Efficient Performance-Centric Bandwidth Allocation with Fairness Tradeoff

Li Chen, Yuan Feng, Baochun Li, *Fellow, IEEE*, and Bo Li, *Fellow, IEEE*

Abstract—Fair bandwidth allocation in datacenter networks has received a substantial amount of research attention, as multiple tenants are hosted by virtual machines in a public cloud. In the context of private datacenters, link bandwidth is shared among applications running data parallel frameworks, such as MapReduce, instead. In this paper, we introduce the rigorous definition of *performance-centric fairness*, with the guiding principle that the performance that data parallel applications will enjoy should be proportional to their weights. We first investigate the problem of maximizing application performance while maintaining strict performance-centric fairness. We then present an inherent tradeoff between fairness and efficiency, which is interpreted from the perspectives of bandwidth utilization and social welfare, respectively. From the first perspective, we propose an algorithm to improve bandwidth utilization by introducing an extended version of fairness. From the second perspective, we formulate an optimization problem of bandwidth allocation that maximizes the social welfare across all the applications, allowing a tunable degree of relaxation on performance-centric fairness. A distributed algorithm is then presented to solve the problem, based on dual based decomposition. With extensive simulations, we demonstrate the effectiveness of our algorithms in improving efficiency and application performance (by up to $1.4X$), with flexible degree of relaxation on the performance-centric fairness.

Index Terms—Datacenter networks, bandwidth allocation, fairness

1 Introduction

Datacenters have become the *de facto* standard computing platform for Web service providers—such as Google and Facebook—to host a wide variety of computationally intensive applications, ranging from PageRank [1] to machine learning [2]. In order to scale up to accomodate the volume of data that these applications need to process, these applications need to embrace *data parallel* frameworks, such as MapReduce [3], Dryad [4] and Spark [5].

In general, data parallel applications typically proceed in several *computation* stages that require *communication* between them. With MapReduce, for example, input data is first partitioned into a set of *splits* [3], so that they can be processed in parallel with *map* computation tasks. The map tasks produce intermediate results, which are then shuffled over the datacenter network to be processed by *reduce* computation tasks.

As multiple data parallel applications share the same private datacenter operated by a Web service provider, we wish to maximize the performance of these applications, measured by their completion times, subject to resource capacity constraints in the datacenter. With respect to resources, the completion time of a data parallel application depends on both computation resources (CPU, memory, *etc.* used in the computation stages) and network resources (link bandwidth used in the communication stages).

It has been shown in Hadoop traces from Facebook that the communication stages usually account for more than 30 percent of the entire completion times for jobs with reduce phases [6]. In the context of a privately operated datacenter shared by such network-intensive applications, which require much more link bandwidth for data transmission than CPU power, how should the critical *link bandwidth* be shared among these applications? It is commonly accepted in the literature that bandwidth should be shared in a *fair* manner (e.g., [7]), yet there has been no general consensus on how the notion of *fairness* should be defined. The traditional wisdom on fair bandwidth sharing has largely focused on datacenters in a public cloud, where virtual machines (VMs) are used to host applications for the tenants. For example, bandwidth on a link can be allocated fairly across different flows, VM pairs, or tenants (according to their payments).

In this paper, we argue that the notions of fairness proposed in the literature are not applicable to the context of data parallel applications sharing a private datacenter. Rather than being fair across competing flows or tenants according to their payments, the thesis of this paper hinges upon the notion of *performance-centric fairness*, in that fairness should be maintained with respect to the *performance* across multiple data parallel applications.

But how, after all, shall we rigorously define the notion of *performance-centric fairness*? As available bandwidth resources are allocated for data parallel applications to transfer

- *L. Chen, Y. Feng, and B. Li are with the Department of Electrical and Computer Engineering, University of Toronto, ON M5S3G4, Canada. E-mail: {lchen, bli}@ece.utoronto.ca, yfeng@eecg.toronto.edu.*
- *B. Li is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. E-mail: bli@cse.ust.hk.*

Manuscript received 17 Apr. 2017; revised 5 Jan. 2018; accepted 6 Feb. 2018. Date of publication 20 Feb. 2018; date of current version 13 July 2018. (Corresponding author: Li Chen.)
Recommended for acceptance by R. Prodan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2018.2808202

data in their communication stages, their performance is best represented by the amount of time needed to complete the data transfer, called the *transfer time*. To achieve their best possible performance with the shortest possible transfer times, the guiding principle of *weighted performance-centric fairness* is that the *reciprocal* of the transfer times should be proportional to their weights across competing applications. To put it simply, applications with equal weights sharing the same private datacenter should enjoy the same performance.

The problem of achieving performance-centric fairness becomes more interesting when we also wish to maximize efficiency, with respect to both the bandwidth utilization and the social welfare. In this paper, we begin with the problem formulation that maximizes the application performance with the constraint that strict performance-centric fairness is to be maintained. Yet, with an example, we will show that there exists an inherent conflict between maximizing bandwidth utilization and maintaining strict fairness, simply because some access links are more heavily loaded than others in a datacenter. To reconcile the conflicting objectives, we introduce an extended version of performance-centric fairness, so that the efficiency, interpreted from the perspective of bandwidth utilization, can be maximized without violating such fairness. From another perspective, the maximal efficiency is achieved if the social welfare, i.e., the total utility across all the applications, is maximized. In this sense, we introduce tunable degrees of relaxation on performance-centric fairness to arbitrate the conflicting objectives, so that the social welfare can be further improved.

The upshot in this paper revolves around the new optimization problem to maximize social welfare while maintaining weighted performance-centric fairness with a certain degree of relaxation. It turns out that this problem is challenging to both formulate and solve. To show the nuances in formulating this problem, consider the link bandwidth to be allocated to a communication stage within an application. The transfer time is determined by the rate of the *slowest* flow between the communicating tasks. To maximize efficiency, we allocate link bandwidth to flows in the same application so that all of them finish at the same time as the slowest flow. Intuitively, we wish to maximize the social welfare in the datacenter with all the applications considered, so that resources are best utilized to improve utilities, with the tradeoff of relaxing fairness to a certain degree.

With a sharp focus on performance-centric fairness in private datacenter networks, in Section 2, we begin our exposition with an illustrating example to establish a convincing case and to provide the formal definition for weighted performance-centric fairness. We then formulate our first optimization problem in Section 3, which derives the maximal possible weighted performance-centric fair share for all the concurrent applications. In Section 4, we illustrate the tradeoff between fairness and bandwidth utilization, and propose an algorithm to resolve such conflicts. Further, in Section 5, we formulate our new problem that better arbitrates the tradeoff between fairness and social welfare. With a detailed analysis on the nature of our optimization problem, we apply dual based decomposition to solve the centralized problem in Section 6, prove that there is no duality gap, and solve the dual problem with a distributed algorithm, based on local measurements and computation at each physical machine. Our performance evaluation in Section 7 has demonstrated that our algorithms following performance-centric fairness largely outperform per-flow fair bandwidth allocation, with respect to both application performance and efficiency.

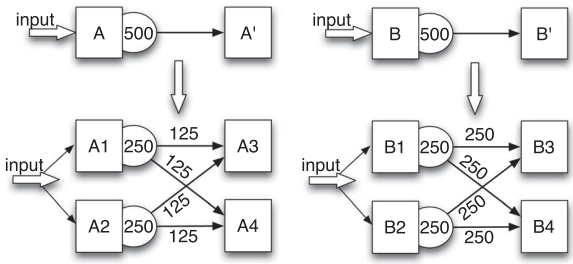


Fig. 1. The amount of data to be transmitted when parallelizing a MapReduce application with the *shuffle* communication pattern, and a machine learning application with the *broadcast* communication pattern.

2 A Case for Performance-Centric Fairness

Data parallel applications partition their input data into multiple splits, which are processed in parallel by computation tasks. As the total amount of computation workload remains the same, a scale-up of n would result in a speed-up of n for the computation stage. To be specific, with n times the computation tasks, the completion time of the computation stage reduces to its $1/n$. However, such a scale-up does not apply to the network transfer time, since the total amount of network traffic may increase with additional parallel tasks, depending on the *communication pattern* between computation tasks.

A typical MapReduce application uses the *shuffle* communication pattern between its map and the reduce tasks, while machine learning applications use a *broadcast* communication pattern [2]. We show an example for both communication patterns in Fig. 1. In the base cases without any parallelization in the computation stages, the only computation task in A (or B) produces 500 MB of intermediate data, which is directly transmitted to the task A' (or B'). In the cases where both applications employ two parallel computation tasks in each computation stage, the input data is then partitioned into two equal splits, and the amount of intermediate data generated by each task is half of the base case. Since A is a MapReduce task with the *shuffle* communication pattern, the data produced by each map task, $A1$ and $A2$, is partitioned into two equal sets, each with a size of 125 MB, to be sent to both $A3$ and $A4$. In contrast, since B uses the *broadcast* communication pattern, each task in the first stage, $B1$ and $B2$, broadcasts all of its produced data to both tasks in the second stage, $B3$ and $B4$.

With the knowledge of the effects of communication patterns on the amount of network traffic, we are now ready to discuss the notion of performance-centric fairness in the context of bandwidth allocation, when A and B share the link bandwidth in a private datacenter as shown in Fig. 2. Specifically, $A1, A2$ co-locate with $B1, B2$ on physical machine $P1$, sharing the egress link bandwidth of $P1$ with a capacity of 500 MB/s. Similarly, $A3, A4$ and $B3, B4$ share the ingress link bandwidth (500 MB/s) of $P2$.

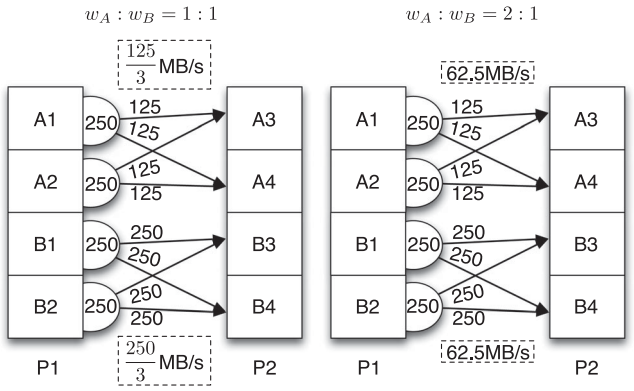


Fig. 2. Examples of bandwidth allocation achieving weighted performance-centric fairness in two cases: 1) both applications have the same weight; 2) the two applications have different weights.

