Strategyproof Allocation of Discrete Jobs on Multiple Machines

ERIC FRIEDMAN, ICSI and UC Berkeley
ALI GHODSI, ICSI and UC Berkeley
CHRISTOS-ALEXANDROS PSOMAS, ICSI and UC Berkeley

We present a model for fair strategyproof allocations in a realistic model of cloud computing centers. This model has the standard Leontief preferences but also captures a key property of virtualization, the use of containers to isolate jobs. We first present several impossibility results for deterministic mechanisms in this setting. We then construct an extension of the well known dominant resource fairness mechanism (DRF), which somewhat surprisingly does not involve the notion of a dominant resource. Our mechanism relies on the connection between the DRF mechanism and the Kalai-Smorodinsky bargaining solution; by computing a weighted max-min over the convex hull of the feasible region we can obtain an ex-ante fair, efficient and strategyproof randomized allocation. This randomized mechanism can be used to construct other mechanisms which do not rely on users' being expected (ex-ante) utility maximizers, in several ways. First, for the case of m identical machines one can use the convex structure of the mechanism to get a simple mechanism which is approximately ex-post fair, efficient and strategyproof. Second, we present a more subtle construction for an arbitrary set of machines, using the Shapley-Folkman-Starr theorem to show the existence of an allocation which is approximately ex-post fair, efficient and strategyproof. This paper provides both a rigorous foundation for developing protocols that explicitly utilize the detailed structure of the modern cloud computing hardware and software, and a general method for extending the dominant resource fairness mechanism to more complex settings.

Categories and Subject Descriptors: J.4 [**Social and Behavioral Sciences**]: Economics; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms: Theory; Economics

Additional Key Words and Phrases: Resource Allocation; Fair Division

1. INTRODUCTION

The problem of allocating jobs in a cloud computing center is of broad practical and theoretical interest. For the former, cloud computing centers (CCCs), both internal (such as used by Twitter) and external (such as Amazon's EC2) are used for a large fraction of the world's computing and growing rapidly. For the internal, the introduction of the dominant resource fairness ([Ghodsi et al. 2011]) protocol has spurred much work on strategyproof allocation for Leontief Economies ([Parkes et al. 2012], [Dolev et al. 2012], [Li and Xue 2013]) leading to important insights and unexpected connections between computer science ([Joe-Wong et al. 2012]) and economics ([Friedman et al. 2011]). DRF has been extended to be used inside routers ([Ghodsi et al. 2012]) as well as large organizations with hierarchies ([Bhattacharya et al. 2013]). In addition, it has

This work is supported by grants NSF-1216073, NSF-1161813, and the A.G. Leventis Foundation. See the Acknowledgements section before REFERENCES.
Author's addresses: Computer Science Division, Electrical Engineering and Computer Sciences Department, University of California at Berkeley, and International Computer Science Institute. Emails: E.Friedman, ejf@icsi.berkeley.edu; A.Ghodsi and C.A.Psomas, { alig , alexpsomi }@cs.berkeley.edu;

EC'14, June 8–12, 2014, Stanford, CA, USA.
http://dx.doi.org/10.1145/2600057.2602889

been deployed in production, running on thousands of nodes at Twitter in the Mesos resource manager ([Hindman et al. 2011]).

The basic model underlying the analysis of DRF is an extremely simplified setting in which there is a single large machine and job allocations can be fractional. While this has led to a widely used mechanism in CCCs, many issues, such as the discreteness of jobs and the distributed computing environment have been dealt with in ad-hoc manners. In this paper we consider a model which captures both the discreteness of jobs and the multiplicity of machines. Our model is both an extension of previous models and a variation as we capture key properties of virtual machines – containers and machine independence – that are hallmarks of modern CCC design.

The importance of containers is that they allow the underlying placement of jobs on machines to be invisible to the users. Many cloud providers (e.g. Amazon.com) indeed hide from users whether two container instances are co-located or not. The user is therefore given the illusion of owning a dedicated machine that is isolated from the other instances, even though she is running on a container on a machine that has other containers. There are many reasons for this. First, it enables container migration. Second, not revealing co-location avoids applications that rely on co-location, limiting the flexibility of the provider (once one customer relies on this, you have to support it forever or break their applications). Finally, cloud providers often do not want to reveal too much information about the exact technology they are using as that is a competitive advantage as well as a potential vulnerability that attackers can exploit. Thus, unlike previous papers on DRF, in our model the system directly allocates containers (the term is borrowed from the Linux literature, c.f. [linuxcontainers.org 2014]), which are isolated bundles of resources. This differs from the model in [Parkes et al. 2012] as users cannot combine bundles. For example, if a user needs 2 units of a certain resource to run a job, but for strategic reasons only requests 1 unit per job, then even if she gets allocated 2 jobs on the same machine she is unable to run her job, even though she does have a total of 2 units of the resource. Of course the user could execute separate tasks in each container and have them use the network to coordinate as if they were a single task, but this would have performance overheads (communication), as well as cost overheads (parallelizing the code). Furthermore, a task might have the consumption vector (3 CPU, 4 GB memory), and it cannot simply separate it into one (3 CPU, 3 GB memory) and one (0 CPU, 1 GB memory) tasks.

Jobs are also not encapsulated in the analysis of DRF with multiple machines in [Wang et al. 2013] as they allow fractional jobs. However, this encapsulation is central to our analysis, complicating algorithmic issues, but simplifying some strategic ones as well, since certain manipulations are clearly unprofitable. For example, as discussed above, if a user halves all their resource requirements for a job, then *every* encapsulated job that they receive will be of no value.

Our first set of results show the impossibility of satisfying sets of important properties in this model. These results are extensions of those in [Parkes et al. 2012]. Our main result is the construction of a fair, efficient and strategyproof randomized mechanism for this model, which we call "Containerized DRF" (CDRF). The CDRF mechanism is modeled on a game theoretic interpretation of the standard DRF mechanism and maintains all the positive aspects of DRF in this setting. Interestingly, we argue that the obvious extension of DRF by generalizing the notion of a dominant resource is flawed and instead work with the identification of the DRF mechanism as the Kalai-Smorodinski solution of a bargaining problem ([Kalai and Smorodinsky 1975]). In this interpretation, one can view the restrictions imposed by containers as creating a feasible region which is discrete, as opposed to the convex feasible region in the original DRF model. We show that by convexifying the feasible region we get a randomized allocation mechanism (CDRF) which is (ex-ante) fair, efficient and strategyproof.

One difficulty with our ex-ante analysis of randomized mechanisms is that it relies on the users being expected utility maximizers. This can be unrealistic because even when the expected outcome has good properties, the realized outcome might not. So, if users were risk averse, these mechanisms would lose many of their desirable properties. Thus, it is important to consider an analysis based on approximate ex-post properties of a mechanism. Such an ex-post analysis assumes that the user is always (approximately) satisfied with the outcome of the mechanism and is valid irrespective of the users' risk preferences. In order to construct mechanisms with good ex-post properties, we compute allocations that are close to the average allocation computed by CDRF. First, we show that in the case where there are at least m of every type of machine, one can directly construct an approximate CDRF mechanism by choosing the allocations of each machine according to a random distribution over allocations of the single machine. Then we show that in the general case we can apply the Shapley-Folkman-Starr theorem, which provides bounds for the non-convexity of Minkowski sums of non-convex sets, to construct a mechanism with good approximate ex-post properties.

