Towards Performance-Centric Fairness in Datacenter Networks

Li Chen[1] Yuan Feng[2] Baochun Li[1] Bo Li[3]
[1]University of Toronto
[2]The Hong Kong Polytechnic University
[3]The Hong Kong University of Science and Technology

***Abstract*— Fair bandwidth allocation in datacenter networks has been a focus of research recently, yet this has not received adequate attention in the context of private cloud, where link bandwidth is often shared among applications running data parallel frameworks, such as MapReduce. In this paper, we introduce a rigorous definition of *performance-centric fairness*, with the guiding principle that the performance of data parallel applications should be proportional to their weights. We investigate the problem of maximizing application performance while maintaining strict performance-centric fairness and present the inherent tradeoff between resource utilization and fairness. We then formulate the link bandwidth allocation problem with the objective of maximizing social welfare across all applications, so that resource utilization can be manipulated and improved by allowing a tunable degree of relaxation on performance-centric fairness. Based on dual based decomposition, we present a distributed algorithm to solve this problem, and evaluate its performance with extensive simulations.**

I. 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] and Dryad [4].

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 CPU (used in the computation stages) and network bandwidth (used in the communication stages).

This work is partially supported by the SAVI NSERC Strategic Networks Grant.

In the context of a privately operated datacenter, how should *link bandwidth* — arguably the most critical resource in datacenter networks — be shared among multiple data parallel applications? It is commonly accepted in the literature that bandwidth should be shared in a *fair* manner (*e.g.*, [5]), 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 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 the resource utilization in a datacenter, in that bandwidth should not be left unused. 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 resource utilization and maintaining strict fairness, simply because some access links are more heavily loaded than others in a datacenter. In this paper, we arbitrate such conflicting objectives by introducing tunable degrees of relaxation on performance-

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

centric fairness, such that the resource utilization can be further improved.

The upshot in this paper revolves around the new optimization problem to maximize resource utilization 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 the resource utilization, 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 with the tradeoff of relaxing fairness to a certain degree.

With a sharp focus on performance-centric fairness in private datacenter networks, in Sec. II, 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 Sec. III, which maximizes resource utilization while maintaining the strict notion of weighted performance-centric fairness. In Sec. IV, we formulate our new problem that better arbitrates the tradeoff between resource utilization and fairness. With a detailed analysis on the nature of our optimization problem, we apply dual based decomposition to solve the centralized problem in Sec. V, 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 Sec. VI has shown that our distributed solution improves the resource utilization while maintaining performance-centric fairness with a tunable degree of relaxation.

II. A Case for Performance-Centric Fairness

Data parallel applications partition their input data into multiple splits, which are processed in parallel by computation tasks. Even though additional computation tasks will help to reduce the completion time in a computation stage, the total amount of computation workload remains the same, and is not affected by the number of parallel tasks used once the size of the input data is fixed. However, the 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 to 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 $B4$.

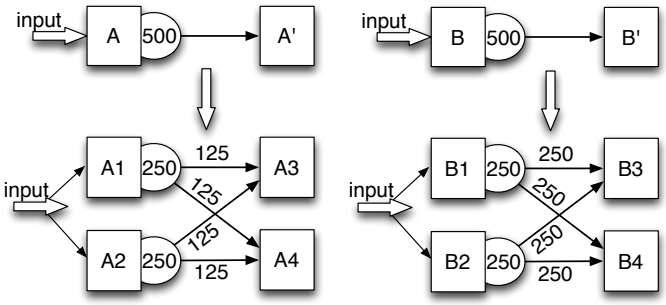

Fig. 1: The amount of data to be transmitted when parallelizing a MapReduce application with the *shuffle* communication pattern, and a machine learning application with the *broadcast* communication pattern.

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 $P2$.