For each application, the transfer time is defined as the completion time of the slowest flow among all flows in its communication stage. To be specific, the transfer time is decided by both the amount of network traffic between each task pair and the bandwidth allocated to each flow. According to their importance, A and B are assigned weights of w_A and w_B, respectively. To satisfy both applications, i.e., to ensure fairness between A and B with respect to their network performance (or transfer time), the egress bandwidth on $P1$ and the ingress bandwidth on $P2$ should be allocated so that the transfer times represented by t_A and t_B satisfy $\frac{1}{t_A} : \frac{1}{t_B} = w_A : w_B$. In this way, the allocation achieves weighted performance-centric fairness for A and B.

To better illustrate this notion, we show two more examples of bandwidth allocation shown in Fig. 2. Since the egress link at $P1$ and the ingress link at $P2$ are both shared by A and B in a symmetric way, we use the term *link bandwidth* for both egress and ingress link bandwidth for simplicity. When both A and B have the same weight, the transfer times of A and B should be equal according to weighted performance-centric fairness. Each flow of A is allocated $\frac{125}{3}$ MB/s, thus the transfer time is $125/\frac{125}{3} = 3$s. With $\frac{250}{3}$ MB/s link bandwidth allocated to each flow, B can achieve a transfer time of $250/\frac{250}{3} = 3$s. Since A and B with the same weight enjoy the same performance with respect to their transfer times, this allocation achieves weighted performance-centric fairness between the two applications.

In the case where A and B have different weights, if we allocate 62.5 MB/s to each flow of both applications as shown in Fig. 2, the transfer time of A is $\frac{125}{62.5} = 2$s, and the transfer time of B is $\frac{250}{62.5} = 4$s. Since $\frac{1}{2} : \frac{1}{4} = w_A : w_B = 2$, weighted performance-centric fairness is again achieved with this allocation.

We argue that weighted performance-centric fairness best meets the requirements of application performance, in the privately operated datacenters. The major reason is that it directly targets the eventual application performance, rather than the amount of resources allocated to each application as previous notions of fairness. Particularly, if two equally important applications, one with the shuffle pattern and the other with the broadcast pattern, scale up by doubling their parallel computation tasks, they should expect the same degree of performance improvement, regardless of their communication patterns. Such an intuitive performance requirement is achieved with our performance-centric fairness, as demonstrated in Fig. 2. In contrast, with traditional fairness, for example, the per-flow fairness, the bandwidth allocated to each flow remains the same, as the application scale-up does not impact the number of sharing flows. However, due to different communication patterns, the amounts data transferred by flows from the two applications become different. Hence, per-flow fairness fails to guarantee the corresponding degree of performance improvement for these applications.

Defining Weighted Performance-Centric Fairness. With an intuitive idea of weighted performance-centric fairness in our illustrative examples, we now present a strict definition in a general setting.

In a privately operated datacenter, multiple data parallel applications share the bandwidth resource by co-locating some of their tasks on some of the physical machines. Each application $k \in \mathcal{K} = \{1, 2, \ldots, K\}$ is assigned the weight w_k according to its importance. If for any application k, its performance, defined as the reciprocal of its transfer time t_k achieved under a certain allocation, satisfies the following condition:

$$(1/t_{k1}) : (1/t_{k2}) = w_{k1} : w_{k2}, \quad \forall k1, k2 \in \mathcal{K}, \tag{1}$$

then weighted performance-centric fairness has been achieved.

Weighted performance-centric fairness is defined with respect to the performance achieved by all applications, rather than the amount of bandwidth resource obtained by each flow or each task. In this sense, this fairness is defined at the level of applications, which is quite different from fairness definitions at the flow level (TCP), VM source level (e.g., [8]), VM-pair level (e.g., [7]) or the tenant level (e.g., [9]) proposed in the literature in the context of datacenters in a public cloud. To achieve such a fairness, the allocation should be aware of the applications' communication patterns, which will affect the amount of network traffic in each flow, and further impact the transfer times of applications.

3 Allocating Bandwidth to Achieve Performance-Centric Fairness

Given the intuitive examples and the strict definition of weighted performance-centric fairness in the previous section, we now study the bandwidth allocation problem with the fairness requirement in a general scenario.

We consider a private datacenter where there are K data parallel applications running concurrently, with their tasks distributed across N physical machines. These applications typically partition the computation among multiple tasks, and communicate the intermediate data between the tasks belonging to different computation stages. The communication pattern can be either *shuffle* as in MapReduce [3], or *broadcast* as in machine learning applications [2].

On each physical machine (or server interchangeably) $n \in \mathcal{N} = \{1, 2, \ldots, N\}$, tasks from different applications will share its link bandwidth, including both the egress link with capacity B_n^E and the ingress link with capacity B_n^I. Since the bisection bandwidth in datacenter networks has been significantly improved by multi-path routing (i.e.,

[10]) and multi-tree topologies (i.e., [11]), we assume a full bisection bandwidth network, where bandwidth is only bottlenecked at the access links of physical machines (which is also assumed in recent works [12], [13], *etc*.). Hence, the completion time of each flow is determined by the bandwidth allocated at the access links. Note that even if the assumption does not hold, our model still works with a minor change, by setting proper bandwidth capacities of physical machines.

Each application $k \in \mathcal{K} = \{1, 2, \ldots, K\}$ requires m_k tasks, represented by $\mathcal{T}_k = \{1, 2, \ldots, m_k\}$. The ith task of application k is represented by $\mathcal{T}_k^i \in \mathcal{T}_k$. For simplicity, we assume that both of the computation stages consist of the same number (i.e., $m_k/2$) of tasks. Given the type of the communication pattern and the number of tasks in each computation stage, we can obtain the network load matrix $\mathcal{D}_k$, where the (i, j)th component $\mathcal{D}_k^{i,j}$ represents the amount of data to be sent by the flow between task $\mathcal{T}_k^i$ and $\mathcal{T}_k^j$. For example, if the total amount of intermediate data generated by application k is d_k, an application with the shuffle pattern will have $\frac{d_k}{(m_k/2)^2}$ data to be sent between each task pair, while an application with the broadcast pattern will have $\frac{d_k}{m_k/2}$ data to be sent by each flow.

Let $r_k^{i,j}$ denote the bandwidth allocated to the (i, j) communicating task pair of application k, then the completion time of the flow between the (i, j) task pair is $\frac{\mathcal{D}_k^{i,j}}{r_k^{i,j}}$. The transfer time of an application is defined as the completion time of the slowest flow in the communication stage, which can be represented as $t_k = \max_{i,j,\mathcal{D}_k^{i,j} \neq 0} \frac{\mathcal{D}_k^{i,j}}{r_k^{i,j}}$ for application k.

As mentioned in Section 2, each application k is associated with a weight w_k. The performance of application k is expressed as $\frac{1}{t_k}$, which indicates that the shorter the transfer time, the better the performance. The fairness definition in Eq. (1) has the following equivalent form:

$$\frac{1}{t_k} = \frac{w_k}{\sum_k w_k} S, \quad \forall k \in \mathcal{K}, \tag{2}$$

where S is a positive variable called the *total performance-centric share*, which is upper bounded given the fixed amount of bandwidth capacity in the datacenter. Our objective is to fairly allocate bandwidth to achieve this upper bound, so that the performance achieved by each application is maximized. Substituting $t_k = \max_{i,j,\mathcal{D}_k^{i,j} \neq 0} \frac{\mathcal{D}_k^{i,j}}{r_k^{i,j}}$ yields

$$\min_{i,j,\mathcal{D}_k^{i,j} \neq 0} \frac{r_k^{i,j}}{\mathcal{D}_k^{i,j}} = \frac{w_k}{\sum_k w_k} S.$$

Now we consider the link bandwidth capacity constraints on each server. Let the binary variable $X_{k,n}^i$ denote whether task i of application k is placed on server n, i.e.,

$$X_{k,n}^i = \begin{cases} 1, \text{ when } \mathcal{T}_k^i \text{ is placed on server } n \\ 0, \text{ otherwise} \end{cases}$$

The total egress rate of each task $\mathcal{T}_k^i$ placed on server n is $\sum_{j, X_{k,n}^j = 0} r_k^{i,j} \cdot X_{k,n}^i$. Note that if the task $\mathcal{T}_k^j$ receiving the intermediate data from $\mathcal{T}_k^i$ is also placed on server n, there will be no data sent through the network. Thus, we add the constraint of $X_{k,n}^j = 0$ in the summation. Summing over all tasks of an application placed on server n, and further summing over all the applications, we obtain the total egress rate of server n, which should not exceed the egress link capacity:

$$\sum_k \sum_i \sum_{j, X_{k,n}^j = 0} r_k^{i,j} \cdot X_{k,n}^i \leq B_n^E$$

The same analysis applies to the ingress link of each server.

We are now ready to formulate the problem of maximizing performance while maintaining weighted performance-centric fairness:

$$\max_{\mathbf{r}} \quad S \tag{3}$$

$$\text{s.t.} \quad \min_{i,j,\mathcal{D}_k^{i,j} \neq 0} \frac{r_k^{i,j}}{\mathcal{D}_k^{i,j}} = \frac{w_k}{\sum_k w_k} S, \quad \forall k \in \mathcal{K} \tag{4}$$

$$\sum_k \sum_i \sum_{j, X_{k,n}^j = 0} r_k^{i,j} \cdot X_{k,n}^i \leq B_n^E, \quad \forall n \in \mathcal{N} \tag{5}$$

$$\sum_k \sum_j \sum_{i, X_{k,n}^i = 0} r_k^{i,j} \cdot X_{k,n}^j \leq B_n^I, \quad \forall n \in \mathcal{N}, \tag{6}$$

where constraint (4) represents weighted performance-centric fairness, while constraints (5) and (6) correspond to the egress and ingress link capacity constraints at each machine.