As the CDRF mechanism is based on convexifying the set of feasible allocations it appears to be quite robust and should be directly extendable in a variety of ways. For example, it directly extends to the case where there are usable and unusable machines ([Ghodsi et al. 2013]), e.g. windows based jobs may not be able to run on linux based machines.

1.1. Previous Work

DRF was initially introduced in the context of cloud computing ([Ghodsi et al. 2011]). It was mainly used within the Mesos resource manager ([Hindman et al. 2011]), which is used at Twitter to schedule thousands of machines. Later, Hadoop MapReduce implemented the DRF model for their Capacity ([apache.org 2014a]) and Fair Schedulers ([apache.org 2014b]). It has also been extended to be used in real-time to do fine-grained packet scheduling inside routers ([Ghodsi et al. 2012]) as well as hierarchical scheduling within large organizations ([Bhattacharya et al. 2013]). Many of its more theoretical properties have been studied, e.g. efficient computation of DRF allocations ([Gutman and Nisan 2012]) and fairness-efficiency tradeoffs ([Joe-Wong et al. 2012]).

The two closest papers to our work are [Parkes et al. 2012] and [Wang et al. 2013]. The former studied the allocation of integer valued jobs while the latter looked at continuous valued jobs with multiple machines. While the former considered alternative properties to those in the original DRF paper, we focus on the original properties but add randomization and approximation to evade their impossibility results. The latter paper relies heavily on the ability to allocate fractional jobs and the use of a generalized dominant resource. However, this mechanism does not satisfy several desirable properties, such as sharing incentives, which encourages users to join the CCC, rather than building their own computing centers. Another violated property is independence of dummy machines, since the addition of useless machines, i.e. machines on which none of the users can run a single job (or fraction of one), can change the allocation significantly under their mechanism.

2. MODEL

Before the formal model we present an illustrating example (Figure 1):

Example: Consider a cluster of two machines, with two resources each: machine 1 has 4 CPU's and 6 GB RAM, while machine 2 has 4 CPU's and 4 GB RAM. Two users want to execute their tasks in this cluster. User 1 wants to execute tasks that require 1 CPU and 3 GB's of RAM each. User 2 wants to execute tasks that require 2 CPU's and 1 GB RAM. In other words, the users have demands $d_1 = (1, 3)$ and $d_2 =$

$(2, 1)$ respectively. In a cluster like this we will assume that each user gets allocated a set of *containers*; each container is a bundle of resources. A user can execute in each container she gets as many tasks as she can "fit". Her total utility is simply the total number of tasks she can execute in the cluster. So, say user 1 is allocated containers $c_1 = (1, 3)$ and $c_2 = (1, 1)$ in machine 1, and container $c_4 = (0, 2)$ in machine 2. User 2 gets allocated container $c_3 = (2, 1)$ in machine 1 and $c_5 = (4, 2)$ in machine 2. User 1 can only use c_1; c_2 and c_4 are too small. Moreover, she can't combine these two containers (this holds even if the containers were in the same machine). Thus, her utility is 1. Similarly, user 2 has utility 3: she can fit one task in c_3 and two tasks in c_5.

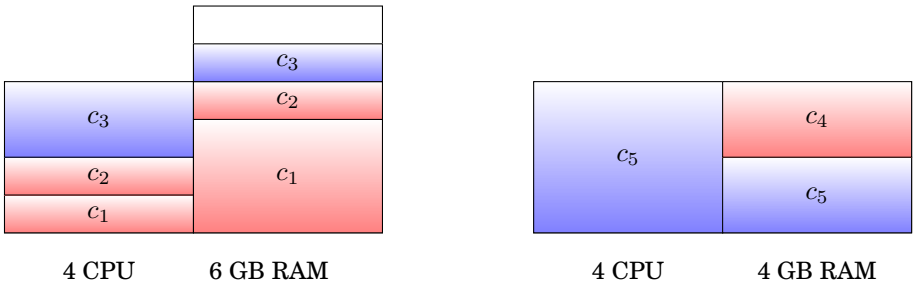

Fig. 1. Example of an allocation in a containerized setting with machines $r_1 = (4, 6)$ and $r_2 = (4, 4)$, and users $d_1 = (1, 3)$ and $d_2 = (2, 1)$. User 1 gets containers $c_1 = (1, 3)$, $c_2 = (1, 1)$ and $c_4 = (0, 2)$. User 2 gets containers $c_3 = (2, 1)$ and $c_5 = (4, 2)$

2.1. Formal model

A cluster has m machines, n users and p resources. Each machine j has a vector of resources $r_j \in \Re_+^p$, i.e. $r_{jk} \geq 0$ of resource k. We will refer to the cluster, the vector of machine vectors, as r. User i has true demand $d_i \in \Re_+^p$ for each job, where d_{ik} is the amount of resource k that she needs to execute a task. We will say that $d = (d_1, d_2, \ldots, d_n)$ is a demand profile.

If there was a single machine and user i was allocated a bundle $x \in \Re_+^p$ of resources, then in a continuous model she could run $\min_k(x_k/d_{ik})$ jobs, while in a model without fractional jobs she could only run $\lfloor \min_k(x_k/d_{ik}) \rfloor$ jobs. On the other hand, in a setting with many machines, such as ours, users shouldn't be able to combine resources across machines. The key to our model is that the system allocates containers. The s'th container $c_s \in \Re_+^p$ is a bundle of resources on a single machine, i.e. c_{sk} is the amount of resource k in the container s. In a containerized model, using container c_s user i can run $\lfloor \min_k(c_{sk}/d_{ik}) \rfloor$ jobs. Combining resources across containers is not possible.

An **allocation** $A =< C, M, P >$ consists of a set C of containers with a machine function $M(s) \in [m]$ and a user function $P(s) \in [n]$. $M(s)$ is the machine on which container s is located and $P(s)$ the user who gets container s. For an allocation A, the number of jobs that user i can execute is

$$u_i(A, d_i) = \sum_{\{s|P(s)=i\}} \lfloor \min_k(c_{sk}/d_{ik}) \rfloor$$

and note that the floor function is inside the sum. This number is exactly a user's utility. Observe that we have Leontief preferences: users require resources in fixed proportions. We will often write $u_i(A)$ when the d_i is implicit.

An allocation is feasible, if

$$\forall j \in [m], k \in [p] : \sum_{\{s|M(s)=j\}} c_{sk} \leq r_{jk}$$

We denote by $F(d)$ the feasible region, that is the set of all feasible allocations under submitted demand profile d. We will also use $F_j(d)$ for the feasible region of machine j.

A **mechanism** $A(d, r)$ is a function that takes as input a demand profile d and a cluster r and outputs an allocation. In much of the following we will consider a fixed set of machines and resources, so will often simply write $A(d)$ when r is implicit.

Our mechanisms work directly with the feasible region $F(d)$, so it is useful to define the following:

Definition 2.1. The Minkowski sum of sets $A_1, A_2, \ldots, A_k$ is the set

$$B = \{\sum_{i=1}^{k} x_i : x_i \in A_i\}$$

Observe that $F(d)$ is the Minkowski sum of the $F_j(d)$'s. We will later use the fact that the Minkowski sum of convex hulls is the convex hull of Minkowski sums.

2.2. Properties of mechanisms

We now consider several important properties of a good mechanism, almost all of which have been discussed in more detail for divisible jobs in the single machine setting in [Ghodsi et al. 2011] and [Parkes et al. 2012].