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

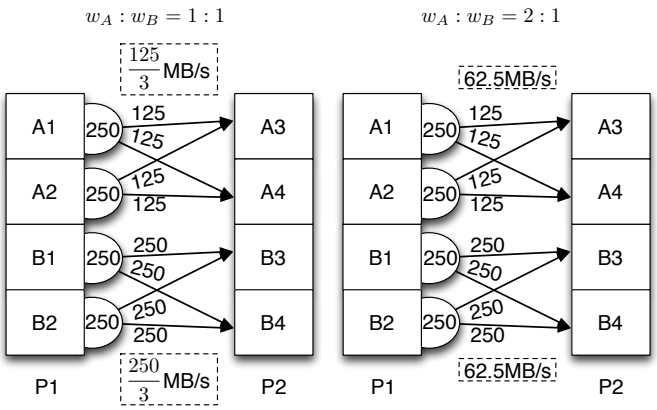

Fig. 2: Examples of bandwidth allocation achieving weighted performance-centric fairness in two cases: 1) both applications have the same weight; 2) the two applications have different weights.

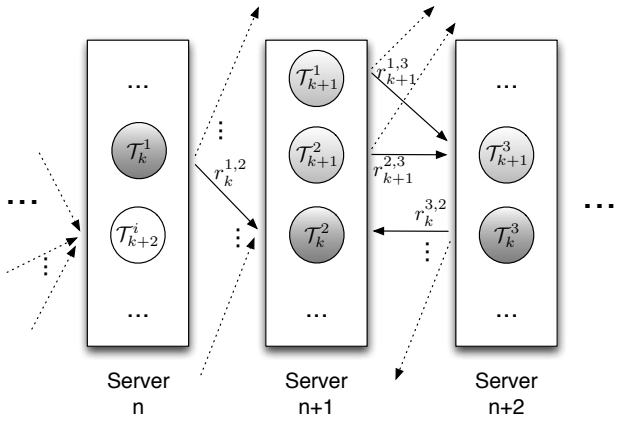

Fig. 3: An illustration of a private datacenter hosting multiple data parallel applications.

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.

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 $k \in \mathcal{K} = \{1, 2, ..., K\}$ 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:

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

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, weighted performance-centric 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.*, [6]), VM-pair level (*e.g.*, [5]) or the tenant level (*e.g.*, [7]) proposed in the literature in the context of datacenters in a public cloud. To achieve weighted performance-centric 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.

III. Allocating Bandwidth to achieve Weighted 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.

Consider a private datacenter shown in Fig. 3, 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) $n \in \mathcal{N} = \{1, 2, ..., N\}$, 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.*, [8]) and multi-tree topologies (*i.e.*, [9]), we assume a full bisection bandwidth network, where bandwidth is only bottlenecked at the access links of physical machines. 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 $k \in \mathcal{K} = \{1, 2, ..., K\}$ requires m_k tasks, represented by $\mathcal{T}_k = \{1, 2, ..., m_k\}$. The i-th 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 Sec. II, 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

$$\min_{i,j,\mathcal{D}_k^{i,j}\neq 0} \frac{r_k^{i,j}}{\mathcal{D}_k^{i,j}} = \frac{w_k}{\sum_k w_k} S \tag{3}$$

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.*,

$$X_{k,n}^i = \begin{cases} 1, \text{ when } \mathcal{T}_k^i \text{ is placed on server } n \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

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:

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

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:

$$\max_r \quad S \tag{6}$$

$$\text{s.t.} \quad \min_{i,j,\mathcal{D}_k^{i,j}\neq 0} \frac{r_k^{i,j}}{\mathcal{D}_k^{i,j}} = \frac{w_k}{\sum_k w_k} S, \quad \forall k \in \mathcal{K} \tag{7}$$

$$\sum_k \sum_i \sum_{j,X_{k,n}^j=0} r_k^{i,j} \cdot X_{k,n}^i \leq B_n^E, \quad \forall n \in \mathcal{N} \tag{8}$$

$$\sum_k \sum_j \sum_{i,X_{k,n}^i=0} r_k^{i,j} \cdot X_{k,n}^j \leq B_n^I, \quad \forall n \in \mathcal{N} \tag{9}$$

where constraint (7) represents weighted performance-centric fairness, while constraints (8) and (9) correspond to the egress and ingress link capacity constraints at each machine.