Let α_k denote the performance of application k, i.e., the reciprocal of its transfer time:

$$\alpha_k = \frac{1}{t_k} = \min_{i,j,\mathcal{D}_k^{i,j} \neq 0} \frac{r_k^{i,j}}{\mathcal{D}_k^{i,j}}. \tag{7}$$

We can obtain the optimal value S^* of the optimization problem (3), (4), (5), and (6) by solving the following optimization problem, which has the same value of S^*:

$$\max_{\boldsymbol{\alpha}} \quad S \tag{8}$$

$$\text{s.t.} \quad \alpha_k = \frac{w_k}{\sum_k w_k} S, \quad \forall k \in \mathcal{K} \tag{9}$$

$$\alpha_k = \frac{r_k^{i,j}}{\mathcal{D}_k^{i,j}}, \quad \forall i, j \in \mathcal{T}_k, \mathcal{D}_k^{i,j} \neq 0 \tag{10}$$

$$\sum_k \alpha_k \sum_i \sum_{j, X_{k,n}^j = 0} \mathcal{D}_k^{i,j} X_{k,n}^i \leq B_n^E, \forall n \in \mathcal{N} \tag{11}$$

$$\sum_k \alpha_k \sum_j \sum_{i, X_{k,n}^i = 0} \mathcal{D}_k^{i,j} X_{k,n}^j \leq B_n^I, \forall n \in \mathcal{N}. \tag{12}$$

The intuition is that since the performance of each application is determined by the completion time of its slowest flow, it is efficient to make all the flows of an application finish at the same time, by allocating flows the amounts of bandwidth that have the same proportionality to their network load. In this way, no bandwidth is wasted in making some of the flows finish faster. Therefore, we can add constraint (10) without impacting the optimal S^* of problem (3), (4), (5), and (6).

Replacing the variables of α_k with S according to constraint (9), we transform problem (8), (9), (10), (11), and (12) as follows:

$$\begin{aligned} \max \quad & S \\ \text{s.t.} \quad & S \cdot \textstyle\sum_k w_k' b_{k,n}^E \le B_n^E, \quad \forall n \in \mathcal{N} \\ & S \cdot \textstyle\sum_k w_k' b_{k,n}^I \le B_n^I, \quad \forall n \in \mathcal{N}, \end{aligned}$$

where $w_k' = \frac{w_k}{\sum_k w_k}$ represents the normalized weight of application k, and

$$b_{k,n}^E = \textstyle\sum_i \sum_{j, X_{k,n}^j = 0} \mathcal{D}_k^{i,j} X_{k,n}^i \tag{13}$$

$$b_{k,n}^I = \textstyle\sum_j \sum_{i, X_{k,n}^i = 0} \mathcal{D}_k^{i,j} X_{k,n}^j, \tag{14}$$

representing the total amount of traffic generated by k to be transmitted through the egress link at server n, and the total amount of data received by k through the ingress link at server n, respectively.

The optimal solution is thus expressed as:

$$S^* = \min \left\{ \begin{array}{l} \min_{n, \sum_k b_{k,n}^E \neq 0} \dfrac{B_n^E}{\sum_k w_k' b_{k,n}^E} \\ \min_{n, \sum_k b_{k,n}^I \neq 0} \dfrac{B_n^I}{\sum_k w_k' b_{k,n}^I} \end{array} \right\} \tag{15}$$

With the maximum total performance-centric share S^*, the optimal performance of each application is obtained as $\alpha_k^* = \frac{w_k}{\sum_k w_k} S^*$.

Finally, according to constraint (10), we derive the optimal rate allocation as:

$$r_k^{i,j*} = \frac{w_k}{\sum_k w_k} S^* \cdot \mathcal{D}_k^{i,j}$$

Note that in our original problem (3), (7), (6), according to Eq. (7), constraint (4) can be transformed as:

$$\begin{aligned} & \alpha_k = \frac{w_k}{\sum_k w_k} S, \quad \forall k \in \mathcal{K} \\ & \alpha_k \le \frac{r_k^{i,j}}{\mathcal{D}_k^{i,j}}, \quad \forall i, j \in \mathcal{T}_k, \mathcal{D}_k^{i,j} \neq 0, \end{aligned} \tag{16}$$

where Eq. (16) indicates the flexibility to increase some of the $r_k^{i,j}$ as long as the capacity constraint is still maintained. However, this would not result in any improvement of the performance α_k. Since the application performance is our main concern, we simply allocate the minimum amounts to achieve the specified performance, according to constraint (10).

4 Reconciling Fairness and Bandwidth Utilization

It is well known that tradeoff exists between fairness and efficiency. In this and the following sections, we will investigate how the bandwidth allocation following the strict definition of weighted performance-centric fairness results in a lack of efficiency, with interpretations from two perspectives. In correspondence, we will present two approaches to improve the efficiency.

The first interpretation of efficiency is from the perspective of bandwidth utilization. When the weighted performance-centric fairness is achieved, the residual bandwidth can be categorized into two classes:

- *useful bandwidth*—the link bandwidth that once allocated to an application, the performance of the application can be improved. It is a concept that is relative to applications, i.e., the useful bandwidth to an application is not necessarily the useful bandwidth to another. For an application k, the useful bandwidth represents a set of available (non-zero) bandwidth on all the access links that k's flows traverse.
- *useless bandwidth*—the link bandwidth that can not improve the performance of any application, *i.e.*, the link bandwidth that is not the *useful bandwidth* to any application.

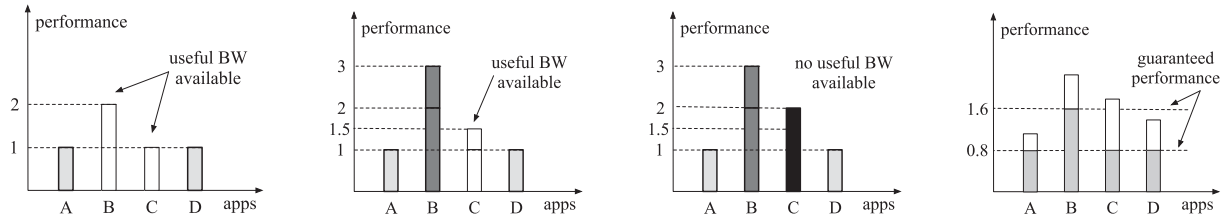
The maximal efficiency is achieved when all the residual bandwidth is *useless bandwidth*.

It is intuitive that when the strict performance-centric fairness is enforced, maximal efficiency may not be achieved, because some applications are not allowed to utilize useful bandwidth for the purpose of maintaining performance proportionality. To eliminate such an inherent conflict, we define an extended version of weighted performance-centric fairness, which allows the residual bandwidth to be iteratively allocated to the applications that can improve their performance with the allocation, according to the performance proportionality regulated by the original weighted performance-centric fairness, until all the residual bandwidth is the *useless bandwidth*. In this way, all the *useful bandwidth* (relative to some applications) will be utilized, and the efficiency regarding bandwidth utilization is maximized.

To better understand the idea, we present an illustrative example of the iterative allocation in Fig. 3. Suppose that according to the weighted performance-centric fairness as previously defined, the maximal total performance-centric share S^* is calculated as 5, so that application A, C and D with the same weight of 1 obtains the performance of 1, and application B with a weight of 2 achieves the performance of 2. However, there is still *useful bandwidth* to B and C that should be utilized to improve efficiency. Hence, in the first round, we simply fix the allocation of A and D to whom there is no longer *useful bandwidth*, as shown in Fig. 3a.

Then we remove A and D from our consideration, and allocate the available bandwidth among B and C in the second round, according to the weighted performance-centric fairness. Still, we compute S^* for the problem where B and C are sharing the available bandwidth, based on which we can derive their performance as 3 and 1.5, proportional to their weights. As shown in Fig. 3b, we fix the allocation of B in this round, while leaving the allocation of C for the next round, since there is still *useful bandwidth* to C. Finally, C will be allocated all the available *useful bandwidth* to it, to achieve a performance of 2, as shown in Fig. 3c. When the allocation completes, no *useful bandwidth* is left idle and the efficiency is maximized.

This allocation process can also be interpreted with progressive filling [14] that is used to achieve max-min fairness in traditional link bandwidth allocation. (Note that progressive filling is mentioned for the ease of illustrating the general idea. Though sharing the similar philosophy, our allocation algorithm to be elaborated later is carefully designed for the more complex context of the performance-centric fairness.) Starting with all the performance equal to 0, we increase the amounts of bandwidth to all the applications so



(a) 1st round: A and D no longer have *useful bandwidth*, and their allocations have been fixed.

(b) 2nd round: B no longer has *useful bandwidth*, and its allocation has been fixed.

(c) 3rd round: C no longer has *useful bandwidth*, and its allocation has been fixed.

(d) Guaranteed performance and add-on performance achieved by A, B, C and D.

Fig. 3. An illustration of allocating bandwidth among applications A, B, C and D iteratively to achieve maximal efficiency, following the extended version of weighted performance-centric fairness. The weights of these four applications are 1, 2, 1, 1, respectively.

that their performance increases at the rate proportional to their weights. When an application no longer has any *useful bandwidth*, its allocation is fixed. Then we continue increasing bandwidth for the remaining applications in a similar way until no *useful bandwidth* is available to any application.

Algorithm 1. Bandwidth Allocation to Achieve Relaxed Performance-Centric Fairness and Maximal Utilization

Input:

Bandwidth capacity: $B_n^E, B_n^I, \ \forall n \in \mathcal{N}$;

Network load matrix $\mathcal{D}_{i,j}^k$ and weight $w_k, \ \forall k \in \mathcal{K}$;

Task placement across physical machines: $X_{k,n}^i$;

Output:

Bandwidth allocation for all applications: $r_{i,j}^k$;

1: Initialize $\mathcal{N}^E = \mathcal{N}, \mathcal{N}^I = \mathcal{N}$;
2: **while** $\mathcal{K} \neq \varnothing$ **do**
3: Calculate $w_k^{'} = w_k / \sum_{k \in \mathcal{K}} w_k$;
4: Calculate S^* according to Eq. (15);
5: Obtain the application set $\mathcal{A}$, where each application has at least one saturated flow;
6: Allocate bandwidth to all applications in $\mathcal{A}$ as:

$$r_k^{i,j} = w_k^{'} S^* \mathcal{D}_k^{i,j};$$

7: Update residual link bandwidth B_n^E and B_n^I;
8: Obtain saturated link sets $\mathcal{M}^E$ and $\mathcal{M}^I$;
9: Update the link sets as: $\mathcal{N}^E = \mathcal{N}^E - \mathcal{M}^E; \mathcal{N}^I = \mathcal{N}^I - \mathcal{M}^I$;
10: Update the application set as: $\mathcal{K} = \mathcal{K} - \mathcal{A}$;

From this perspective, we can see that in Fig. 3, the four applications increases their performance with the ratio of their increasing rates as $1:2:1:1$, same to their weight proportionality. When the performance of A increases to 1, A and D no longer have *useful bandwidth*, so that their performance is fixed as 1 in Fig. 3a. Then we continue increasing the allocation to B and C, until B exhausts all the *useful bandwidth*. Since their performance increasing rates are proportional to their weights, C achieves the performance of 1.5 when B is fixed at the performance of 3, as shown in Fig. 3b. Finally, we increase the allocation of the last application C until all the *useful bandwidth* is used and C achieves the performance of 2, as illustrated in Fig. 3c.

