Joint Coflow Routing and Scheduling in Leaf-Spine Data Centers

Yang Chen and Jie Wu

Center for Networked Computing, Temple University, USA

Email: {yang.chen, jiewu}@temple.edu

Abstract

Communication in data centers often involves many parallel flows that all share the same performance goal (e.g. to minimize the average completion time). A useful abstraction, *coflow*, is proposed to express the communication requirements of prevalent data parallel paradigms such as MapReduce and Spark. The multiple coflow routing and scheduling problem makes it challenging to derive a good theoretical performance ratio, as coexisting coflows may compete for the same network resources such as link bandwidths. In this paper, we focus on the coflow problem in one popular data center infrastructure: the Leaf-Spine topology. We first formulate the problem and study the path selection issue on this two-tier structure. In order to minimize the average coflow completion time (CCT), we propose the Multi-hop Coflow Routing and Scheduling strategy (MCRS) and prove that our method has a reasonably good competitive ratio. Extensive experiments and large-scale simulations show that MCRS outperforms the state-of-the-art heuristic schemes under the Leaf-Spine topology.

Keywords: Data centers, Leaf-Spine, coflow, routing, scheduling.

1. Introduction

With the explosive growth of data-parallel computation frameworks such as MapReduce [1], Spark [2], Google Dataflow [3], etc., modern data centers are able to process large-scale data sets at an unprecedented speed. In these frameworks, data flows for one job may share a common performance, utilization, or isolation goal [4]. In other words, applications do not care about an individual flow's behavior in their completion time or the fair sharing among

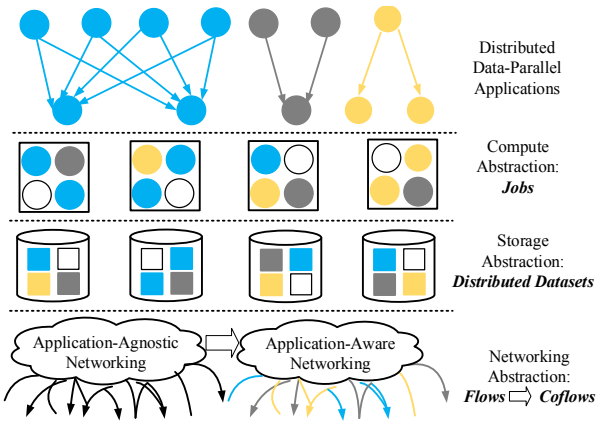


Figure 1: Coflow abstraction.

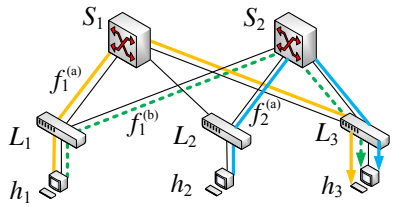
them, as the last flow of an application dictates the job completion. An abstraction, coflow [5; 6], is used to model such application-level information scenarios. A coflow is defined as a collection of parallel flows with a common performance goal [7], shown in Fig. 1. Similar to compute and storage abstractions [8], coflows expose application-level relationships to the network. In contrast, traditional point-to-point flows from different applications are indistinguishable [9]. The all-or-nothing observation has been captured for other resource types in data-parallel clusters, e.g., a job represents a set of tasks and a distributed file or dataset represents a collection of on-disk or in-memory data blocks are leveraged in almost all aspects of building systems, improving performance, scheduling resources, and providing fault tolerance [10; 11; 12]. What makes coflows unique and more general is the coupled nature of the network, unlike independent resources such as CPU speed or disk/memory capacity. One must consider both senders and receivers to allocate network resources. With the rise and development of Software Defined Networks (SDNs) [13], the requirements of high performance networks are becoming more and more intense. For example, in the aspect of the packet loss rate, data centers usually claim it to be around 2% [14], while the requirements of Wide Area Networks (WANs) and carrier-grade networks are much higher [15]. Specifically, the carrier-grade performance is often associated with the term, five-nines, representing an availability of 99.999%. As a result, the requirement of routing and scheduling coflows becomes more and more intense.

In this paper, we aim to minimize the average coflow completion time

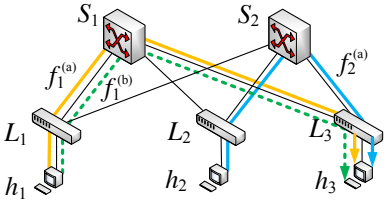
(CCT). The start time of a coflow is defined as the earliest arrival time of all flows within the coflow. Our network topology is a typical two-tier Clos topology consisting of only two switch layers: the spine switch tier and the leaf switch tier. This network structure is called Leaf-Spine. The reasons for applying Leaf-Spine are introduced in the forth and fifth sections. Additionally, we only route the flows within the coflows once in order to avoid the unnecessary packet reorder cost and the chaos of deleting and inserting new forwarding rules in the forwarding tables [16]. Meanwhile, we rescale the assigned bandwidth in order to further improve network utilization only when some flows finish their transmissions.

Take Fig. 2 as an illustrating example. In this example, there are two coflows: coflow a has two flows, f_1^a and f_2^a, with the size of 1 Mb and 3 Mb respectively; coflow b has only one flow, f_1^b, with a size of 5 Mb. They are generated at the same time and all the link bandwidths have a 1 Mbps capacity. As a reference point, the optimal average CCT of this example is 4.5 s. The optimal routing and scheduling patterns are shown in Fig. 2(c) and Fig. 2(f). The routing strategy tends to balance the traffic loads and the scheduling method is the Minimum Remaining Time First strategy.