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 the following constraint:

$$\min_{i,j} \frac{r_k^{i,j}}{\mathcal{D}_k^{i,j}} = \frac{r_k^{i,j}}{\mathcal{D}_k^{i,j}} = \frac{1}{t_k} = \alpha_k, \quad \forall i,j \in \mathcal{T}_k, \mathcal{D}_k^{i,j} \neq 0$$

where α_k denotes the performance of application k. Replacing the variables of $r_k^{i,j}$ with α_k yields

$$r_k^{i,j} = \frac{\mathcal{D}_k^{i,j}}{t_k} = \alpha_k \cdot \mathcal{D}_k^{i,j}, \quad \forall k \in \mathcal{K}, i,j \in \mathcal{T}_k \tag{10}$$

As a result, we can transform the previous optimization problem (6) to the following:

$$\begin{aligned} \max_{\alpha} \quad & S \\ \text{s.t.} \quad & \alpha_k = \frac{w_k}{\sum_k w_k} S, \quad \forall k \in \mathcal{K} \\ & \textstyle\sum_k \alpha_k \cdot b_{k,n}^E \leq B_n^E, \quad \forall n \in \mathcal{N} \\ & \textstyle\sum_k \alpha_k \cdot b_{k,n}^I \leq B_n^I, \quad \forall n \in \mathcal{N} \end{aligned}$$

where

$$b_{k,n}^E = \textstyle\sum_i \sum_{j,X_{k,n}^j=0} \mathcal{D}_k^{i,j} X_{k,n}^i \tag{11}$$

$$b_{k,n}^I = \textstyle\sum_j \sum_{i,X_{k,n}^i=0} \mathcal{D}_k^{i,j} X_{k,n}^j \tag{12}$$

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 easily derived as:

$$S^* = \min\{ \quad \min_{n,\sum_k b_{k,n}^E \neq 0} \frac{B_n^E}{\sum_k w_k' b_{k,n}^E} \quad , \quad \min_{n,\sum_k b_{k,n}^I \neq 0} \frac{B_n^I}{\sum_k w_k' b_{k,n}^I} \quad \} \tag{13}$$

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

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

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

However, by allocating bandwidth following the strict definition of weighted performance-centric fairness, some link bandwidth may not be fully utilized, which is not efficient with respect to bandwidth utilization.

Consider the example shown in Fig. 4, in which applications A, B and C all have the same weight. $A1$ has 200 MB data to be sent, while $B1$, $C1$ both generate 100 MB of intermediate data to be transmitted. All four physical machines have the same link (both egress and ingress link) bandwidth capacity of 300 MB/s. Based on Eq. (13), the maximum total performance-centric share is computed as $S^* = min\{min\{\frac{300}{\frac{1}{3}\cdot 200+\frac{1}{3}\cdot 100}, \frac{300}{\frac{1}{3}\cdot 100}\}, min\{\frac{300}{\frac{1}{3}\cdot 200}, \frac{300}{\frac{1}{3}\cdot 100+\frac{1}{3}\cdot 100}\}\} = 3$ (we omit the unit of s^{-1} in this paper for simplicity), which is constrained by the egress link capacity at $P1$. Hence, the

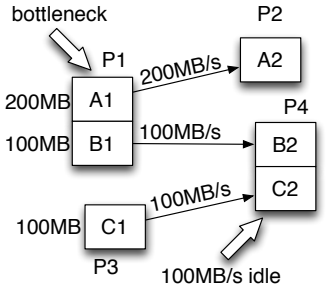

Fig. 4: The inefficiency of bandwidth utilization under weighted performance-centric fairness.

flow between $A1$ and $A2$ is allocated 200 MB/s, while the flows of B and C both obtain 100 MB/s, so that all three applications achieve the same transfer time of 1s. However, the link bandwidth on $P4$ is not efficiently utilized, with 100 MB/s bandwidth idled. To achieve better utilization of bandwidth, we should consider the tradeoff between fairness and resource utilization: with a relaxed notion of fairness, we allow C to utilize the idle bandwidth so that it can achieve a higher performance, improving the overall utilization of the datacenter.