Our iterative allocation approach is summarized in Algorithm 1. We first compute the normalized weight of each application (`Line 3`) and the maximal total performance-centric fair share (`Line 4`) according Eq. (15) as: $S^* = \min\{\min_{n \in \mathcal{N}^E, \sum_{k \in \mathcal{K}} b_{k,n}^E \neq 0} \frac{B_n^E}{\sum_{k \in \mathcal{K}} w_k^{'} b_{k,n}^E}, \min_{n \in \mathcal{N}^I, \sum_{k \in \mathcal{K}} b_{k,n}^I \neq 0} \frac{B_n^I}{\sum_{k \in \mathcal{K}} w_k^{'} b_{k,n}^I}\}$, which is the analytic solution to the optimization problem (8), (9), (10), (11), and (12) over the application set $\mathcal{K}$, the egress link set $\mathcal{N}^E$ and the ingress link set $\mathcal{N}^I$. Then we find all the applications that have at least one of their flows traversing a saturated link (`Line 5`), represented by the set $\mathcal{A} = \{k \in \mathcal{K} \mid \sum_k w_k^{'} b_{k,n}^E = B_n^E || \sum_k w_k^{'} b_{k,n}^I = B_n^I\}$, which also implies that there is no *useful bandwidth* to them. (Note that for such an application, there may be some bandwidth available for its flows that traverse non-saturated links. However, there is no *useful bandwidth* to this application, since it can not improve its performance, as long as one of its flows does not obtain any available bandwidth in a saturated link.) These applications in the set $\mathcal{A}$ are allocated with the amounts of bandwidth in `Line 6` to achieve their respective fair share of performance. The residual bandwidth in each link is then updated in `Line 7` by subtracting the amount that has been allocated to applications in $\mathcal{A}$, represented as follows:

$$B_n^E = B_n^E - S^* \sum_{k \in \mathcal{A}} w_k^{'} b_{k,n}^E, \ \forall n \in \mathcal{N}^E$$
$$B_n^I = B_n^I - S^* \sum_{k \in \mathcal{A}} w_k^{'} b_{k,n}^I, \ \forall n \in \mathcal{N}^I;$$

Finally, we obtain the saturated egress and ingress link sets as $\mathcal{M}^E = \{n \in \mathcal{N}^E \mid B_n^E = 0\}$, $\mathcal{M}^I = \{n \in \mathcal{N}^I \mid B_n^I = 0\}$, remove them from the link sets $\mathcal{N}^E$ and $\mathcal{N}^I$ (`Lines 8-9`), and remove all the applications in $\mathcal{A}$ from the application set $\mathcal{K}$ (`Line 10`).

As long as there are still applications that have not yet been allocated, i.e., $\mathcal{K} \neq \varnothing$, we continue with a new round of allocation following the weighted performance-centric fairness, by solving the problem (8), (9), (10), (11), and (12) over a reduced set of sharing applications and links. When the algorithm terminates, all the *useful bandwidth* have been utilized so that the maximal efficiency is achieved. Meanwhile, in each round, the weighted performance-centric fairness is achieved among the applications in $\mathcal{A}$, thus the extended version of weighted performance-centric fairness is satisfied by Algorithm 1.

Now we analyze the complexity of our Algorithm 1. In each iteration, `Line 4` calculates S^* by looping through the application set $\mathcal{K}$ and the server set $\mathcal{N}$. Hence, the complexity of this step is $O(NK)$, where N is the number of servers (where at least a flow is originated or destined) and K is the number of concurrent applications. `Line 6` loops through all the flows of applications in $\mathcal{A}$ with a complexity of $O(F)$, where F is the total number of concurrent flows across all the applications. The complexity of `Line 7` is $O(NK)$ and other steps are $O(N)$ or $O(K)$. Therefore, each iteration has

the complexity of O(NK+F). With at most K iterations, the time complexity of Algorithm 1 is $O((NK+F)K)$. As we observe, the complexity is linearly or at most quadratically increasing with an input, which is feasible and practical for bandwidth allocation in modern datacenters.

5 Trading Fairness for Social Welfare

Another interpretation of efficiency is from the perspective of the social welfare. Each application k has a utility function, determined by its performance α_k. For simplicity, we choose the $\log$ function of performance as the utility function for each application:

$$U_k(\alpha_k) = \log \alpha_k, \ \forall k \in \mathcal{K},$$

The *social welfare* is the total utility achieved by all the applications, i.e., $\sum_{k\in\mathcal{K}} U_k(\alpha_k)$. The maximal efficiency is achieved when the *social welfare* across all the applications is maximized.

For this interpretation, there is still conflicts between the requirements of weighted performance-centric fairness and maximal social welfare. The underlying reason is twofold, which will be illustrated by two simple examples as follows, where application A and B are sharing the same set of links:

- To achieve 1 unit of performance, A requires 1 unit of bandwidth, while B requires 2 units. A and B have the same weights. According to the allocation that satisfies the weighted performance-centric fairness, A and B will achieve the same performance of 2 units, and the social welfare is $\log 4$. If we reduce 1 unit of bandwidth from B, and reallocate it to A, then the performance achieved by A and B is 3 and 1.5, respectively. With this allocation, the social welfare is computed as $\log 4.5$, greater than that achieved by weighted performance-centric allocation.
- To achieve 1 unit of performance, A and B both require 1 unit of bandwidth. The weights of A and B are 1 and 2, respectively. According to the allocation that satisfies the weighted performance-centric fairness, A will achieve the performance of 2 units, while B achieves 4 units. The social welfare is $\log 8$. If we reduce 1 unit of bandwidth from B, and reallocate it to A, then both A and B will achieve the performance of 3. With this allocation, the social welfare is computed as $\log 9$, greater than that achieved by weighted performance-centric allocation.

To strive for a balance between the conflicting requirements of fairness and efficiency, we set the principle of $\alpha_k \geq w_k^{'} S$ for bandwidth allocation, where $S \in (0, S^*]$ is considered as the degree of fairness relaxation that can be tuned to trade fairness for efficiency. This constraint ensures that, at a minimum, each application can be guaranteed a weighted fair share $w_k^{'} S$ with respect to its performance. A larger S indicates a larger weighted fair share that each application will be guaranteed, while a smaller S represents more relaxation on fairness.

As shown in Fig. 3d, the shaded areas represent the *guaranteed performance* of the four applications, which requires a guaranteed amount of bandwidth. The residual bandwidth, called the *flexible bandwidth*, will be allocated to maximize the social welfare without the concern of fairness, which results in the *add-on performance* represented by the white areas. In this example, S is 4, so that the *guaranteed performance* of A is $4 \cdot \frac{1}{5} = 0.8$, and that of B is $4 \cdot \frac{2}{5} = 1.6$. If we decrease S, there will be more relaxation on fairness, as the *guaranteed performance* will be reduced. However, more *flexible bandwidth* will be available to be allocated for the purpose of improving the social welfare.

To maximize the overall social welfare of all applications with a certain degree of relaxation on performance-centric fairness, we formulate the following optimization problem:

$$\max_{\alpha} \quad \sum_k U_k(\alpha_k) \tag{17}$$

$$\text{s.t.} \quad \alpha_k \geq w_k^{'} S, \ \ \forall k \in \mathcal{K} \tag{18}$$

$$\sum_k \alpha_k \cdot b_{k,n}^E \leq B_n^E, \ \ \forall n \in \mathcal{N} \tag{19}$$

$$\sum_k \alpha_k \cdot b_{k,n}^I \leq B_n^I, \ \ \forall n \in \mathcal{N} \tag{20}$$

where $b_{k,n}^E$ and $b_{k,n}^I$ are expressed in Eqs. (13) and (14). Constraint (18) follows the aforementioned principle, while (19) and (20) are the capacity constraints of the egress and ingress link at each server.

Changing $\max \sum_k U_k(\cdot)$ to $\min - \sum_k U_k(\cdot)$, we can transform the previous optimization problem (17)-(20) to the following:

$$\min_{\alpha} \quad -\sum_k U_k(\alpha_k) \tag{21}$$

$$\text{s.t.} \quad Eq.\ (18), (19), (20) \tag{22}$$

Since $U_k(\alpha_k) = \log \alpha_k$ is strictly concave [15], the objective function of problem (21) and (22) is strictly convex. Moreover, all of the constraints are affine. Hence, problem (21) and (22) is a convex optimization problem [15].

Let $\lambda_k, \forall k \in \mathcal{K}$ denote the Lagrange multipliers associated with constraint (18), and $\mu_n^E, \mu_n^I, \forall n \in \mathcal{N}$ associated with capacity constraints (19) and (20) respectively. The Lagrangian of problem (21)-(22) is as follows:

$$\begin{aligned}
&\mathcal{L}(\alpha, \lambda, \mu^E, \mu^I) \\
&\quad = -\sum_k \log \alpha_k + \sum_k \lambda_k (w_k^{'} S - \alpha_k) \\
&\qquad + \sum_n \mu_n^E (\sum_k \alpha_k b_{k,n}^E - B_n^E) + \sum_n \mu_n^I (\sum_k \alpha_k b_{k,n}^I - B_n^I)
\end{aligned}$$

It is obvious that there exists $\alpha = (\alpha_1, \ldots, \alpha_K)$ in the relative interior of the intersection of domains of all constraint functions, i.e., $\alpha = (\alpha_1, \ldots, \alpha_K)$ satisfies the constraints: $\alpha_k > w_k^{'} S, \forall k \in \mathcal{K}$, $\sum_k \alpha_k b_{k,n}^E < B_n^E, \forall n \in \mathcal{N}$ and $\sum_k \alpha_k b_{k,n}^I < B_n^I, \forall n \in \mathcal{N}$. Hence, Slater's condition [15] is satisfied. And since the optimization problem (21) and (22) is differentiable and convex, the Karush-Kuhn-Tucker (KKT) conditions [15] are both sufficient and necessary for the optimality. Thus, we can derive the optimal solution by applying the KKT conditions:

$$\nabla_{\alpha_k} \mathcal{L}(\alpha, \lambda, \mu^E, \mu^I) = 0, \ \forall k \in \mathcal{K} \Longleftrightarrow$$

$$-1/\alpha_k - \lambda_k + \sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I = 0, \ \forall k \in \mathcal{K}$$

and

$$\begin{cases} \lambda_k(w_k^{'}S-\alpha_k)=0,\ \forall k\in\mathcal{K} \\ \mu_n^E(\sum_k \alpha_k b_{k,n}^E - B_n^E)=0,\ \forall n\in\mathcal{N} \\ \mu_n^I(\sum_k \alpha_k b_{k,n}^I - B_n^I)=0,\ \forall n\in\mathcal{N} \\ \lambda_k\geq 0,\ \forall k\in\mathcal{K} \\ \mu_n^E\geq 0,\ \mu_n^I\geq 0,\ \forall n\in\mathcal{N} \end{cases}$$

