Towards Multi-Resource Fair Allocation with Placement Constraints

Wei Wang[†], Baochun Li[‡], Ben Liang[‡], Jun Li[‡]
[†]Hong Kong University of Science and Technology, [‡]University of Toronto
weiwa@cse.ust.hk, {bli, liang, junli}@ece.utoronto.ca

ABSTRACT

Multi-resource fair schedulers have been widely implemented in compute clusters to provide service isolation guarantees. Existing multi-resource sharing policies, notably *dominant resource fairness* (DRF) and its variants, are designed for unconstrained jobs that can run on all machines in a cluster. However, an increasing number of datacenter jobs specify *placement constraints* and can only run on a particular class of machines meeting specific hardware/software requirements (*e.g.*, GPUs or a particular kernel version). We show that directly extending existing policies to constrained jobs either compromises isolation guarantees or allows users to gain more resources by deceiving the scheduler. It remains unclear how multi-resource fair sharing is defined and achieved in the presence of placement constraints. We address this open problem by a new sharing policy, called *task share fairness* (TSF), that provides provable isolation guarantees and is strategy-proof against gaming the allocation policy. TSF is shown to be envy-free and Pareto optimal as well.

1. INTRODUCTION

Datacenter jobs are characterized by a high degree of demand diversity across *multiple resource types*. For example, business analytics jobs are usually CPU-intensive; machine learning jobs are often memory- or I/O-bound. Multi-resource fair allocation is therefore needed to provide *predictable service isolation*, where a job is guaranteed to receive a fair share of datacenter resources, regardless of the other jobs' behaviors.

Dominant resource fairness (DRF) [2] is the *de facto* sharing policy for multi-resource allocation. With DRF, jobs receive the same share of *dominant resource*—defined for each job as the one whose percentage share is the maximum across all resources. DRF is proved to possess a number of highly desirable sharing properties—notably *sharing incentive* and *strategy-proofness*—and is widely implemented as a resource scheduler in systems like Hadoop YARN and Apache Mesos.

SIGMETRICS '16 June 14-18, 2016, Antibes Juan-Les-Pins, France

ACM ISBN 978-1-4503-4266-7/16/06.
DOI: http://dx.doi.org/10.1145/2896377.2901493

We note that the successful application of DRF is limited to *unconstrained jobs* that can run on all machines in a datacenter, provided that the machine has sufficient resources to accommodate a job. However, incompatibilities between machine configurations and prerequisites of job execution are frequently encountered in datacenters. As a result, an increasing number of jobs specify *placement constraints*, restricting them to run on a particular class of machines meeting specific hardware/software requirements [4]. For example, a CUDA job must run on machines with GPUs; a DNS service requires machines with public IP addresses. It is reported in [4] that approximately 50% of Google's cluster jobs have simple, yet restrictive, placement constraints. We show that directly applying DRF and its variants to these jobs compromises fairness guarantees. It remains unclear how multi-resource fair allocation should be defined and achieved for jobs with placement constraints.

In this paper, we answer this open question with a new sharing policy, called *task share fairness* (TSF). We show that TSF satisfies a number of desirable sharing properties that are widely recognized to be most important for cluster resource management [2, 3]. In particular, with TSF, jobs are better off sharing the datacenter dynamically (*sharing incentive*); no job can receive more resources by lying about its demands and/or constraints (*strategy-proofness*); no job would envy the allocation of another (*envy-freeness*); no job can increase its allocation without decreasing that of another (*Pareto optimality*). To our knowledge, TSF is the first multi-resource allocation policy meeting all these important properties in the presence of placement constraints.

2. DESIRABLE SHARING PROPERTIES

In this paper, we focus on *hard, non-combinatorial* placement constraints (*i.e.*, a job's constraint is independ of the other jobs' placements). In particular, we model constraints as a bipartite graph consisting of job vertices and machine vertices. An edge is connected between a job vertex and a machine vertex if the job can run on the machine. We model a datacenter as a shared cluster consisting of a number of machines with multiple resources, *e.g.*, CPU, memory, *etc.* Each job runs many parallel tasks. A task is characterized by a *demand vector*, which specifies the amount of resources needed during the task's runtime.

As observed by DRF [2], there are four properties that *any* datacenter scheduler should satisfy: sharing incentive, strategy-proofness, envy-freeness, and Pareto optimality.

Sharing incentive. Consider some *arbitrary* original allocation of resources where each job is given a dedicated

Table 1: Properties of DRF variants and TSF in the presence of constraints: sharing incentive (SI), strategy-proofness (SP), envy-freeness (EF), and Pareto optimality (PO).

Property	Per-Machine DRF	DRFH	CDRF	TSF
SI			✓	✓
SP	✓	✓		✓
EF	✓	✓		✓
PO		✓	✓	✓

resource pool to run its tasks. Suppose now the jobs share their resources with others. A new allocation of the shared resources is said to provide *sharing incentive*, if it allows each job to run no fewer tasks than the job would have run in its original dedicated resource pool.

Strategy-proofness. No job can run more tasks by lying about its resource demands and/or constraints.

Envy-freeness. Assume job i can run n_i tasks with its own allocation and can run $n_{i\leftrightarrow j}$ tasks after exchanging its allocation with another job j. We have $n_i \geq \frac{w_i}{w_j} n_{i\leftrightarrow j}$ for all i and j, where w_i and w_j are the *weights* of two jobs. The weight of a job can be computed from its original dedicated resource pool as defined in the sharing incentive property.