The first property we want our mechanisms to satisfy is *sharing incentives* (also known as individual rationality): every user must prefer the CCC than running her own computing facility with a fraction of the resources. To define this property, we first define user's i *stand alone allocation* in the CCC to be the number of jobs user i could run if she had the entire system for herself

$$sa_i(d, r) = \sum_{j=1}^{m} \lfloor \min_k (r_{jk}/d_{ik}) \rfloor$$

We will usually write sa_i to relax notation. A mechanism satisfies *Sharing Incentives* if every user's utility for the output allocation is at least her *fair share* $\lfloor sa_i/n \rfloor$.

A mechanism is *Pareto optimal* if the output allocation is efficient: there does not exist an allocation which is strictly better for (at least) one user and every other user is at least as well off.

Our users are selfish, i.e. user i will misreport her demand if that results in an allocation where she can execute more jobs. We say that a mechanism is *Strategyproof* if no user i can gain by misreporting her demand.

Next, we consider *population monotonicity*: A mechanism M is *population monotonic* if no user is harmed (receives fewer jobs) when some other user leaves the system. The last property we want our mechanisms to satisfy is independence of dummy machines, *IDM*, where a dummy machine is one on which no user can execute any job: A mechanism satisfies *IDM* if adding or removing dummy machines does not change the allocation. IDM is a simply proxy for robustness, since if a mechanism fails IDM then it will fail a variety of other important robustness considerations.

In the next couple of subsections we present some warm-up results in this multidimensional multiple-machine setting, concerning deterministic mechanisms that satisfy a subset of these properties.

2.3. Limitations of Deterministic Mechanisms

As in the single machine case, when we demand integer allocations no mechanism can satisfy sharing incentives, Pareto optimality, and strategyproofness simultaneously. For the model without containers and a single machine, this was proven by the following example in [Parkes et al. 2012]: consider a system with a resource of size 1 and two users with demands $d_1 = d_2 = \frac{1}{2} + \epsilon$. Assume w.l.o.g. that a mechanism that satisfies all 3 properties allocates to user 1. If user 2 reports $d_2' = \frac{1}{2}$ then sharing incentives dictates that she should get half the resource. But, since the remaining half cannot be used by user 1 and it can be used by user 2, Pareto-optimality dictates that user 2 gets that as well. So, user 2 has a profitable deviation and the mechanism is not strategyproof.

However, this example does not prove impossibility in our model. To see this consider a mechanism that allocates containers equal to demand vectors. Then user 2 would get containers of size $\frac{1}{2}$ when she reported d_2'. But she cannot execute any jobs in these containers, and she cannot combine them, so her utility when misreporting would be zero. Nonetheless, a slightly more complicated example does work as seen below:

THEOREM 2.2. *No deterministic mechanism can satisfy sharing incentives, Pareto optimality and strategyproofness simultaneously, even for one machine, two resources and two users.*

PROOF. Assume such a mechanism M exists and consider a cluster with one machine and two resources $r = (1, 1)$ and two users with demands $d_1 = (0.25, 0.1)$ and $d_2 = (0.1, 0.25)$. Since M satisfies sharing incentives then each user should be allowed to execute at least 2 tasks. A Pareto optimal mechanism should allocate one more job for either user 1 or user 2. Without loss of generality assume that M chooses the first option and allocates containers which result to utility 3 for user 1 and utility 2 for user 2.

Now, examine what happens when user 2 deviates with $d_2' = (\frac{1}{6}, 0.25)$. Again, sharing incentives requires M to allocate containers that give utility at least 2 to each user. This will use at least $(\frac{5}{6}, 0.7)$ of the available resources, leaving $(\frac{1}{6}, 0.3)$ available, which is enough for user 2 to schedule a task, but not user 1. Pareto optimality dictates that the remainder should be allocated to user 2. Since d_2 "fits" in d_2', user 2's utility strictly increases, thus she has an incentive not to report her true demand vector and so, M is not strategyproof, a contradiction. □

2.4. SI allocations always exist

Even though all three main properties are impossible to satisfy at the same time, it is very easy to construct a mechanism that satisfies only Pareto optimality and strategyproofness. For example, a serial dictatorship, in which we choose a fixed order of users and allow each user, in order, to allocate as many jobs as possible. However, it is not clear if there always exists an allocation that satisfies sharing incentives for every cluster. In this subsection we will prove that such an allocation always exists by showing the correctness of a moving knife-like algorithm ([Dubins and Spanier 1961]). In addition to providing a useful result for multiple machine settings, this will also be necessary for the correctness proof of our main mechanism.

Recall the moving knife procedure for allocating a divisible cake to n players, when each player's value for the whole cake is 1: the referee moves a knife from left to right until some player i calls "cut". Then the referee cuts at the point where the knife was and allocates to player i everything on the left of the cut. If each player calls "cut" when she thinks the amount of cake on the left of the knife is worth $\frac{1}{n}$ the allocation is fair: everybody gets utility at least $\frac{1}{n}$.

Now, in our setting, given an arbitrary ordering of the machines, imagine a water filling procedure for every user i that works as follows: starting from the first machine and following the ordering, user i progressively fills the cluster using a constant fraction of the resources. Every time she has "filled" enough to be able to allocate her fair share $\lfloor \frac{sa_i}{n} \rfloor$ she puts a mark $\mu_{i,k}$. So, the marks are such that, if she were allocated everything between two consecutive marks, she would get exactly her fair share, with the exception maybe of the last mark, where she would possibly get more if allocated everything from that mark until the end. There are $n-1$ such marks.

We will write $\mu_{i,k} = (f, j)$ to denote that the k-th mark of user i is in machine j, when she reached an f fraction of that machine. Notice that this doesn't mean that she needs an f fraction of machine j to get her fair share, since two consecutive marks can be many machines apart. We will say that $\mu_{i,k} \leq \mu_{j,k}$ if $\mu_{i,k}$ is a mark in a machine earlier in the ordering, or in the case that both marks are on the same machine, $\mu_{i,k}$ uses a smaller (or equal) fraction.

We are now ready to state the algorithm:

ALGORITHM 1: Moving Knife Algorithm

Input: Cluster $r = (r_1, \ldots, r_m)$ and demand profile d
Output: A feasible allocation A

Pick an arbitrary ordering of the machines
For each user i compute $n-1$ marks $\mu_{i,k}$
$S =$ set of users
$\mu = (0, 1)$ (mark the beginning of the first machine)
while $|S| > 1$ **do**
    Pick user $j \in S$ with smallest $\mu_{j,n-|S|+1}$ mark
    Allocate to j everything from μ until $\mu_{j,n-|S|+1}$
    $\mu = \mu_{j,n-|S|+1}$
    $S = S \setminus \{j\}$
end while
Allocate the rest to the last user

Example: Consider the cluster from the previous section (Figure 1), with an extra user $d_3 = (1, 2)$, and an extra machine with 4 CPU's and 2 GB RAM. User 1, with $d_1 = (1, 3)$, would be able to execute 3 jobs if alone in the system: 2 jobs in machine 1 and 1 job in machine 2, so her fair share is $\frac{3}{n} = 1$. She will put $n - 1 = 2$ marks: $\mu_{1,1} = (\frac{1}{2}, 1)$ and $\mu_{1,2} = (1, 1)$. Similarly user 2, with $sa_2 = 6$, would have marks $\mu_{2,1} = (1, 1)$ and $\mu_{2,2} = (1, 2)$. User 3, with $sa_3 = 6$ as well, would have $\mu_{3,1} = (\frac{2}{3}, 1)$ and $\mu_{3,2} = (\frac{1}{2}, 2)$, because she can get her fair share with $\frac{1}{3}$ of machine 1 and $\frac{1}{2}$ of machine 2. Figure 2 shows the execution of the algorithm in this example.