Analyzing the solution $(\alpha_k^*, \forall k\in\mathcal{K})$ of the equations above, we derive the following insights:

1) If $\alpha_k^* = w_k^{'}S$ for some k, we have

$$-\tfrac{1}{w_k^{'}S} - \lambda_k + \textstyle\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I = 0. \tag{23}$$

Since $\lambda_k\geq 0$ and $\frac{1}{w_k^{'}S}>0$, to satisfy Eq. (23) we have $\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I > 0$. Hence, there exists a server $n\in\{n|X_{k,n}^i\neq 0, \exists i\in\mathcal{T}^k\}$ that satisfies $\mu_n^E\neq 0$ or $\mu_n^I\neq 0$, from which we have

$$\textstyle\sum_k \alpha_k^* b_{k,n}^E - B_n^E = 0 \ \text{ or } \ \sum_k \alpha_k^* b_{k,n}^I - B_n^I = 0.$$

This indicates that among all the servers hosting application k's tasks, there is at least one of them that has no idle link to improve k's performance. Thus, the performance of k only achieves its minimum guaranteed share.

2) If $\alpha_k^* > w_k^{'}S$, then we have $\lambda_k = 0$, and

$$-\tfrac{1}{\alpha_k^*} + \textstyle\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I = 0.$$

Similar with the analysis when $\alpha_k^* = w_k^{'}S$, there exists such $n\in\{n|X_{k,n}^i\neq 0, \exists i\in\mathcal{T}^k\}$ that satisfies $\sum_k \alpha_k^* b_{k,n}^E - B_n^E = 0$ or $\sum_k \alpha_k^* b_{k,n}^i - B_n^I = 0$, which indicates that the bandwidth is bottlenecked at some servers where tasks of application k are placed.

In this case, we can represent α_k^* as follows:

$$\alpha_k^* = \frac{1}{\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I}. \tag{24}$$

The optimal solution can be obtained by solving the equations of KKT conditions in a centralized manner. The centralized solver needs to maintain the network load matrices $\mathcal{D}_k$ with the dimension of m_k for each application k, as well as the task placement state matrices $X_{k,n}^i$ $(i\in\mathcal{T}^k)$ for the entire datacenter. With $K+2N$ constraints in the optimization problem, the computational complexity and storage overhead will increase significantly as the number of applications and physical machines scales up.

To overcome the limit of the centralized approach, we propose a distributed algorithm to allocate bandwidth at each server with less computation and less state information, which will be presented in the next section.

6 Distributed Bandwidth Allocation for Maximizing Social Welfare

In this section, we first prove that there is no duality gap between the dual and the primal problem, and then apply the dual based decomposition to design a distributed algorithm for bandwidth allocation that can be implemented at each physical machine in parallel.

6.1 Dual Based Decomposition

Let p^* represent the optimal value of the primal optimization problem (21) and (22). Considering the general situation that $\alpha_k > w_k^{'}S$, we use $\mathcal{X}\subset\mathcal{R}^K$ to denote the solution space defined by constraint (18). The Lagrangian of the primal problem under the general case is defined as $\mathcal{L}(\cdot):\mathcal{X}\times\mathcal{R}^N\times\mathcal{R}^N\rightarrow\mathcal{R}$.

$$\begin{aligned}\mathcal{L}(\alpha,\mu^E,\mu^I) = &-\textstyle\sum_k \log\alpha_k + \sum_n \mu_n^E(\sum_k \alpha_k b_{k,n}^E - B_n^E) \\ &+ \textstyle\sum_n \mu_n^I(\sum_k \alpha_k b_{k,n}^I - B_n^I),\end{aligned} \tag{25}$$

where $\mu^E=(\mu_1^E,\ldots,\mu_N^E)$ and $\mu^I=(\mu_1^I,\ldots,\mu_N^I)$ are the dual variables associated with constraints (19, 20). The Lagrange dual function $g(\cdot):\mathcal{R}^N\times\mathcal{R}^N\rightarrow\mathcal{R}$ is defined as the minimum value of the Lagrangian $\mathcal{L}(\alpha,\mu^E,\mu^I)$ over α:

$$g(\mu^E,\mu^I) = \min_{\alpha}\mathcal{L}(\alpha,\mu^E,\mu^I) = \mathcal{L}(\alpha^*,\mu^E,\mu^I) \tag{26}$$

Thus, the dual problem of the optimization problem (21) and (22) is:

$$d^* = \max_{\mu^E\in\mathcal{R}_+^N,\mu^I\in\mathcal{R}_+^N} g(\mu^E,\mu^I). \tag{27}$$

It has been shown in the previous section that Slater's condition holds, which means that there exists a strictly feasible solution that satisfies strict inequality constraints. Based on Slater's theorem, the strong duality holds [15]. Therefore, there is no duality gap between p^* and d^*, i.e., there exists μ^{E*},μ^{I*} such that $d^* = g(\mu^{E*},\mu^{I*}) = \mathcal{L}(\alpha^*,\mu^{E*},\mu^{I*}) = p^*$.

To sum up, in order to obtain the optimal solution of the primal problem, we can solve the dual problem (27) that has zero duality gap. Substituting $g(\mu^E,\mu^I)$ using Eqs. (26) and (25), we have the following equivalent dual problem:

$$\begin{aligned}\max_{\mu_n^E\geq 0,\mu_n^I\geq 0}\ &\min_{\alpha}\textstyle\sum_n \mu_n^E(\sum_k \alpha_k b_{k,n}^E - B_n^E) \\ &+ \textstyle\sum_n \mu_n^I(\sum_k \alpha_k b_{k,n}^I - B_n^I) - \sum_k \log\alpha_k.\end{aligned}$$

As clearly observed, this problem is a maximization problem over variables μ_n^E and μ_n^I, subject to the constraints that each variable is a positive real number. In the objective function, only the previous two terms are dependent upon the variables. Without coupling constraints, it can be further divided into N subproblems as follows:

$$\begin{aligned}\max_{\mu_n^E,\mu_n^I}\quad &\min_{\alpha}\mu_n^E(\textstyle\sum_k \alpha_k b_{k,n}^E - B_n^E) + \mu_n^I(\sum_k \alpha_k b_{k,n}^I - B_n^I) \\ \text{s.t.}\quad &\mu_n^E\geq 0,\ \ \mu_n^I\geq 0.\end{aligned}$$

Each of the subproblem has two variables μ_n^E and μ_n^I, which can be solved at the corresponding server n.

6.2 Distributed Algorithm Using Gradient Projection

We now design a distributed algorithm based on the gradient projection method to solve the dual problem, thus the optimal solution α^* for the primal problem can be further derived. The algorithm is proved to converge and is easy to

be implemented at each physical machine, with a small overhead.

According to the gradient projection method, we update variable μ_n (representing both μ_n^E and μ_n^I for simplicity) iteratively at each server $n \in \{1, 2, \ldots, N\}$ and each $t \geq 0$ with the following recursion scheme:

$$\begin{aligned}\mu_n^{(t+1)} &= \Big[\mu_n^{(t)} + \zeta \frac{\partial g}{\partial \mu_n}\Big]^+, \\ &= \Big[\mu_n^{(t)} + \zeta\Big(\textstyle\sum_k \alpha_k b_{k,n} - B_n\Big)\Big]^+,\end{aligned} \tag{28}$$

where $\zeta > 0$ denotes the step-size, $[x]^+ = \max\{0, x\}$, μ_n stands for (μ_n^E, μ_n^I), and the same applies to $b_{k,n}$ and B_n.

Theorem 1. *Given $\mu^{(0)} \in \mathcal{R}_+^{2N}$ and $\zeta \in (0, 2/\tilde{K}]$, where $\tilde{K} = \sqrt{2N}\sum_n \sum_k \big(b_{k,n}^E + b_{k,n}^I\big) C_k \overline{\alpha}_k^2$ and $C_k = \max_n\{b_{k,n}^E, b_{k,n}^I\}$, the recursive sequence $\{\mu^{(t)}\}$ generated by Eq. (28) converges to the dual optimum μ^*, i.e., $\lim_{t\to\infty} \mu^{(t)} = \mu^*$.*

Proof. We first prove that the gradient of the dual function $g(\mu)$ is $\tilde{K}$-Lipschitz continuous.

Let us define a function $\theta_k(\cdot)$ as

$$\theta_k(x) = 1/x, \quad x \geq 1/\overline{\alpha}_k,$$

where $\overline{\alpha}_k = \min\{\min_{n\in\mathcal{N}, b_{k,n}^E \neq 0} \frac{B_n^E}{b_{k,n}^E}, \min_{n\in\mathcal{N}, b_{k,n}^I \neq 0} \frac{B_n^I}{b_{k,n}^I}\}$, representing the upper bound for the performance of application k (achieved when all the link bandwidth is allocated to k). The Lipschitz constant of $\theta_k(x)$ is easily obtained as $\overline{\alpha}_k^2$.

