource allocation,” in Proc. Autonomous Agents, 2000, pp. 349–356.

[14] V. Jalaparti and G. D. Nguyen, “Cloud resource allocation games,”Tech. Rep., 2010.

[15] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, “A game-theoreticmethod of fair resource allocation for cloud computing services,” The journal of supercomputing, vol. 54, no. 2, 2010.

[16] X. Xu and H. Yu, “A game theory approach to fair and efficientresource allocation in cloud computing,” Mathematical Problems in Engineering, vol. 2014, 2014.

[17] Q. Zhu and J. C. Oh, “Learning fairness under constraints: Adecentralized resource allocation game,” in Proc. IEEE ICMLA. IEEE, 2016, pp. 214–221.

[18] “Equality or efficiency: A game of distributed multi-type fairresource allocation on computational agents,” in Proc. IEEE/ACM WI-IAT, vol. 2, 2015, pp. 139–142.

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

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

[21] 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.

[22] J. Khamse-Ashari, G. Kesidis, I. Lambadaris, B. Urgaonkar, andY. Zhao, “Efficient and fair scheduling of placement constrained threads on heterogeneous multi-processors,” in Proc. IEEE DCPerf,Atlanta, USA, May 2017.

[23] “Constrained max-min fair scheduling of variable-lengthpacket-flows to multiple servers,” Springer Annals of Telecom., Aug 2017.

[24] S. Dimopoulos, C. Krintz, and R. Wolski, “Justice: A deadlineaware, fair-share resource allocator for implementing multianalytics,” in proc. IEEE Cluster Computing, Sep 2017, pp. 233–244.

[25] D. Parkes, A. Procaccia, and N. Shah, “Beyond dominant resourcefairness: Extensions, limitations, and indivisibilities,” in Proc. ACMEC, Valencia, Spain, June 2012.

[26] J. Mo and J. Walrand, “Fair end-to-end window-based congestioncontrol,” IEEE/ACM Trans. Networking, vol. 8, no. 5, Oct 2000.

[27] J. B. Rosen, “Existence and uniqueness of equilibrium points forconcave n-person games,” Econometrica: Journal of the EconometricSociety, pp. 520–534, 1965.

[28] J. Khamse-Ashari, I. Lambadaris, G. Kesidis, B. Urgaonkar, andY. Zhao, “An efficient and fair multi-resource allocation mechanism for heterogeneous servers,” Tech. Rep., Dec., 2017, Availableat

http://arxiv.org/abs/1712.10114.

AUTHOR’s PROFILE

NAVA RAVI KIRAN is pursuing M.Tech(CSE) in the department of CSE from Kakinada Institute of Engineering & Technology,Korangi.

D.SRINUVAS is working as an Assistant Professor in Department of CSE, Kakinada Institute of Engineering & Technology ,Korangi,Kakinada