IV. Trading Fairness for Utilization

To strive for a balance between the conflicting requirements of performance-centric fairness and maximum bandwidth utilization, we set the following principle for bandwidth allocation:

1) $\alpha_k \geq w_k^{'}S$, where $S \in (0, S^*]$ is considered as the degree of fairness relaxation that can be tuned to trade fairness for utilization. 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.

2) $\alpha_k \leq \overline{\alpha}$, where $\overline{\alpha}$ is the upper bound of performance for each application that can be decided by the datacenter operator. With respect to achieving a better utilization of bandwidth, some applications are more competitive than others. This constraint limits the performance of each application below a maximum, so as to prevent the extreme unfairness resulted from one application capturing all the available bandwidth while other applications starve.

With a certain degree of relaxation on performance-centric fairness, our objective of bandwidth allocation is to achieve the best utilization of bandwidth so as to maximize the overall social welfare of all applications. 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: $U_k(\alpha_k) = \log \alpha_k,\ \forall k \in \mathcal{K}$. As a result, we can formulate the following optimization problem:

$$\begin{aligned}
\max_{\alpha} \quad & \textstyle\sum_k U_k(\alpha_k) && (14)\\
\text{s.t.} \quad & \alpha_k \geq w_k^{'}S, \quad \forall k \in \mathcal{K} && (15)\\
& \alpha_k \leq \overline{\alpha}, \quad \forall k \in \mathcal{K} && (16)\\
& \textstyle\sum_k \alpha_k \cdot b_{k,n}^E \leq B_n^E, \quad \forall n \in \mathcal{N} && (17)\\
& \textstyle\sum_k \alpha_k \cdot b_{k,n}^I \leq B_n^I, \quad \forall n \in \mathcal{N} && (18)
\end{aligned}$$

where $b_{k,n}^E$ and $b_{k,n}^I$ are expressed in Eq. (11) and Eq. (12). Constraints (15) and (16) follow the aforementioned principles, while (17) and (18) 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 (14) to the following:

$$\begin{aligned}
\min_{\alpha} \quad & -\sum_k U_k(\alpha_k) && (19)\\
\text{s.t.} \quad & Eq.\ (15), (16), (17), (18)
\end{aligned}$$

Since $U_k(\alpha_k) = \log \alpha_k$ is strictly concave [10], the objective function of problem (19) is strictly convex. Moreover, all of the constraints are affine. Hence, problem (19) is a convex optimization problem [10].

Let $\lambda_k, \beta_k, \forall k \in \mathcal{K}$ denote the Lagrange multipliers associated with constraints (15) and (16) respectively, and $\mu_n^E, \mu_n^I, \forall n \in \mathcal{N}$ associated with capacity constraints (17) and (18) respectively. The Lagrangian of problem (19) is as follows:

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

It is obvious that there exists $\alpha = (\alpha_1, ..., \alpha_K)$ in the relative interior of the intersection of domains of all constraint functions, *i.e.*, $\alpha = (\alpha_1, ..., \alpha_K)$ satisfies the constraints: $w_k^{'}S < \alpha_k < \overline{\alpha}, \forall k \in \mathcal{K}$, $\sum_k \alpha_k b_{k,n}^E < B_n^E, \forall n \in \mathcal{N}$ and $\sum_k \alpha_k b_{k,n}^I < B_n^I, \forall n \in \mathcal{N}$. Hence, Slater's condition [10] is satisfied. And since the optimization problem (19) is differentiable and convex, the Karush-Kuhn-Tucker (KKT) conditions [10] are both sufficient and necessary for the optimality. Thus, we can derive the optimal solution by applying the KKT conditions:

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

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

and

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

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 $\beta_k = 0$ and

$$-\frac{1}{w_k^{'}S} - \lambda_k + \sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I = 0. \qquad (20)$$