THEOREM 2.3. *The mechanism defined by the moving knife Algorithm 1 satisfies sharing incentives.*

PROOF. Assume some user i did not get her fair share of the system. She was allocated some part of the system either in the first step, in some step $k > 1$ or in the last step. In the first and last step cases we have an immediate contradiction from the description of the algorithm; the user was definitely allocated her fair share. The only interesting case is if she got picked in some step $k > 1$. In that case, we know that she was not picked in step $k - 1$ because there was some other user j that had a mark $\mu_{j,k-1} \leq \mu_{i,k-1}$. User i then got allocated everything between $\mu_{j,k-1}$ and $\mu_{i,k}$ which is more than everything between $\mu_{i,k-1}$ and $\mu_{i,k}$, which is by definition her fair share. □

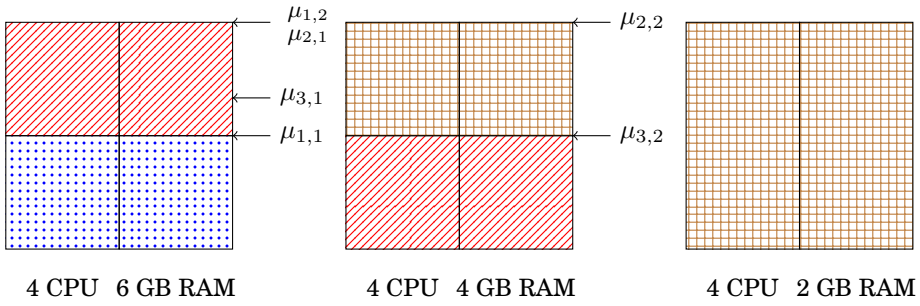

Fig. 2. Marks in a cluster with $r_1 = (4, 6), r_2 = (4, 4)$ and $r_3 = (4, 2)$, and users with demands $d_1 = (1, 3), d_2 = (2, 1)$ and $d_3 = (1, 2)$. After the execution of algorithm 1, the dotted area shows the fraction user 1 gets, the lined area shows the fraction user 3 gets, and the grid the fraction user 2 gets

3. CONTAINERIZED DRF

3.1. DRF and DRFH

To motivate our main mechanism, we first recall the standard DRF mechanism for a single machine with no integrality constraints. In the standard presentation, the dominant resource of user i is defined as the resource k^* which maximizes d_{ik}/r_k. A user's dominant share is $s_i = x_{ik^*}/r_{k^*}$, where x_{ik} is the amount of resource k user i gets allocated. Then, the DRF mechanism finds the max-min optimal allocation of the dominant shares. Note that in the case where $d_{ik} > 0$ for all users i this occurs when all users have the same dominant shares; however, if some $d_{ik} = 0$ then it is possible that there is no Pareto optimal allocation in which all users have the same dominant shares. Thus, we will focus on the broader interpretation of max-min optimization in which the max-min procedure uses a lexicographic order after ordering the shares. For example in this sense, the vector $(1, 2, 3)$ is dominated by $(3, 1, 4)$ even though user 2 has a lower value in the second vector, since when the vectors are sorted we see that $(1, 3, 4)$ is larger than $(1, 2, 3)$. Also recall that $(3, 2, 2)$ is larger than $(10, 1, 10)$ since its sorted first element is larger that the sorted first element of the second vector.

In the case of a single machine with divisible jobs, DRF satisfies all of our properties: sharing incentives, Pareto optimality, strategyproofness and population monotonicity simultaneously (IDM has no meaning in a single machine setting). However, as discussed in [Wang et al. 2013], one cannot directly extend DRF to the setting with multiple machines, even without integrality constraints on the jobs. For example, if one directly applies DRF to each machine separately, then the allocation may not be Pareto optimal: consider two machines $r_1 = (2, 12)$ and $r_2 = (12, 2)$, with two users $d_1 = (0.2, 1)$ and $d_2 = (1, 0.2)$. If one tries to equalize dominant resources in each machine separately, the resulting allocation will give 5 jobs in machine 1 and 1 job in machine 2 for user 1, and 1 job in machine 1 and 5 jobs in machine 2 for user 2. On the other hand, giving machine 1 to user 1 and machine 2 to user 2 allows them to schedule 10 jobs each. Thus, in [Wang et al. 2013] the authors construct a heterogeneous version of DRF which they call DRFH.

DRFH is directly modeled on DRF. It considers the aggregate of all the resources, $\sum_j r_j$, defines the dominant resource for user i using this aggregate resource vector and then computes the max-min optimal allocation in terms of the global dominant shares, which is the fraction of global dominant resources for each user.

One could directly extend DRFH to our containerized model, by maximizing the global dominant resource shares; however, this mechanism fails to satisfy some basic properties:

Example: Consider one machine with resource vector $r_1 = (15, 15)$ and two users with demand vectors $d_1 = (1, \frac{1}{2})$ and $d_2 = (\frac{1}{2}, 1)$. Under DRFH, 1's global dominant resource is resource 1, since in order to execute a single task she needs a $\frac{1}{15}$ fraction of resource 1 and a $\frac{1}{30}$ fraction of resource 2. Similarly, the global dominant resource of user 2 is resource 2. DRFH allocates 10 tasks to each user. Note that each user's stand alone share is $sa_i = 15$ so this allocation satisfies the resource sharing property. However, if we add a second machine with $r_2 = (16, 0)$ then even though this machine is unable to run a single job for either user, it changes the allocation significantly, giving 12 jobs to user 1 and 6 to user 2. This happens because user 1's dominant resource is now resource 2.

This example shows that DRFH need not satisfy the sharing incentives property and that it also lacks the important independence of dummy machines property, as the addition of dummy machine changes the allocation significantly. As we discuss in the next section, these problems can be resolved by a fundamental reinterpretation of the DRF mechanism without the notion of a dominant resource.

3.2. Randomized Mechanisms

As we showed in Theorem 2.2, one cannot exactly satisfy the basic properties with a deterministic mechanism. However, as we shall now show there does exist a randomized mechanism, which is the natural extension of DRF, that satisfies all of the desired properties on average (in an ex-ante sense). Since our mechanisms allocate containers equal to demands we will relax notation and refer to allocations simply as vectors $y \in \mathbb{N}^n$, where y_i is user's i utility, i.e. the number of jobs she can execute in that allocation. On the other hand, when talking about non integral solutions, allocations are vectors in $\mathbb{R}_+^n$. This distinction should be clear from the context.

We will restrict our randomized mechanisms to be a random mixture of a finite set of deterministic mechanisms. Our goal is to construct a randomized mechanism which satisfies all of the desirable properties on average before the randomization is revealed, i.e. ex-ante. We note that these do not necessarily satisfy the properties ex-post, that is when the actual allocation is initialized, and discuss this issue later.

For most properties ex-post is a stronger guarantee than ex-ante. Notice though that, somewhat surprisingly, ex-ante Pareto optimality implies ex-post Pareto optimality, but not the converse. This happens because in order to achieve ex-ante Pareto optimality one must make sure that the average allocation is not dominated. To see this distinction, consider a randomized serial dictatorship: pick a random order on the players and allocate as many jobs as possible to the first player in the order, then to the second player etc. This mechanism is fair, strategyproof and ex-post Pareto optimal, but not ex-ante Pareto optimal. For example, in a cluster with two users with demands $d_1 = (10, 1)$ and $d_2 = (1, 10)$ and two machines with $r_1 = (100, 10)$ and $r_2 = (10, 100)$ the randomized serial dictatorship yields allocations $(11, 0)$ and $(0, 11)$ with equal probability combining for an ex-ante allocation of $(5.5, 5.5)$. This is a dramatic loss of efficiency considering that allocation $(10, 10)$ is also feasible. Incidentally, $(10, 10)$ will be the output of our randomized mechanism with probability 1.