Pareto optimality. Any attempt to launch more tasks for a job results in fewer tasks for another.

It is worth mentioning that our requirement of sharing incentive generalizes that of existing work as the dedicated resource pools given to jobs can be *arbitrary*, while existing work only considers *equal partitioning* [1, 2, 5] (each of N jobs is given $1/N$ of the total resources). In addition, jobs now have *two-dimensional* strategies, one for demand and another for constraints. Existing work, however, can only handle one-dimensional strategies, where jobs either game the demands [1, 2, 5] or the constraints [3], but not both.

Can the four properties be satisfied using existing sharing policies? To answer this question, we start with CMMF allocation [3] that generalizes max-min fairness to constrained jobs for single-resource sharing. However, because CMMF is a *market-based* allocation [3], it is not strategy-proof in the multi-resource environment [2]. We next turn to three DRF variants that generalize DRF to a compute cluster consisting of multiple heterogeneous machines: Per-Machine DRF [1, 5], DRFH [5], and CDRF [1]. Table 1 summarizes their properties. We see that none of them retains all the required properties in the presence of placement constraints.

3. TASK SHARE FAIRNESS

We propose a new multi-resource sharing policy, called *task share fairness* (TSF), that retains all the properties described in §2 for constrained jobs (see Table 1). TSF computes the *task share* for each job, defined as the ratio between the number of tasks allocated and the maximum number of tasks the job can run if *we remove its constraints and allocate it the entire datacenter*. One can interpret task share as the *job slowdown* due to resource sharing and placement constraints. TSF simply applies *max-min fair allocation with respect to the jobs' task share*.

A toy example of TSF. We consider a 3-node cluster shown in Fig. 1, where machines m_1 and m_3 have the same configuration of $\langle 9 \text{ CPUs}, 12 \text{ GB RAM}\rangle$, while m_2 has $\langle 3 \text{ CPUs}, 4 \text{ GB RAM}\rangle$. There are three jobs. The task of j_1 demands $\langle 1 \text{ CPU}, 2 \text{ GB}\rangle$ and can be executed on all machines but m_3; the task of j_2 demands $\langle 3 \text{ CPUs}, 1 \text{ GB}\rangle$ and can run on m_2 only; the task of j_3 demands $\langle 1 \text{ CPU}, 4 \text{ GB}\rangle$ and can run on all machines. Suppose that we have allocated 6 tasks to j_1 on m_1, 1 to j_2 on m_2, and 3 to j_3 on m_3. We compute the task shares of these three jobs as follows.

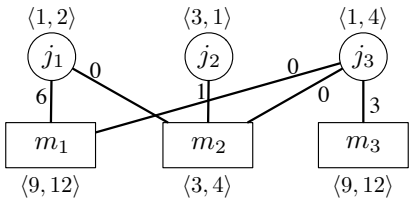

Figure 1: An example of a TSF allocation.

- For j_1, we remove its constraints and let it monopolize the entire cluster. In this hypothetical scenario, j_1 can run 14 tasks: 6 on m_1, 2 on m_2, and 6 on m_3. Therefore, the task share of j_1 is $\frac{6}{14} = \frac{3}{7}$.
- For j_2, we remove its constraints and let it monopolize the cluster. This allows j_2 to run 7 tasks: 3 on m_1, 1 on m_2, and 3 on m_3. The task share of j_2 is $\frac{1}{7}$.
- For j_3, its tasks can be executed on all machines. If monopolizing the cluster, j_3 can run 7 tasks: 3 on m_1, 1 on m_2, and 3 on m_3. The task share of j_3 is $\frac{3}{7}$.

To see that the allocation above realizes TSF, we first show that the minimum task share (*i.e.*, $\frac{1}{7}$ for j_2) is maximized. This is because j_2 is allocated the entire m_2—the only machine j_2 can run tasks on—so that its task share cannot be further increased. We next see that the second lowest task share is also maximized, because both j_1 and j_3 receive the same share $\frac{3}{7}$, and no more task can be allocated.

We may generalize TSF to weighted jobs. In particular, an allocation is said to achieve *weighted TSF* if any attempt to allocate more tasks to a user would result in fewer tasks allocated to another with an equal or lower *weighted task share*, defined for each job as the task share normalized by its weight. The following theorem states that weighted TSF retains all the required properties described in §2.

Theorem 1. *Assume originally each job i is given a dedicated resource pool and can run k_i tasks in it. If jobs instead share their resources and re-allocate them using TSF with weight $w_i = k_i/h_i$ for job i, the allocation is strategy-proof, envy-free, Pareto optimal, and provides sharing incentive.*

4. REFERENCES

[1] E. Friedman, A. Ghodsi, and C.-A. Psomas. Strategyproof allocation of discrete jobs on multiple machines. In *ACM EC*, 2014.

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

[3] A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *ACM EuroSys*, 2013.

[4] B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat, and C. R. Das. Modeling and synthesizing task placement constraints in google compute clusters. In *ACM SoCC*, 2011.

[5] W. Wang, B. Liang, and B. Li. Multi-resource fair allocation in heterogeneous cloud computing systems. *IEEE Trans. Parallel Distrib. Syst.*, 26(10):2822–2835, 2015.