We also define $d_n^E(\mu)$ as the partial derivative of $g(\mu^E, \mu^I)$ with respect to μ_n^E:

$$d_n^E(\mu) = \frac{\partial g(\mu^E, \mu^I)}{\partial \mu_n^E} = \textstyle\sum_k \alpha_k^* b_{k,n}^E - B_n^E \tag{29}$$

Based on Eq. (24), we can represent α_k^* as a $\theta_k(\cdot)$ function:

$$\begin{aligned}\alpha_k^*(\mu) &= 1/\big(\textstyle\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I\big) \\ &= \theta_k\big(\textstyle\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I\big),\end{aligned}$$

Thus, $d_n^E(\mu)$ can be represented as:

$$d_n^E(\mu) = \textstyle\sum_k b_{k,n}^E \theta_k\big(\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I\big) - B_n^E,$$

Then we can derive the following:

$$\begin{aligned}&|d_n^E(\mu) - d_n^E(\mu')| \\ &\leq \textstyle\sum_k b_{k,n}^E \big|\theta_k\big(\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I\big) \\ &- \theta_k\big(\textstyle\sum_n \mu_n'^E b_{k,n}^E + \sum_n \mu_n'^I b_{k,n}^I\big)\big| \\ &\leq \textstyle\sum_k b_{k,n}^E \overline{\alpha}_k^2 \sum_n \big(b_{k,n}^E |\mu_n^E - \mu_n'^E| + b_{k,n}^I |\mu_n^I - \mu_n'^I|\big) \\ &\leq \textstyle\sum_k b_{k,n}^E \overline{\alpha}_k^2 C_k \sum_n \big(|\mu_n^E - \mu_n'^E| + |\mu_n^I - \mu_n'^I|\big) \\ &= \big(\textstyle\sum_k b_{k,n}^E C_k \overline{\alpha}_k^2\big) \|\mu - \mu'\|_1,\end{aligned} \tag{30}$$

where $\|\cdot\|_1$ is the L_1 norm in vector space, and $C_k = \max_n\{b_{k,n}^E, b_{k,n}^I\}$. Similarly, we have

$$|d_n^I(\mu) - d_n^I(\mu')| \leq \big(\textstyle\sum_k b_{k,n}^I C_k \overline{\alpha}_k^2\big) \|\mu - \mu'\|_1. \tag{31}$$

Summing up Eqs. (30) and (31) over all $n \in \mathcal{N}$ yields:

$$\begin{aligned}&\|d(\mu) - d(\mu')\|_1 \\ &= \textstyle\sum_n \Big(|d_n^E(\mu) - d_n^E(\mu')| + |d_n^I(\mu) - d_n^I(\mu')|\Big) \\ &\leq \Big(\textstyle\sum_n \sum_k \big(b_{k,n}^E + b_{k,n}^I\big) C_k \overline{\alpha}_k^2\Big) \|\mu - \mu'\|_1\end{aligned}$$

For any $\mu \in \mathcal{R}_+^{2N}$, we have $\|\mu\| \leq \|\mu\|_1 \leq \sqrt{2N}\|\mu\|$ ($\|\cdot\|_2$ or $\|\cdot\|$ is the L_2 norm or Euclidean norm) in metric space. Thus, we have the following result:

$$\begin{aligned}&\|d(\mu) - d(\mu')\| \leq \|d(\mu) - d(\mu')\|_1 \\ &\leq \Big(\textstyle\sum_n \sum_k \big(b_{k,n}^E + b_{k,n}^I\big) C_k \overline{\alpha}_k^2\Big) \|\mu - \mu'\|_1 \\ &\leq \Big(\sqrt{2N}\textstyle\sum_n \sum_k \big(b_{k,n}^E + b_{k,n}^I\big) C_k \overline{\alpha}_k^2\Big) \|\mu - \mu'\|\end{aligned}$$

Therefore, the Lipschitz constant of $\nabla g(\mu)$ is $\tilde{K} = \sqrt{2N}\sum_n \sum_k \big(b_{k,n}^E + b_{k,n}^I\big) C_k \overline{\alpha}_k^2$. According to the proof in [16], if $\nabla g(\mu)$ is $\tilde{K}$-Lipschitz continuous, then given a step-size $\zeta \in (0, 2/\tilde{K}]$, $\mu^{(t)}$ will converge in μ^* as $t \to \infty$. □

Algorithm 2. Distributed Bandwidth Allocation to Achieve Maximal Social Welfare, with a Tunable degree of Relaxation on Performance-Centric Fairness

Input:
Bandwidth capacity: $B_n^E, B_n^I, \ \forall n \in \mathcal{N}$;
Network load matrix $\mathcal{D}_{i,j}^k$ and weight $w_k, \ \forall k \in \mathcal{K}$;
Tunable relaxation on fairness: $S \in (0, S^*]$;
Task placement across physical machines: $X_{k,n}^i$;
Iteration times: T; Step-size: $\zeta \in (0, 2/\tilde{K}]$;
Output:
Bandwidth allocation for all applications: $r_{i,j}^k$;

```
Initialize α_k = w'_k S, ∀k ∈ K;
Calculate b^E_{k,n} and b^I_{k,n} based on Eqs. (13) and (14);
while iterations < T do
    for all physical machine n do
        μ^E_n = max(0, μ^E_n + ζ(Σ_k α_k b^E_{k,n} − B^E_n));
        μ^I_n = max(0, μ^I_n + ζ(Σ_k α_k b^I_{k,n} − B^I_n));
    for all α_k do
        α_k = 1 / (Σ_n μ^E_n b^E_{k,n} + Σ_n μ^I_n b^I_{k,n});
        if α_k < w'_k S then
            α_k = w'_k S;
    iterations ++;
for all r^{i,j}_k do
    r^{i,j}_k = α_k · D^{i,j}_k;
```

Since there is no duality gap between the dual problem and the primal problem, $\alpha(\mu^{(t)})$ associated with μ converges to the primal optimum, i.e.,

$$\lim_{t\to\infty} \alpha(\mu^{(t)}) = \alpha^*,$$

With the theoretical guidelines so far, we are able to design Algorithm 2 for distributed bandwidth allocation, which can be implemented at each physical machine in parallel. The only required state information to be maintained on each machine is the weights and network load matrices of

the applications that have their tasks placed on this machine, which is easily obtained. For example, in MapReduce, the job tracker can monitor the progress of all the tasks, thus have an overview of the volumes of traffic to be transferred among tasks.

In each iteration, the dual variables μ_n^E and μ_n^I will be updated at each machine n following Eq. (28), given the step-size ζ bounded by the global constant $2/\tilde{K}$ and the performance of each application k that has its tasks placed on this machine, *i.e.*, $b_{k,n}^E \neq 0$ or $b_{k,n}^I \neq 0$. Given the updated dual variables, application performance α_k can be updated in the application manager, according to Eq. (24). Note that $b_{k,n}^E$ and $b_{k,n}^I$ are 0 if none of the tasks of application k is placed on machine n. Hence, computing α_k only requires μ_n^E and μ_n^I from those machines where at least one task of application k is placed, which incurs a small communication overhead.

If the gradient in Eq. (29) is negative, which means that there is residual egress bandwidth on machine n, then μ_n^E will decrease and α_k will increase, indicating that the idle bandwidth will be utilized to increase application performance. If the computed performance is beyond its lower bound, it will be set as the bound to restrict the allocation in the feasible sets. Finally, when the dual variables converge to the optimum, the rates of the flows on server n can be easily computed and allocated.

7 Performance Evaluation

In this section, we first evaluate the overall application performance and the total utilities achieved by our bandwidth allocation algorithms with the guidance of weighted performance-centric fairness (either the extended version of fairness focusing on the bandwidth utilization or the tunable relaxation on fairness targeting the social welfare), compared with traditional per-flow fairness. Then we investigate how our second algorithm performs in tuning the tradeoff between maximizing the total utility and maintaining performance-centric fairness. We further evaluate the performance of the distributed algorithm with respect to the convergence. Finally, we discuss the practical concern of our algorithms.

7.1 Comparing Different Versions of Fairness

We first simulate a small private datacenter with 20 homogeneous physical machines, with 10 GB/s egress and ingress links. Without loss of generality, there are 6 data parallel applications running concurrently, each with 6 tasks randomly placed across the datacenter. The weights of these applications, referred to as A, B, C, D, E and F for convenience, are set as 1, 2, 3, 4, 5, 6, respectively. For each application, half of their tasks are transferring the intermediate data to another half, with the traffic volumes randomly chosen in the range of $[200, 300]$ MB. Each machine is able to shape the bandwidth of its flows.

Given a problem setting resulted from a randomization, we realize our two algorithms respectively. For Algorithm 2, the tunable relaxation on fairness S is set as the maximal value, which represents no relaxation. For comparison, we also apply the traditional flow-level max-min fair bandwidth allocation (referred to as *Per-flow fair*) used in TCP to the same setting. The performance of each application, measured as the reciprocal of its network transfer time as in Eq. (7), achieved by the three algorithms is illustrated in Fig. 4.

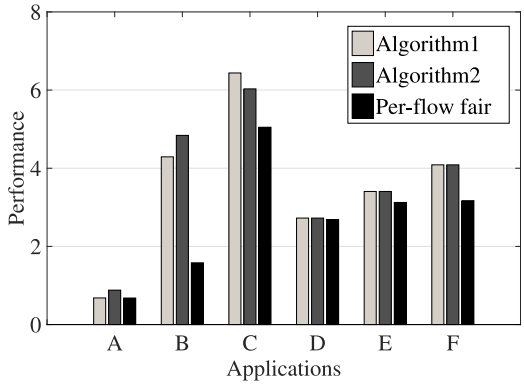


Fig. 4. Performance of applications A, B, C, D, E and F achieved by Algorithm 1, Algorithm 2 and *Per-flow fair*, in a small datacenter with 20 servers. The weights are 1, 2, 3, 4, 5 and 6, respectively.

It is obvious that Algorithm 1 and Algorithm 2 outperform *Per-flow fair* to a large extent, both with up to $1.4X$ performance improvement. The reason is that weighted performance-centric fairness is aware of the characteristic of data parallel applications, so that any amount of bandwidth allocated among applications contributes to the performance gain. In contrast, with *Per-flow fair* allocation that is application agnostic, although some flows of an application may get a large amount of bandwidth to finish quickly, the application performance is determined by the slowest flow.

The application performance achieved by Algorithm 1 and Algorithm 2 is slightly different, because of the different allocation of the residual bandwidth after guaranteeing maximal weighted performance-centric fair share for all the applications. Algorithm 1 iteratively finds the applications that can still improve their performance with more bandwidth, and allocates available bandwidth to them so that the improvement of their performance is weight proportional. Algorithm 2, in contrast, allocates the residual bandwidth so as to maximize the total utility of all the applications. Note that in this particular case, the performance of D, E and F is exactly their respective fair share, with the same proportionality to their weights, while A, B and C are allocated the residual bandwidth by the two algorithms to improve efficiency, so that they achieve better performance beyond their original share. It is also worth noting that D achieves almost the same performance with the three algorithms. The reason is that with different allocation from a different algorithm, the original bottlenecked flow may be accelerated, but another flow may become the bottleneck as its sharing flows are allocated with more bandwidth. If the new bottlenecked flow has the same completion time with the original one, the application performance will remain the same, which explains the performance of D.