First, before we start describing our mechanism, we note that one can re-interpret DRF. In the case of a single machine with divisible jobs, notice that the dominant share $s_i = x_{ik^*}/r_{k^*}$ is equal to the ratio of allocated jobs to user i divided by the number of potential jobs for user i if she had the entire machine to herself. This equivalence allows us to connect DRF to the Kalai-Smorodinsky bargaining solution ([Kalai and Smorodinsky 1975]). To see this most clearly, recall that the total number of jobs which can be allocated to user i is sa_i, which in the single machine with divisible jobs is simply $\min_k \frac{r_k}{d_{ik}} = \frac{r_{k^*}}{d_{ik^*}}$. The generalized Kalai-Smorodinski solution, if translated in this

setting, is the weighted max-min allocation of jobs with weight vector $(1/sa_1, \ldots, 1/sa_n)$ over the feasible region. The DRF allocation is exactly this weighted max-min vector of jobs.

One could extend this definition to our container based model; however, if done directly, this fails because the set of feasible allocations $F(d)$ is not convex, a fact crucial to the analysis. To resolve this, we convexify the feasible region to $CH(F(d))$, the convex hull of set $F(d)$, and compute the max-min $\frac{1}{sa_i}-$weighted vector of jobs over $CH(F(d))$. A geometrical interpretation of this solution is this: the weighted max-min is just the intersection of the Kalai-Smorodinski line, the line connecting the origin to the $(sa_1, \ldots, sa_n)$ point, with $CH(F(d))$.

As we will show, this procedure, which we denote Containerized DRF (CDRF), preserves the properties of DRF in this multiple-machine with indivisible jobs setting. In addition, one can directly interpret this mechanism as a randomized mechanism, since the resulting allocation is a convex combination of allocations in $F(d)$, by the definition of a convex hull. Moreover these allocations are Pareto optimal. In pseudocode, CDRF is given as Algorithm 2.

ALGORITHM 2: Containerized-DRF

Input: Demand profile d and cluster r

Output: A feasible allocation $z \in \mathbb{N}^n$.

1: Compute the allocation $CDRF(d, r) = (k_1, k_2, \ldots, k_n) \in \Re^n$ which is the max-min $\frac{1}{sa_i}-$weighted vector of jobs over $CH(F(d))$

2: Compute n allocations $z_1, \ldots, z_n \in F(d)$ such that $CDRF(d, r)$ is their convex combination, i.e. $\exists b_1, \ldots, b_n \in \Re_+$ s.t.

$$\sum_{j=1}^{n} b_j z_j = CDRF(d, r) \text{ and } \sum_{j=1}^{n} b_j = 1$$

3: Output allocation z_j with probability b_j

Example: The feasible region is n dimensional, so, for simplicity, we use the example from Figure 1: two machines $r_1 = (4, 6)$ and $r_2 = (4, 4)$ and users with demands $d_1 = (1, 3)$ and $d_2 = (2, 1)$. See Figure 3. The feasible region is the set of integer points and the colored area is it's convex hull. The set of Pareto optimal points of the feasible region are $\{(3, 1), (2, 2), (1, 3), (0, 4)\}$. The stand alone shares are $sa_1 = 4$ and $sa_2 = 3$. The intersection of the Kalai-Smorodinski line, connecting the origin with the (sa_1, sa_2) point, is the point that maximizes the minimum $\frac{u_i}{sa_i}$ over the colored area. In this example that point is $CDRF(d, r) = (\frac{12}{7}, \frac{16}{7})$. This can be written as a convex combination of $(1, 3)$ and $(2, 2)$ with coefficients $\frac{2}{7}$ and $\frac{5}{7}$. Thus, CDRF will output the allocation $(1, 3)$ with probability $\frac{2}{7}$, and the allocation $(2, 2)$ with probability $\frac{5}{7}$.

Let $CDRF(d, r) \in \Re^n$ be the weighted max-min allocation for demand profile d; the Kalai-Smorodinski solution. Observe that the expected number of containers user i gets is exactly $CDRF_i(d, r)$. If i reports her true demand that $CDRF_i(d, r)$ is also her utility. Since the region we're maximizing over is convex, it immediately follows that Containerized-DRF satisfies several key properties.

THEOREM 3.1. *If users are expected utility maximizers then CDRF satisfies ex-ante sharing incentive, ex-ante Pareto optimality, ex-ante population monotonicity and independence of dummy machines.*

PROOF. Pareto optimality is ex-ante satisfied since the expected allocation is on the convex hull of the feasible region. Sharing incentives is similarly satisfied ex-ante

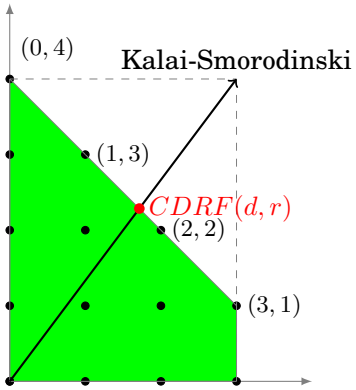

Fig. 3. CDRF in a cluster with $r_1 = (4, 6)$ and $r_2 = (4, 4)$, and users with demands $d_1 = (1, 3)$ and $d_2 = (2, 1)$. $CDRF(d, r)$ is the weighted max-min allocation, i.e. $CDRF(d, r)_i$ is the expected utility of user i in CDRF. In higher dimensions $CDRF(d, r)$ is not necessarily the intersection of $CH(F(d))$ and this "Kalai-Smorodinski" line, as one needs to use the lexicographic max-min.

since the Kalai-Smorodinski solution always dominates the $(\lfloor sa_1/n \rfloor, \ldots, \lfloor sa_n/n \rfloor)$ allocation, and we know from Theorem 2.3 that this allocation is always feasible. Population monotonicity also follows directly since when a user leaves the system, the feasible region can only expand. The independence of dummy machines property holds since our mechanism works directly with the feasible region: if the feasible region is not changed, then the allocation is not changed. □

CDRF is also strategy proof in expectation, although the proof is somewhat more complex.

THEOREM 3.2. *If users are expected utility maximizers then CDRF is ex-ante strategyproof.*

The proof can be found in Appendix A.

4. APPROXIMATE EX-POST MECHANISMS

In certain situations, the randomized mechanism might be directly applicable. For example, if individual job times are short then in a randomized mechanism the fluctuations in users' job shares would average out quickly. However, one potential problem with this mechanism in practice is the possibility of large fluctuations in allocations which may rely too heavily on the assumption that users maximize expected utility. For example, consider the case with $r_j = (1)$ for $1 \leq j \leq m$ and $d_1 = d_2 = (1)$. The randomized CDRF mechanism could choose to randomize between two allocations, the first gives all the machines to user 1 and the second gives all the machines to user 2. Thus, while the average allocation for each user is $m/2$ jobs, the fluctuations are huge.