Since $\lambda_k \geq 0$ and $\frac{1}{w_k^{'}S} > 0$, to satisfy Eq. (20) 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

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

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 $\alpha_k^* = \overline{\alpha}$ for some k, we have $\lambda_k = 0$ and

$$-\frac{1}{\overline{\alpha}} + \beta_k + \sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I = 0.$$

To satisfy the equation, it is obvious that $0 \leq \beta_k \leq \frac{1}{\overline{\alpha}}$. If $\beta_k = \frac{1}{\overline{\alpha}}$, then for any server $n \in \{n|X_{k,n}^i \neq 0, \exists i \in \mathcal{T}^k\}$, we have $\mu_n^E = 0$ and $\mu_n^I = 0$, thus

$$\sum_k \alpha_k^* b_{k,n}^E - B_n^E < 0 \quad \text{and} \quad \sum_k \alpha_k^* b_{k,n}^I - B_n^I < 0,$$

which implies that at all the servers where some tasks of application k are placed, the bandwidth is more than sufficient for it to achieve the maximum performance $\overline{\alpha}$.

3) If $w_k^{'}S < \alpha_k^* < \overline{\alpha}$, then we have $\lambda_k = 0, \beta_k = 0$, and

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

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. This explains why the performance of application k does not achieve the maximum.

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

$$\alpha_k^* = \frac{1}{\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I}. \tag{21}$$

The optimal solution can be obtained by solving the equations of KKT conditions in a centralized manner. The centralized solver need 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 $2(K+N)$ 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.

V. Distributed Algorithm for Bandwidth Allocation

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.

A. Dual Based Decomposition

Let p^* represent the optimal value of the primal optimization problem (19). Considering the general situation that $w_k^{'}S < \alpha_k < \overline{\alpha}$, we use $\mathcal{X} \subset \mathcal{R}^K$ to denote the solution space defined by constraints (15, 16). The Lagrangian of the primal problem under the general case is defined as $\mathcal{L}(\cdot) : \mathcal{X} \times \mathcal{R}^N \times \mathcal{R}^N \to \mathcal{R}$.

$$\mathcal{L}(\alpha, \mu^E, \mu^I) = -\sum_k \log \alpha_k + \sum_n \mu_n^E (\sum_k \alpha_k b_{k,n}^E - B_n^E) + \sum_n \mu_n^I (\sum_k \alpha_k b_{k,n}^I - B_n^I) \tag{22}$$

where $\mu^E = (\mu_1^E, ..., \mu_N^E)$ and $\mu^I = (\mu_1^I, ..., \mu_N^I)$ are the dual variables associated with constraints (17, 18). The Lagrange dual function $g(\cdot) : \mathcal{R}^N \times \mathcal{R}^N \to \mathcal{R}$ is defined as the minimum value of the Lagrangian $\mathcal{L}(\alpha, \mu^E, \mu^I)$ over α:

$$g(\mu^E, \mu^I) = \min_{\alpha} \mathcal{L}(\alpha, \mu^E, \mu^I) = \mathcal{L}(\alpha^*, \mu^E, \mu^I) \tag{23}$$

Thus, the dual problem of the optimization problem (19) is:

$$d^* = \max_{\mu^E \in \mathcal{R}_+^N, \mu^I \in \mathcal{R}_+^N} \quad g(\mu^E, \mu^I) \tag{24}$$

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 [10]. 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 (24) that has zero duality gap. Further, without coupling constraints, problem (24) can be divided into N subproblems to be solved at each physical server.