With respect to the total utility, Algorithm 2 achieves 9.9305, larger than those achieved by Algorithm 1 and *Per-flow fair*, which are 9.4796 and 7.1747, respectively. Since the fairness relaxation parameter S is set as the maximal value, the utility maximization in Algorithm 2 is restricted by the fairness constraint to the largest extent. If we reduce S to have more relaxation on fairness, the total utility will be further increased.

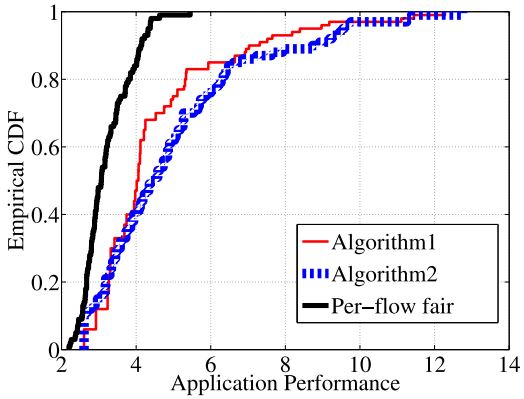


Fig. 5. CDFs of the performance of 100 applications achieved by Algorithm 1, Algorithm 2 and *Per-flow fair*, in a large datacenter with 100 servers. The weights are the same.

Next, we extend our evaluation to a larger scale, with the number of machines increased to 100, and the number of applications increased to 100. For simplicity, all the applications have the same weight, and the fairness relaxation parameter S is set as its maximal value. Fig. 5 presents the empirical CDFs of the application performance of the 100 applications achieved with three different algorithms. As is shown, all the application performance achieved with *Per-flow fair* is no larger than 5.5, while 17 percent of applications with Algorithm 1 and 30 percent of applications with Algorithm 2 achieve no smaller than 5.5, respectively. Clearly, the application performance achieved by our algorithms is much better than that achieved by *Per-flow fair*. Moreover, with respect to the total utilities, which are 209.4812, 219.7600 and 165.6202, respectively, our algorithms also outperform *Per-flow fair*.

7.2 Investigating Tunable Relaxation on Fairness

We continue to investigate the properties and evaluate the performance of Algorithm 2, with a detailed analysis in two typical scenarios shown in Fig. 6.

In Scenario 1, three applications A, B and C, with weights of 2, 1 and 1, encounter the same bottleneck at physical machine P, where their tasks have 100 MB, 100 MB and 200 MB intermediate data to be transferred, respectively. The link (both ingress and egress) bandwidth capacity of P is 1 GB/s. Based on Eq. (15), the maximum total performance-centric share (S^*) of all the applications is computed as 8.

As shown in Fig. 7, as we relax the performance-centric fairness, i.e., as we reduce S, the total utility of all the applications increases, indicating that the bandwidth becomes more efficiently utilized. Fig. 8 delves into several points of the tradeoff curve to show the performance achieved by these applications.

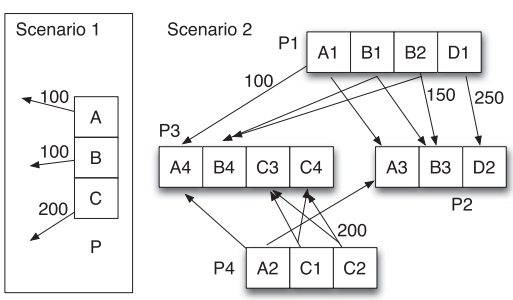


Fig. 6. Two scenarios for evaluating the proposed bandwidth allocation.

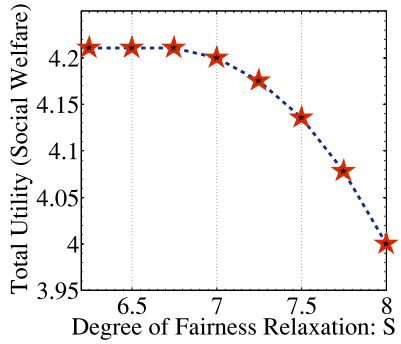


Fig. 7. The tradeoff between the total performance-centric fair share and the total utility in scenario 1.

When S is at its maximum as 8, the performance of A, B and C is 4, 2, 2 respectively, proportional to their weights, while the total utility is the lowest in the tradeoff curve. When we reduce S, the amount of bandwidth required to guarantee the total performance-centric share is reduced. The residual bandwidth can thus be freely allocated to the application whose utility will increase the most given this amount of bandwidth. In this scenario, B is the most competitive in grabbing the free bandwidth. It has fewer data to be sent compared with C and its current performance is lower than A, so that its performance improvement will make the total utility increase the most.

This analysis is supported by Fig. 8. While all the applications are guaranteed the minimum performance-centric share, which is decreasing as we reduce S, B can achieve higher performance than its minimum share by grabbing the free bandwidth. For example, when $S = 7$, the minimum performance-centric share is $\frac{2}{2+1+1} \cdot 7 = 3.5$ for A, and $\frac{1}{2+1+1} \cdot 7 = 1.75$ for B and C. As shown in Fig. 8, A and C achieve the performance of 3.5 and 1.75 respectively, while B achieves 3, which is more than its minimum share. When S decreases to 6.5, the total utility will achieve its maximum. It will not increase any further with S decreasing to 6 or below. Correspondingly, the application performance does not change when S decreases below 6.5 according to our results (we only show $S = 6.5$ and $S = 6$ as examples in the figure).

Now we consider Scenario 2 as shown in Fig. 6, where four applications A, B, C and D with the same weight have their tasks placed across 4 servers, each with the bandwidth

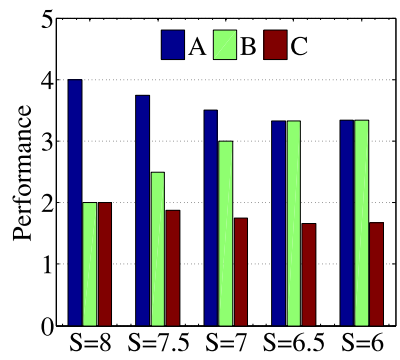


Fig. 8. Performance of application A, B and C when S is tuned at different values in scenario 1.

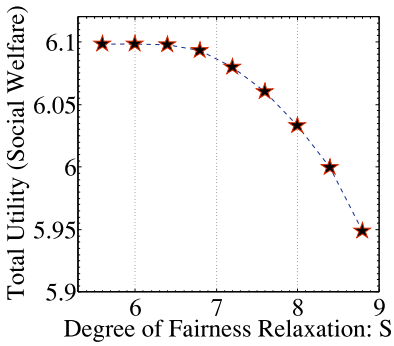


Fig. 9. The tradeoff between the total performance-centric fair share and the total utility in scenario 2.

capacity of 3 GB/s. The bottleneck link encountered by A, B and C is at $P3$, while the bottleneck of D is $P1$. If we strictly follow performance-centric fairness, S^* is computed as $\frac{120}{13}$, and D will not be allowed to use the idle bandwidth on server $P1$ and $P2$. With the relaxation on fairness by reducing S, the idle bandwidth will be utilized, and there will be residual bandwidth after guaranteeing the minimum share to all the applications. The residual bandwidth can thus be allocated among applications to improve the total utility the most. Such a tradeoff between the total utility and the degree of fairness relaxation is verified in Fig. 9.

Fig. 10 illustrates the application performance achieved at different degrees of relaxation on performance-centric fairness. As we tune S, B always achieves no more than its minimum guaranteed performance-centric share, while other applications achieve higher performance than their minimum shares. This can be explained by B's heavy network load, which results in a smaller amount of utility increase compared with other applications, given the same amount of bandwidth allocation. Hence, the free bandwidth at $P1$ will be allocated to D, and the free bandwidth at $P3$ will be allocated to A, to improve the overall utility.

7.3 Convergence

Finally, we evaluate the convergence of Algorithm 2 in Scenario 1. As shown in Fig. 11, the dash lines and solid lines represent the performance of applications at different values of S, respectively. We choose the step-size according to Theorem 1 to guarantee the convergence. The performance of all the applications converges within about 70 iterations. To further evaluate our algorithm at a much larger scale, we simulate a datacenter with 100 physical machines hosting 100 data parallel applications, each of which has 4 tasks placed on different machines. Tasks of applications

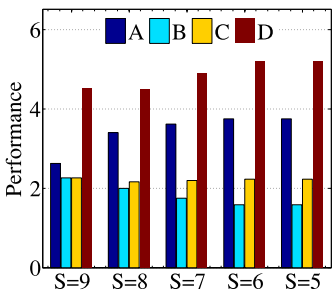


Fig. 10. Performance of application A, B C and D when S is tuned at different values in scenario 2.

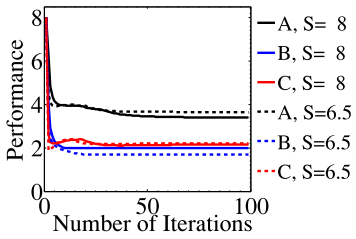


Fig. 11. Convergence of performance of A, B and C when S is 8 and 6, respectively, in Scenario 1.

have different amounts (randomly chosen between 100 MB to 200 MB for simplicity) of data to be transferred in their communication stages. Fig. 12 shows the performance change of two randomly selected applications with an increasing number of iterations. We can see that the application performance is able to converge within 800 iterations.

7.4 Discussion

Algorithm 1 can be solved in polynomial time, as analyzed in Section 4, which is feasible in practice and fast especially in modern datacenters with powerful computation engine. Implementing this algorithm requires coordinating network transfers based on application demand, which can be realized with Software-Defined Network (SDN) [17]. The application demand can either be conveyed from application managers [18], [19], or anticipated by the SDN controller [20]. To be particular, the network load matrices of all the sharing flows should be obtained as input to Algorithm 1. With Apache Spark [5] as an example, the size and location of the output data from each `map` task can be obtained from the `MapOutputTracker` module. Once the placement of `reduce` tasks has been scheduled, the network load matrices can be easily obtained and conveyed to the SDN controller. As application weights and bandwidth capacities are also available, Algorithm 1 will run in the controller to calculate the bandwidth allocation, which is further enforced by setting maximal rates for switch queues and forwarding packets correspondingly [21].

Similarly, SDN can be applied to enforce the calculated bandwidth allocation by Algorithm 2. Specifically, the calculated rates at each server should be conveyed to the SDN controller, which then set up rate-limiting queues and forward packets accordingly. Moreover, a protocol is required for the communication between application managers and servers, to update the parameters when running the algorithm.