In the following, we present several mechanisms which help resolve this issue. They produce allocations which not only satisfy the properties in the expectation/ex-ante sense, but also approximately satisfy these properties ex-post, i.e. after the actual allocations are realized. The key idea is that if we can always find an allocation that is close to the CDRF allocation, then the mechanism that produces that allocation will "inherit" approximate ex-post bounds for the properties satisfied by CDRF. We formalize this as follows:

Definition 4.1. A mechanism M is an ϵ-approximate CDRF mechanism if for all demand profiles d and clusters r,

$$\frac{|M_i(d,r) - CDRF_i(d,r)|}{CDRF_i(d,r)} \leq \epsilon$$

for all users i, and M is ex-ante Pareto Optimal.

In order to simplify the notation, in this section we let $M_i(d,r)$ be the number of jobs allocated to user i and require that the mechanism allocates containers of size d_i. Thus the explicit discussion of containers will not be necessary.

It is straightforward to show that such a mechanism approximately satisfies the ex-post versions of our properties, where we define these directly in terms of the fractional changes to each user. For example, ϵ-approximate ex-post sharing incentive implies that the actual allocation

$$M_i(d,r) \geq (1-\epsilon)\lfloor \frac{sa_i(d,r)}{n} \rfloor$$

and similarly for ϵ-approximate ex-post population monotonicity and ϵ-approximate IDM. For ϵ-approximate ex-post strategyproofness we require that no user can gain a fractional increase in jobs of ϵ from a deviation.

THEOREM 4.2. *If mechanism M is an ϵ-approximate CDRF mechanism then it satisfies:*
1) ϵ-approximate ex-post sharing incentives,
2) ex-ante Pareto Optimality,
3) ϵ-approximate ex-post population monotonicity,
4) ϵ-approximate IDM,
5) 3ϵ-approximate ex-post strategyproofness.

PROOF. Parts 1,3 and 4 follow immediately from the definition of "ϵ-approximate ex-post" versions. Part 2 is immediate from the assumption that M is ex-ante Pareto Optimal. Part 5 follows from the strategyproofness of CDRF: Consider a deviation for a single user from d_i to d'_i, with d and d' the respective demand profiles. Assuming that $d_{ik} \leq d'_{ik} < 2d_{ik}$, for all resources k, we can see that

$$\frac{|u_i(M_i(d'), d'_i) - u_i(CDRF_i(d'), d'_i)|}{u_i(CDRF_i(d'), d'_i)} \leq \epsilon$$

as is

$$\frac{|u_i(M_i(d), d_i) - u_i(CDRF_i(d), d_i)|}{u_i(CDRF_i(d), d_i)} \leq \epsilon$$

by the definition of an ϵ-approximate mechanism. Next, by strategyproofness of CDRF, we see that $u_i(CDRF_i(d'), d_i) - u_i(CDRF_i(d), d_i) \leq 0$ and by the assumption on d'_i, $u_i(CDRF_i(d'), d'_i) = u_i(CDRF_i(d'), d_i)$, since user i can only fit one job in a container of size d'_i. Similarly $u_i(M_i(d'), d') = u_i(M_i(d'), d)$. Also, from the definition of ϵ-approximate CDRF we can get $\frac{u_i(M_i(d), d_i)}{1-\epsilon} \geq u_i(CDRF_i(d), d_i)$. Combining these facts we get:

$$\frac{u_i(M_i(d'), d_i) - u_i(M_i(d), d_i)}{u_i(CDRF_i(d), d_i)} \leq 2\epsilon$$

which implies

$$\frac{u_i(M_i(d'), d_i) - u_i(M_i(d), d_i)}{u_i(M_i(d), d_i)} \leq \frac{2\epsilon}{1-\epsilon} < 3\epsilon$$

which is the definition of 3ϵ approximate ex-post strategyproofness. In the case where d'_i does not satisfy our assumption, we use the same techniques as in the proof of theorem 3.2 to complete the analysis. □

We note here that an ϵ-approximate CDRF mechanism doesn't have to be randomized. In case it is deterministic then the properties above are approximately satisfied in the obvious way.

4.1. Identical Machines

While a typical CCC may contain a thousand to a hundred thousand machines, there are typically only a few specific types of machines in the CCC, due to scalability concerns, both for hardware and software. In this setting we can get an ϵ-approximate CDRF mechanism.

For simplicity, we first consider the case when all m machines are identical. In this setting we compute $z = CDRF(d, r_1)$ which is the CDRF expected allocation for a single machine. Since this z is a point in the convex hull of an n-dimensional space, we can then find n feasible allocations $z^t \in F_1(d)$ and their associated α_t's such that $z = \sum_{t=1}^{n} \alpha_t z^t$ where each $\alpha_i \geq 0$ and $\sum_{t=1}^{n} \alpha_t = 1$. Then we randomly assign each machine to allocation z^t with probability α_t. Define $\epsilon = \sqrt{\frac{12n \log n}{m}}$ and check that all users i, are allocated at least $(1-\epsilon)mz_i$ jobs. If this is not true, then repeat the randomized procedure until it is. We denote this mechanism the Identical Machines-CDRF, or just IM-CDRF.

THEOREM 4.3. *Suppose that there are m identical machines. Then IM-CDRF is $O(\sqrt{\frac{n \log n}{m}})$-approximate CDRF. In addition, the number of iterations is less than 2 on average, and more than γ with probability less than $2^{-\gamma}$.*

The proof of this theorem can be found in Appendix B.

We can extend this to the case where there are several classes of identical machines, but computing the IM-CDRF allocation separately for each class to get the GIM-CDRF allocation.

COROLLARY 4.4. *Suppose that there are $t < n^b$ machine types and at least c copies of every machine. Then GIM-CDRF is $O\left(\sqrt{\frac{nb \log n}{c}}\right)$-approximate CDRF. In addition, IM-CDRF requires 2 iterations on average and only requires more than γ iterations with probability less than $2^{-\gamma}$.*

PROOF. The constant in the approximation is increased by a factor of $2^{1/2}$, since in order to guarantee that the tail bounds apply to all classes of machines simultaneously we need to choose $k^2 = (12+b)\log(n)$ in the proof of 4.3. □

4.2. Deterministic mechanisms

One can improve this mechanism by reducing the randomization. For example, instead of choosing the allocation for each machine independently on can directly assign $\lfloor m\alpha_t \rfloor$ machines to allocation z^t and then ignore the remaining machines. This mechanism has the same properties as IM-CDRF but a somewhat different error bound of $\epsilon = O(n^2/m)$, and also is not Pareto Optimal.

Thus, in a common setting, we can directly construct approximate deterministic allocations. In particular, one key property of GIM-CDRF is its simplicity as an algorithm: One only needs to find the CDRF allocation on a single machine of each type and then GIM-CDRF provides a simple algorithm for allocating all the machines. However, for more general sets of machines computing a good allocation is more complex. In the following, we provide a general approximation theorem for arbitrary sets of machines. Our analysis relies on a famous result in convex analysis [Starr 1969]. This result is non-constructive; it shows the existence of a good approximate CDRF deterministic mechanism, but does not provide a reasonable way of computing it. There is a constructive version of this result ([Starr 1981]) but it comes with worse approximation bounds and doesn't solve our key algorithmic issue: computing maximum allocations over the feasible region is NP-hard.

Given a set S_i define the inner radius, $IR(S_i)$ to be the smallest value of ρ such that for any point $y \in CH(S_i)$ the ball of radius ρ centered at y will contain a subset of S_i whose convex hull contains y. For example, in Figure 3 the inner radius of the feasible region is $\sqrt{2}$. The ball with radius $\sqrt{2}$ with center $CDRF(d, r)$ contains $(1, 3)$ and $(2, 2)$, whose convex combination can give $CDRF(d, r)$.