B. 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.

Let $\zeta > 0$ denote the step-size of the following recursion scheme of dimension N. For each $n \in \{1, 2, ..., N\}$ and $t \geq 0$, we have:

$$\begin{aligned} \mu_n^{(t+1)} &= \Big[\mu_n^{(t)} + \zeta \frac{\partial g}{\partial \mu_n}\Big]^+, \\ &= \Big[\mu_n^{(t)} + \zeta\Big(\sum_k \alpha_k b_{k,n} - B_n\Big)\Big]^+ \end{aligned} \tag{25}$$

where $[x]^+ = \max\{0, x\}$, 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 $\tilde{K} = \sqrt{2N}\Big(\sum_n \sum_k \big(b_{k,n}^E + b_{k,n}^I\big) C_k\Big)\overline{\alpha}^2$ and $C_k = \max_n \{b_{k,n}^E, b_{k,n}^I\}$, the recursive sequence $\{\mu^{(t)}\}$ generated by Eq. (25) 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 $\theta_k(\cdot)$ over $\mathcal{R}$ as $\theta_k(x) = \frac{1}{x}$, where $x \geq 1/\overline{\alpha}$. The Lipschitz constant of $\theta_k(x)$ is easily obtained as $\overline{\alpha}^2$. We also define $d_n^E(\mu)$ as the partial derivative of $g(\mu^E, \mu^I)$ with respect to μ_n^E:

$$d_n^E(\mu) = \frac{\partial g(\mu^E, \mu^I)}{\partial \mu_n^E} = \sum_k \alpha_k^* b_{k,n}^E - B_n^E \tag{26}$$

Based on Eq. (21), we can represent α_k^* as a $\theta_k(\cdot)$ function:

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

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

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

Then we can derive the following:

$$\begin{aligned} & |d_n^E(\mu) - d_n^E(\mu')| \\ \leq\ & \textstyle\sum_k b_{k,n}^E |\theta_k(\sum_n \mu_n^E b_{k,n}^E + \sum_n \mu_n^I b_{k,n}^I) \\ -\ & \theta_k(\textstyle\sum_n \mu_n'^E b_{k,n}^E + \sum_n \mu_n'^I b_{k,n}^I)| \\ \leq\ & \textstyle\sum_k b_{k,n}^E \overline{\alpha}^2 \sum_n (b_{k,n}^E |\mu_n^E - \mu_n'^E| + b_{k,n}^I |\mu_n^I - \mu_n'^I|) \\ \leq\ & \textstyle\sum_k b_{k,n}^E \overline{\alpha}^2 C_k \sum_n (|\mu_n^E - \mu_n'^E| + |\mu_n^I - \mu_n'^I|) \\ =\ & (\textstyle\sum_k b_{k,n}^E C_k) \overline{\alpha}^2 \|\mu - \mu'\|_1 \end{aligned} \tag{27}$$

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

$$|d_n^I(\mu) - d_n^I(\mu')| \leq (\textstyle\sum_k b_{k,n}^I C_k) \overline{\alpha}^2 \|\mu - \mu'\|_1 \tag{28}$$

Summing up Eq. (27) and Eq. (28) over all $n \in \mathcal{N}$ yields:

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

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:

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

Therefore, the Lipschitz constant of $\nabla g(\mu)$ is $\tilde{K} = \sqrt{2N} \Big(\sum_n \sum_k \big(b_{k,n}^E + b_{k,n}^I \big) C_k \Big) \overline{\alpha}^2$. According to the proof in [11], 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 $t \to \infty$. ■

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 1 for distributed bandwidth allocation, which can be implemented at each physical machine in parallel. The only required state information to be maintained at each machine is the weights and network load matrices of the applications that have their tasks placed on this machine. The network load matrics can be readily measured using standard tools, such as Bandwidth Monitor NG (`bwm-ng`).

Algorithm 1: The distributed algorithm for performance-centric bandwidth allocation.