8 Related Work

Fair Bandwidth Allocation. Bandwidth allocation among multiple tenants in public cloud datacenters has received a

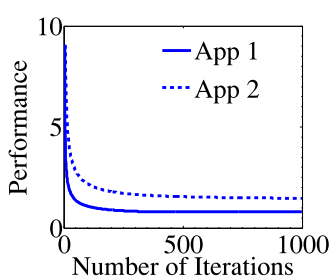


Fig. 12. Convergence of performance of two applications in a large scale private datacenter.

substantial amount of recent research attention [7], [8], [9], [13], [21], [22], [23], [24], [25]. The general focus of these works has been on ensuring fair allocation among different tenants according to their payments. For example, NetShare [9] achieves tenant level fairness while Seawall [8] achieves fairness between VM sources. FairCloud [7] allocates bandwidth on congested links based on the weights of the communicating VM-pairs, thus achieving VM-pair level fairness. However, in our setting of a private datacenter running data parallel frameworks, the previous notion of fairness is not applicable.

In the context of a private datacenter, Kumar et al. proposed that bandwidth should be allocated with the awareness of the communication patterns of data parallel applications [26]. Their focus is mainly on effective parallelization for each application, i.e., the completion time should be N times faster if the application parallelizes by N. However, when tasks of one application share bandwidth with tasks of different applications at different bottlenecks, it is not known what performance each application should expect, without a clear definition of fairness with respect to application performance. In contrast, our proposition of performance-centric fairness [27] fills this gap, and offers a definitive guide to the problem of bandwidth allocation among multiple data parallel applications in a private datacenter.

Such a definition of fairness has been further explored in our following works which jointly considered the flexibility of task placement [28] and path selection [29], respectively. It has been demonstrated that performance-centric fairness has effectively served as a guidance in more complex scenarios in practice, when additional dimensions of problems are involved. Complementary to such a broader extension of the performance-centric fairness, in this paper, we choose a deeper extension to investigate the inherent tradeoff between performance-centric fairness and efficiency from two different perspectives. In particular, a new bandwidth allocation algorithm has been designed and evaluated to explore the fairness-efficiency tradeoff thoroughly. With our study, insights can be provided for service providers to deploy their bandwidth allocation strategy at the sweet spot.

Game-theoretic Approach. Guo, et al. modeled their bandwidth allocation problems as Nash Bargaining games, where each VM as a player cooperates in the game to achieve both the max-min fairness and the optimal social utility represented by the Nash product [13], [21]. In contrast, our formulation is a general resource allocation problem, where the social utility is not restricted to be the Nash product and the guarantee of fairness is modeled in the constraints. Although our problem formulation has a similar form with these works, which naturally can be solved similarly by deducting analytic solution in a centralized manner or applying the standard technique of gradient descent in a distributed fashion, the meaning of our problem is different.

9 Concluding Remarks

Our focus in this paper is to study the sharing of link bandwidth among applications running data parallel frameworks in a private datacenter. With the guideline that performance achieved by applications should be proportional to their weights, we propose a rigorous definition of performance-centric fairness and investigate the tradeoff between efficiency and fairness. From two perspectives of interpreting the efficiency, we design two algorithms to arbitrate the efficiency-fairness tradeoff. The first algorithm gradually improves the bandwidth utilization by introducing a relaxed version of fairness. With respect to the second interpretation of efficiency, a distributed algorithm is designed by solving a utility-maximization problem, constrained by a tunable degree of relaxation on performance-centric fairness. Such an algorithm can be implemented on each physical machine in a lightweight fashion. Our extensive simulations have shown that both algorithms have effectively improved efficiency with some relaxation on fairness, which result in up to $1.4X$ better application performance compared with the traditional fairness. The distributed algorithm is demonstrated to provide the flexibility to balance the tradeoff between the total utility and the degree of fairness relaxation.

Acknowledgments

The research is supported in part by the NSERC Collaborative Research and Development Grant, and by grants from RGC GRF under the contracts 16211715 and 16206417, a grant from RGC CRF under the contract C7036-15G.

References

[1] A. Langville and C. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton, NJ, USA: Princeton Univ. Press, 2006.

[2] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the Netflix prize," in *Algorithmic Aspects in Information and Management*. Berlin, Germany: Springer, 2008, pp. 337–348.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[4] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," *ACM SIGOPS Operating Syst. Rev.*, vol. 41, no. 3, pp. 59–72, 2007.

[5] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. USENIX Symp. Networked Syst. Des. Implementation*, 2012, pp. 2–2.

[6] M. Chowdhury, M. Zaharia, J. Ma, M. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proc. ACM SIGCOMM*, 2011.

[7] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "FairCloud: Sharing the Network in Cloud Computing," in *Proc. ACM SIGCOMM Conf.*, 2012, pp. 98–109.

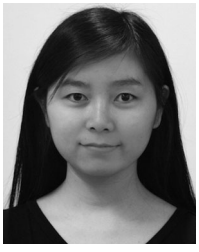
[8] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center network," in *Proc. USENIX Conf. Networked Syst. Des. Implementation*, 2011, pp. 309–322.

[9] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese, "NetShare: Virtualizing data center networks across services," University of California, San Diego, CA, USA, Tech. Rep. CS2010–0957, May 2010.

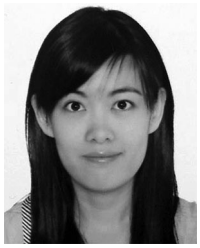
[10] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 266–277.

[11] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 51–62.

[12] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, 2011, pp. 242–253.
[13] J. Guo, F. Liu, D. Zeng, J. Lui, and H. Jin, "A cooperative game based allocation for sharing data center networks," in *Proc. IEEE INFOCOM*, 2013, pp. 2139–2147.
[14] M. Welzl, *Network Congestion Control: Managing Internet Traffic*. Hoboken, NJ, USA: Wiley, 2005.
[15] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[16] H. Yaïche, R. Mazumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 667–678, Oct. 2000.
[17] N. McKeown, "Software-defined networking," *INFOCOM keynote Talk*, 2009, http://infocom2009.ieee-infocom.org/keynotes.html
[18] G. Wang, T. Ng, and A. Shaikh, "Programming your network at run-time for big data applications," in *Proc. ACM Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 103–108.
[19] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory networking: An API for application control of SDNs," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 327–338.
[20] A. Das, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and C. Yu, "Transparent and flexible network management for big data processing in the cloud," in *Proc. USENIX HotCloud Workshop*, 2013, pp. 1–6.
[21] J. Guo, F. Liu, H. Tang, Y. Lian, H. Jin, and J. C. Lui, "Falloc: Fair network bandwidth allocation in IaaS datacenters via A bargaining game approach," in *Proc. IEEE Int. Conf. Netw. Protocols*, 2013, pp. 1–10.
[22] J. Guo, F. Liu, J. Lui, and H. Jin, "Fair network bandwidth allocation in IaaS datacenters via A cooperative game approach," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 873–886, Apr. 2016.
[23] V. Jeyakumar, M. Alizadeh, D. Mazieres, B. Prabhakar, C. Kim, and A. Greenberg, "EyeQ: Practical network performance isolation at the Edge," in *Proc. USENIX Networked Syst. Des. Implementation*, 2013, pp. 297–312.
[24] H. Ballani, K. Jang, T. Karagiannis, C. Kim, D. Gunawardena, and G. OShea, "Chatty tenants and the cloud network Sharing problem," in *Proc. USENIX Conf. Networked Syst. Des. Implementation*, 2013, pp. 171–184.
[25] L. Popa, P. Yalagandula, S. Banerjee, and J. Mogul, "ElasticSwitch: Practical work-conserving bandwidth guarantees for cloud computing," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 351–362.
[26] G. Kumar, M. Chowdhury, S. Ratnasamy, and I. Stoica, "A case for performance-centric network allocation," in *Proc. USENIX Conf. Hot Topics Cloud Comput.*, 2012, pp. 9–9.
[27] L. Chen, Y. Feng, B. Li, and B. Li, "Towards performance-centric fairness in datacenter networks," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 1599–1607.
[28] L. Chen, B. Li, and B. Li, "Barrier-aware max-min fair bandwidth sharing and path selection in datacenter networks," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2016, pp. 151–160.
[29] L. Chen, B. Li, and B. Li, "Surviving failures with performance-centric bandwidth allocation in private datacenters," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2016, pp. 52–61.



Li Chen received the BEngr degree from the Department of Computer Science and Technology, Huazhong University of Science and Technology, China, in 2012 and the MASc degree from the Department of Electrical and Computer Engineering, University of Toronto, in 2014. She is currently working toward the PhD degree in the Department of Electrical and Computer Engineering, University of Toronto. Her research interests include big data analytics systems, cloud computing, datacenter networking, and resource allocation.



Yuan Feng received the BEngr degree from the School of Telecommunications, Xidian University, Xi'an, China, in 2008, and the MASc and PhD degrees from the Department of Electrical and Computer Engineering, University of Toronto, Canada, in 2010 and 2013, respectively. Her research interests include optimization and design of large-scale distributed systems and cloud services.

Baochun Li received the PhD degree from the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, in 2000. Since then, he has been in the Department of Electrical and Computer Engineering, University of Toronto, where he is currently a professor. He holds the bell canada endowed chair in Computer Engineering since August 2005. His research interests include large-scale distributed systems, cloud computing, peer-to-peer networks, applications of network coding, and wireless networks. He is a member of the ACM and a fellow of the IEEE.

Bo Li is a professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He holds the Cheung Kong chair professor in Shanghai Jiao Tong University. Prior to that, he was with IBM Networking System Division, Research Triangle Park, North Carolina. He was an adjunct researcher with Microsoft Research Asia-MSRA and was a visiting scientist in Microsoft Advanced Technology Center (ATC). He has been a technical advisor for China Cache Corp. (NASDAQ CCIH) since 2007. He is an adjunct professor with the Huazhong University of Science and Technology, Wuhan, China. His recent research interests include: large-scale content distribution in the Internet, Peer-to-Peer media streaming, the Internet topology, cloud computing, green computing and communications. He is a fellow of the IEEE for "contribution to content distributions via the Internet". He received the Young Investigator Award from the National Natural Science Foundation of China (NSFC) in 2004. He served as a Distinguished lecturer of the IEEE Communications Society (2006-2007). He was a corecipient for three Best Paper Awards from IEEE, and the Best System Track Paper in ACM Multimedia (2009).