Another example, where we can see that a point on the convex hull can be far from any point in the feasible region is the following: Consider a cluster with $n + 1$ users and $n + 1$ resources. Each user i, $i = 1 \dots n$, demands 1 unit of resource i and 1 unit of resource 0. Player 0 demands n units of resource 0. Assume that this cluster has n machines, where each machine j has n units of resource j and n units of resource 0. For every machine j, $(1, 0, \dots, 0) \in F_j(d)$ and $(0, \dots, 0, n, 0, \dots, 0) \in F_j(d)$ where n is in the j-th position. $\left(\frac{n}{2}, \frac{n}{2}, \dots, \frac{n}{2}\right) \in CH(F(d))$ is the CDRF allocation. The closest integral allocation is $(0, \frac{n}{2}, \dots, \frac{n}{2})$, which is clearly not Pareto optimal, since $(0, n, n, \dots, n)$ is feasible; however in computing the inner radius the latter point is used. Thus the inner radius here is $2n\sqrt{n}$.

THEOREM 4.5 (SHAPLEY-FOLKMAN-STARR). *Let $S_1, ..., S_m$ be a family of m compact subsets of R^n, $W = \sum_{i=1}^{m} S_i$. Then for any $x \in CH(W)$ there is $y \in W$ such that $\|x - y\|_2^2$ is bounded by the sum of squares of the n largest $IR(S_i)$.*

In our setting, the Shapley-Folkman-Starr theorem states that one can approximate any allocation in the convex hull of the feasible region, such as the CDRF allocation, with an actual (integral) feasible allocation.

We are now ready to define the **Shapley-Folkman-Starr** mechanism: First we compute the CDRF allocation $z = CDRF(d, r)$. Remember z_i is the expected number of containers user i gets. Next, from the definition of Minkowski sums and the fact that the convex hull of the sum is equal to the sum of the convex hulls, we can decompose $z = \sum_{j=1}^{m} z_j$ where $z_j \in CH(F_j(d))$. Since z is Pareto optimal, all of the z_j's must also be Pareto optimal and furthermore they must lie on a face of $CH(F_j(d))$. This face is composed of at most n points in $F_j(d)$, which we will denote G_j; G_j is just the set of (Pareto optimal) allocations on machine j that can span z_j. Then, we apply the Shapley-Folkman-Starr theorem to the Minkowski sum of the G_j's to approximate z. We call the resulting allocation $y = SFS\,(d, r)$ the SFS mechanism.

In order to prove a reasonable bound on the SFS mechanism we need to make some assumptions about the heterogeneity of the machines. For example, if one machine is much larger than all the other machines then approximations will be difficult. Define I^* to be the maximum inner radius out of all the $F_j(d)$'s: $I^* = \max_j IR(F_j(d))$. The bound will depend on I^*. Next, since the bounds need to be multiplicative, if a user is allocated a small number of jobs, then even a small additive approximation will be problematic. Thus we make the simple, and reasonable, assumption that each user

can run at least one job per machine, on average. Thus, with m machines each user will have a stand alone bound of $sa_i \geq m$. Under these assumptions we can prove the following:

THEOREM 4.6. *Let I^* be the max inner radius of a feasible region of a machine, and suppose that for each user $sa_i \geq m$. Then SFS $O\left(\frac{n^{\frac{3}{2}}}{m} I^*\right)$-approximates CDRF.*

PROOF. First, we note that by applying the Shapley-Folkman-Starr bound to the G_j's instead of the $F_j(d)$'s directly, we get a Pareto Optimal allocation. Remember that SFS is deterministic, so ex-ante Pareto Optimal is the same as Pareto Optimal. Next, we see from the Shapley-Folkman-Starr bound that $\|SFS(d,r) - CDRF(d,r)\|_2 \leq \sqrt{n} I^*$. This implies that for each user i, $|SFS_i(d,r) - CDRF_i(d,r)| \leq \sqrt{n} I^*$, since $\|\cdot\|_2 \geq \|\cdot\|_\infty$. Thus the fractional difference between the SFS allocation and the CDRF allocation is less than $\frac{\sqrt{n} I^*}{(m/n)} = \frac{n^{\frac{3}{2}} I^*}{m}$. This holds because the fair share of user i is at least $\frac{m}{n}$ by assumption. Thus, we see that the SFS mechanism satisfies the stated approximation ratio. □

5. EXTENDABILITY AND FUTURE WORK

Our generic construction appears to be quite extendable to other types of allocation constraints due to the robustness of our convexification approach. For example, one might have jobs that can only run on a subset of the machines. However, it is less obvious how to extend it to "softer constraints" where an allocation is not impossible, but is less preferred. For example, one important extension is to include locality preferences. Disk-locality ([Hindman et al. 2011]), rack-locality ([Dean and Ghemawat 2001]), and memory-locality ([Ananthanarayanan et al. 2011]) have all been important considerations. For example, it can be efficient to put related jobs on the same machine or nearby machines (same rack or same building) to reduce data transfer delays. The adaptation of our analysis to these and other "soft constraints" appears challenging.

In addition, the construction of efficient algorithms for these mechanisms is an important open problem. Most computational problems involving allocating discrete jobs on a large number of machines are quite difficult and heuristics are widely used in practice. Even simpler subproblems, such as bin packing and knapsack are NP-complete. Nonetheless, under structural assumptions, such as identical machines, we expect that fast algorithms are possible.

Acknowledgments

This research has been supported by grants NSF-1216073, NSF-1161813, and the A.G. Leventis Foundation. We would also like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

REFERENCES

ANANTHANARAYANAN, G., GHODSI, A., SHENKER, S., AND STOICA, I. 2011. Disk-locality in datacenter computing considered irrelevant. In Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems. HotOS'13. USENIX Association, Berkeley, CA, USA, 12–12.

APACHE.ORG. 2014a. YARN DRF extension to the Capacity Scheduler. https://issues.apache.org/jira/browse/YARN-2.

APACHE.ORG. 2014b. YARN DRF extension to the Fair Scheduler. https://issues.apache.org/jira/browse/YARN-326.

BHATTACHARYA, A. A., CULLER, D., FRIEDMAN, E., GHODSI, A., SHENKER, S., AND STOICA, I. 2013. Hierarchical scheduling for diverse datacenter workloads. In

Proceedings of the 4th Annual Symposium on Cloud Computing. SOCC '13. ACM, New York, NY, USA, 4:1–4:15.

DEAN, J. AND GHEMAWAT, S. 2001. Mapreduce: Simplified data processing on large clusters, osdi'04: Sixth symposium on operating system design and implementation, san francisco, ca, december, 2004. S. Dill, R. Kumar, K. McCurley, S. Rajagopalan, D. Sivakumar, ad A. Tomkins, Self-similarity in the Web, Proc VLDB.

DOLEV, D., FEITELSON, D. G., HALPERN, J. Y., KUPFERMAN, R., AND LINIAL, N. 2012. No justified complaints: On fair sharing of multiple resources. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. ITCS '12. ACM, New York, NY, USA, 68–75.

DUBINS, L. E. AND SPANIER, E. H. 1961. How to cut a cake fairly. The American Mathematical Monthly 68, 1, pp. 1–17.

FRIEDMAN, E. J., GHODSI, A., SHENKER, S., AND STOICA, I. 2011. Strategyproofness, leontief economies and the kalai-smorodinsky solution. Manuscript.

GHODSI, A., SEKAR, V., ZAHARIA, M., AND STOICA, I. 2012. Multi-resource fair queueing for packet processing. SIGCOMM Comput. Commun. Rev. 42, 4, 1–12.