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: $S \in (0, S^*]$;
The maximum performance: $\overline{\alpha}$;
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 Eq. (11) and (12);
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));
    end for
    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});
        α_k = α_k > ᾱ ? ᾱ : (α_k < w_k' S ? w_k' S : α_k);
    end for
    iterations ++;
end while
for all r^{i,j}_k do
    r^{i,j}_k = α_k · D^{i,j}_k;
end for
```

In each iteration, machine n updates μ_n^E and μ_n^I following Eq. (25), given the step-size ζ bounded by the global constant $2/\tilde{K}$ and the gradient of the dual function in Eq. (26). The performance α_k of each application k is computed given the updated dual variables according to Eq. (21). If the gradient 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. 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 computed performance is beyond its lower bound or upper bound, it will be set as its nearest 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.

VI. Performance Evaluation

In this section, we investigate how the proposed bandwidth allocation algorithm performs in tuning the tradeoff between

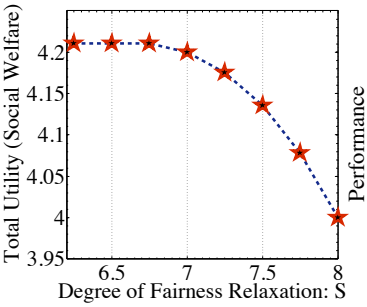

Fig. 5: The tradeoff between the total performance-centric fair share and the total utility in scenario 1.

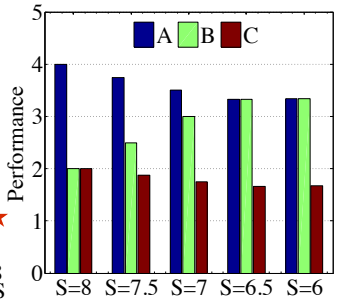

Fig. 6: Performance of application A, B and C when S is tuned at different values in scenario 1.

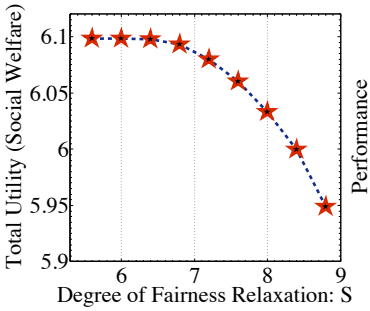

Fig. 7: The tradeoff between the total performance-centric fair share and the total utility in scenario 2.

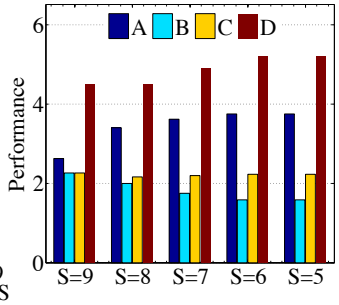

Fig. 8: Performance of application A, B C and D when S is tuned at different values in scenario 2.

maximizing the total utility and maintaining performance-centric fairness, with a detailed analysis in two typical scenarios. We also evaluate the performance of our algorithm with respect to the convergence.

We simulate a private datacenter with homogeneous physical machines. Each application has multiple tasks placed across several machines. Each machine is able to shape the bandwidth of its flows. We evaluate the behaviour of our algorithm under two scenarios shown in Fig. 9.

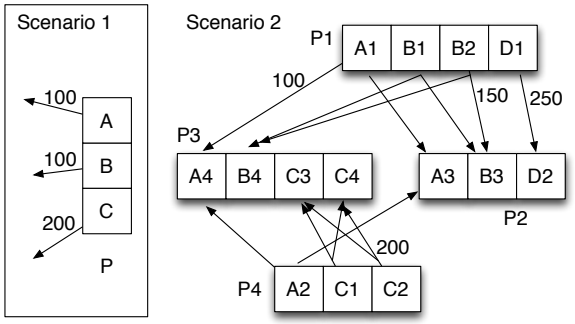

Fig. 9: Two scenarios for evaluating the proposed bandwidth allocation.

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. (13), the maximum total performance-centric share (S^*) of all the applications is computed as 8.

As shown in Fig. 5, 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. 6 delves into several points of the tradeoff curve to show the performance achieved by these applications.

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 less 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. 6. 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. 6, 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 below 6.5, the total utility will achieve its maximum and will not increase any further.

Now we consider Scenario 2 as shown in Fig. 9, where four applications A, B, C and D with the same weight have their tasks placed across 4 servers, each with the bandwidth capacity of 3 GB/s. The bottleneck link encountered by A, B and C is at $P3$, while D is bottlenecked at $P1$. 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. 7.

Fig. 8 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, in order to improve the overall utility.

Finally, we evaluate the convergence of our algorithm in Scenario 1. As shown in Fig. 10, 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.

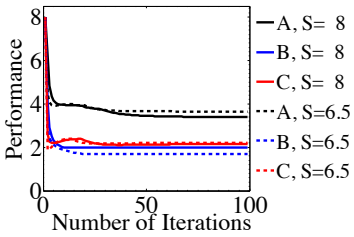

Fig. 10: Convergence of performance of A, B and C when S is 8 and 6, respectively, in Scenario 1.

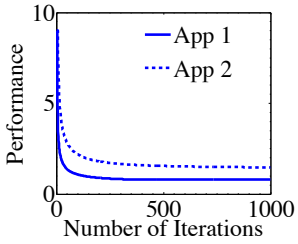

Fig. 11: Convergence of performance of two applications in a large scale private datacenter.