GHODSI, A., ZAHARIA, M., HINDMAN, B., KONWINSKI, A., SHENKER, S., AND STOICA, I. 2011. Dominant resource fairness: Fair allocation of multiple resource types. In Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation. NSDI'11. USENIX Association, Berkeley, CA, USA, 24–24.

GHODSI, A., ZAHARIA, M., SHENKER, S., AND STOICA, I. 2013. Choosy: Max-min fair sharing for datacenter jobs with constraints. In Proceedings of the 8th ACM European Conference on Computer Systems. EuroSys '13. ACM, New York, NY, USA, 365–378.

GUTMAN, A. AND NISAN, N. 2012. Fair allocation without trade. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. International Foundation for Autonomous Agents and Multiagent Systems, 719–728.

HINDMAN, B., KONWINSKI, A., ZAHARIA, M., GHODSI, A., JOSEPH, A. D., KATZ, R., SHENKER, S., AND STOICA, I. 2011. Mesos: A platform for fine-grained resource sharing in the data center. In Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation. NSDI'11. USENIX Association, Berkeley, CA, USA, 22–22.

JOE-WONG, C., SEN, S., LAN, T., AND CHIANG, M. 2012. Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework. In INFOCOM. 1206–1214.

KALAI, E. AND SMORODINSKY, M. 1975. Other solutions to nash's bargaining problem. Econometrica 43, 3, pp. 513–518.

LI, J. AND XUE, J. 2013. Egalitarian division under leontief preferences. Economic Theory 54, 3, 597–622.

LINUXCONTAINERS.ORG. 2014. http://linuxcontainers.org/.

PARKES, D. C., PROCACCIA, A. D., AND SHAH, N. 2012. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. In Proceedings of the 13th ACM Conference on Electronic Commerce. EC '12. ACM, New York, NY, USA, 808–825.

STARR, R. M. 1969. Quasi-equilibria in markets with non-convex preferences. Econometrica 37, 1, pp. 25–38.

STARR, R. M. 1981. Approximation of points of the convex hull of a sum of sets by points of the sum: an elementary approach. Journal of Economic Theory 25, 2, 314–317.

WANG, W., LI, B., AND LIANG, B. 2013. Dominant resource fairness in cloud computing systems with heterogeneous servers. CoRR abs/1308.0083.

Appendix A

Proof of Theorem 3.2 :
Let d' be the demand profile where the i-th user reports some d'_i and every other user l truthfully reports d_l, and d the demand profile where everyone is truthful. Also, let $f_i(d'_i, d_i)$ be the number of tasks user i can execute in a container of size d'_i. Remember that $CDRF_i(d', r)$ is the expected number of containers user i gets in profile d'. So, her expected utility can be written $CDRF_i(d', r)\ f_i(d'_i, d_i)$. We have to show that this value is maximum for $d'_i = d_i$.

First observe that by definition reporting a vector d'_i with $d'_{ir} < d_{ir}$ for some resource r gives zero utility, since $f_i(d'_i, d_i)$ is zero. So, we only have to consider deviations d'_i that over demand resources.

Let $p_k = \lfloor \frac{d'_{ik}}{d_{ik}} \rfloor$. If all p_k are not equal, the largest can be reduced without any loss in utility: $f_i(d'_i, d_i)$ will be the same, $CH(F(d))$ and sa_i can only increase, thus $CDRF_i(d', r)$ can only increase. Using the exact same argument we can see that $p_k = \frac{d'_{ik}}{d_{ik}}$, that is d'_i is a better deviation for our mechanism when it is an integer multiple of d_i. So, w.l.o.g. we can assume that all p_k's are equal to some integer p, and that $d'_{ik} = pd_{ik}$.

All that's left to show is that the best such p is $p = 1$. Let's examine what happens when the demand of user i changes from d_i to $d'_i = p\ d_i$: sa_i becomes at least p times smaller, since it is harder to allocate big tasks of size d'_i than it is to allocate tasks of size d_i. For the same reason, $CH(F(d'))$ is a subset of $CH(F(d))$ with the i-th dimension rescaled by p. So, $CDRF_i(d', r) \leq \frac{CDRF_i(d,r)}{p}$, while $f_i(d'_i, d_i) = p$, for any $p \geq 1$. Thus there is no profitable deviation.

□

Appendix B

Proof of Theorem 4.3 : First note that the feasible region of the cluster is the Minkowski sum of the single machine feasible region m times with itself. Since the Minkowski sum of convex hulls is the convex hull of Minkowski sums, if $z = CDRF(d, r_1)$ then $mz = CDRF(d, r)$. Now, let Y_j be random variable for the allocation on the j'th machine which takes on value z^t with probability α_t.

To simplify the presentation, consider the "normalized variables", $W_{ji} = Y_{ji}/sa_i(d, r_1)$ and $v_i = z_i/sa_i(d, r_1)$ and note that both are contained in the interval $[0, 1]$.

First, we note that $E[W_{ji}] = v_i$ by construction, so $E\left[\sum_{j=1}^{m} W_{ji}\right] = mv_i$. Next we note that $Var[W_{ji}] \leq v_i(1 - v_i)$ since the W_{ji} has mean v_i and is contained on the interval $[0, 1]$ and the maximum variance for such a random variable arises when the random variable takes on only the values 0 and 1. This implies that $Var\left[\sum_{j=1}^{m} W_{ji}\right] \leq mv_i(1 - v_i)$ and thus for the standard deviation σ we have $\sigma = \sigma\left[\sum_{j=1}^{m} W_{ji}\right] \leq \sqrt{mv_i(1 - v_i)}$.

Then, using Chernoff's inequality we get that

$$Pr\left[\left|\sum_{j=1}^{m} W_{ji} - mv_i\right| \geq k\sigma\right] \leq 2e^{-k^2/4}.$$

Setting $k^2 = 12 \log n$ gives us

$$Pr\left[\left|\sum_{j=1}^{m} W_{ji} - mv_i\right| \geq k\sigma\right] \leq 2/n^3.$$

We then apply a union bound to obtain

$$Pr\left[\left\{\left|\sum_{j=1}^{m} W_{ji} - mv_i\right| \geq k\sigma\right\} \quad \forall i\right] \leq 2n/n^3 = 2/n^2$$

which is less than $1/2$ for $n \geq 2$.

To complete the proof we note that

$$Pr\left[\left|\sum_{j=1}^{m} W_{ji} - mv_i\right| \geq k\sigma\right] = Pr\left[\left|\frac{\sum_{j=1}^{m} W_{ji} - mv_i}{mv_i}\right| \geq \frac{k\sigma}{mv_i}\right]$$

which is the fractional difference between the number of jobs allocated to user i under IM-CDRF and the expected number allocated under CDRF. Thus, we see that the ϵ in the approximation bound is given by

$$\epsilon \leq \frac{k\sigma}{mv_i} \leq \frac{\sqrt{12 \log n \; mv_i(1 - v_i)}}{mv_i} \leq \frac{\sqrt{12 \; \log n}}{\sqrt{m/n}} = \sqrt{\frac{12 \; n \log n}{m}}$$

which is the desired bound, where we used the fact that $1 - v_i < 1$ and $v_i > 1/n$ by resource sharing.

Since the probability of this bound being exceeded is less than $1/2$, the number of iterations is bounded by a geometric distribution, leading to the stated bound on iterations. Lastly, ex-ante Pareto optimality is immediate, since all the z^t's are Pareto optimal and lie on the face that contains the CDRF allocation. □