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 at different machines. Tasks of applications have different amounts (randomly chosen between 100 MB to 200 MB for simplicity) of data to be transferred in their communication stages. Fig. 11 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.

VII. Related Work

Bandwidth allocation among multiple tenants in public cloud datacenters have received a substantial amount of recent research attention [5]–[7], [12]–[16]. The general focus of these works has been on ensuring fair allocation among different tenants according to their payments. For example, NetShare [7] achieves tenant level fairness while Seawall [6] achieves fairness between VM sources. FairCloud [5] 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 [17]. 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 fills this gap, and offers a definitive guide to the problem of bandwidth allocation among multiple data parallel applications in a private datacenter. Moreover, having investigated the tradeoff between the total utility and performance-centric fairness, we design a new bandwidth allocation algorithm to maximize the total utility given tunable degrees of fairness relaxation.

VIII. 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 study the problem of performance maximization while maintaining fairness. Then we point out that there is an inherent conflict between maximizing bandwidth utilization and maintaining strict fairness. Considering this tradeoff, we introduce tunable degrees of relaxation on performance-centric fairness, and formulate the problem of maximizing the total utility across all applications with such a fairness relaxation. A distributed algorithm for bandwidth allocation is designed to solve the problem, which can be implemented on each physical machine in a lightweight fashion. Our extensive simulations have shown that our algorithm not only converges, but also provides the flexibility to balance the tradeoff between the total utility and the degree of fairness relaxation.

References

[1] A. Langville and C. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University 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*. Springer, 2008, pp. 337–348.
[3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the 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 Systems Review*, vol. 41, no. 3, pp. 59–72, 2007.
[5] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "FairCloud: Sharing the Network in Cloud Computing," in *Proc. ACM SIGCOMM*, 2012.
[6] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the Data Center Network," in *Proc. USENIX Networked Systems Design and Implementation (NSDI)*, 2011.
[7] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese, "NetShare: Virtualizing Data Center Networks across Services," UCSD-CSE, Tech. Rep. CS2010-0957, May 2010.
[8] 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*, 2011.
[9] 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*, 2009.
[10] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[11] H. Yaïche, R. Mazumdar, and C. Rosenberg, "A Game Theoretic Framework for Bandwidth Allocation and Pricing in Broadband Networks," *IEEE/ACM Trans. Networking*, vol. 8, no. 5, pp. 667–678, 2000.
[12] 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.
[13] 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 ICNP*, 2013.
[14] 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 Systems Design and Implementation (NSDI)*, 2013.
[15] H. Ballani, K. Jang, T. Karagiannis, C. Kim, D. Gunawardena, and G. OShea, "Chatty Tenants and the Cloud Network Sharing Problem," in *Proc. USENIX Networked Systems Design and Implementation (NSDI)*, 2013.
[16] L. Popa, P. Yalagandula, S. Banerjee, and J. Mogul, "ElasticSwitch: Practical Work-Conserving Bandwidth Guarantees for Cloud Computing," in *Proc. ACM SIGCOMM*, 2013.
[17] G. Kumar, M. Chowdhury, S. Ratnasamy, and I. Stoica, "A Case for Performance-Centric Network Allocation," in *Proc. ACM SIGCOMM HotCloud Workshop*, 2012.