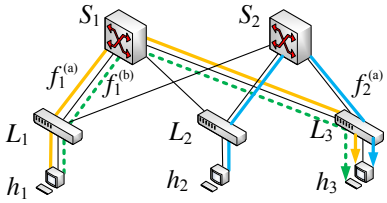
When paths of all flows are fixed, excellent scheduling strategies can minimize the average CCT based on the given routing by determining the sequence and bandwidth of flows to send out traffic. However, taking no consideration of routing cannot optimize the average CCT. We use Fig. 2(a) and (c) to illustrate it. Fig. 2(a) shows a possible case of routing. With a naive scheduling such as fair sharing, both coflow a' and b's CCTs are 8s. Obviously, this is not good enough scheduling, which indicates that scheduling also plays an important role in minimizing the CCT. Fig. 2(c) shows the optimal scheduling approach based on the given routing case. The average CCT is 5.5s, which still has a 1s gap to the optimal value. This indicates that a proper routing strategy is also necessary for a better performance in minimizing the average CCT. Additionally, separately arranging inter and intra coflows cannot optimize the average CCT, either. Fig. 2(b) and Fig. 2(d) show the results when we apply the separate inter and intra coflow routing and scheduling strategy from [17]. This strategy first optimizes a single coflow's CCT when it occupies the network resource, then, it proportionally scales down the bandwidth according to different coflows' optimal CCT values. With this method, all the coflows tend to be finished at the same time. In our example, its average CCT is 6s, which has a 1.5s gap to the optimal value. This example motivates our design of jointly routing and scheduling



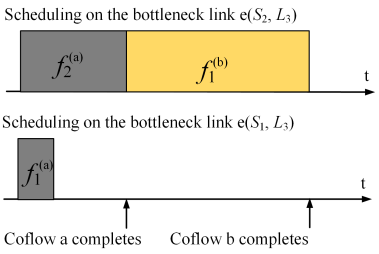
(a) A possible routing case.



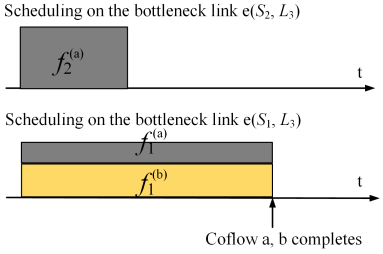
(b) Optimal routing.



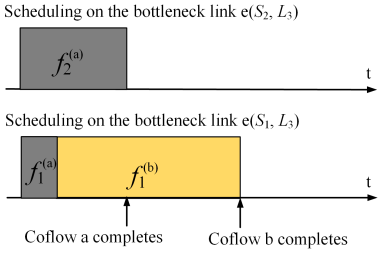
(c) Optimal routing.



(d) Optimal scheduling up on (a).



(e) A fair scheduling up on (b).



(f) Optimal scheduling up on (c).

Figure 2: A motivating example, where Figs. (a)-(c) show different routing schemes, and Figs. (d)-(f) show the executing flow scheduling for Figs. (a)-(c).

multiple coflows. We made three novel contributions in this paper.

First, instead of routing in only one-hop paths, we allow for two-hop paths and give a blocking probability comparison analysis, which can further improve network bandwidth utilization [18].

Second, we present both inter-coflow and intra-coflow algorithms based on two crucial observations. Our objective is to minimize the average CCT under the special Leaf-Spine topology. A sound competitive performance ratio is given with a detailed proof. What's more, in order to further reduce the average CCT, we apply the work conservation method to coflows when there are available leisure bandwidth resources in the network.

Our third contribution is a comprehensive performance evaluation of our algorithms by simulations and experiments. We have built a prototype to validate the solutions in a real testbed, and we have also conducted simulations to obtain performance result with the real data trace.

The remainder of the paper is organized as follows: Section 2 surveys the related work. Section 3 states the motivation and the problem formulation. Section 4 studies the path blocking probability and introduces the alternative options for coflow routing paths. Section 5 focuses on the intra and inter coflow routing and scheduling solutions over key observations. We also provide a theoretical performance analysis of our scheme and prove it has a reasonably competitive ratio. Sections 6 and 7 include the experiments and simulations, respectively. The paper concludes in Section 8.

2. Related Work

Though coflow management is a relatively novel topic, a growing body of recent work [19; 20; 21; 17] has demonstrated that using coflows can significantly improve the communication performance of distributed data-parallel applications [22]. The coflow abstraction is first proposed as "a networking abstraction to express the communication requirements of prevalent data parallel programming paradigms" [6]. Coflows are studied through the communication requirements of diverse cluster computing applications and are proposed to use grouped data flows as the network model for data parallel jobs. Then two solutions, Varys [20] and Aalo [21], both employ the coflow concept to develop flow scheduling solutions to minimize the coflow completion time. Specifically, Varys optimizes the completion time given the size of each flow of all coflows, while Aalo assumes there is no prior knowledge

about the coflow sizes. However, neither of them provides a performance-guaranteed solution for multiple coflows. Specifically, Varys focuses on the single coflow routing and scheduling, while Aalo introduces only heuristic solutions.

Originally, some papers [20; 21; 23; 24] study coflow scheduling only. For example, Varys [20] proposes heuristic scheduling algorithms to minimize the average CCT and meet coflow deadlines. Aalo [21] divides coflows into various priorities using the Coflow-Aware Lest-Attained Service (D-CLAS) algorithm based on the data amount they have already sent. Barrat [25] also brings task-awareness into network optimization. Unlike the above centralized works, Barrat [25] proposes a distributed algorithm for task-aware flow scheduling. Qiu et al. [23] make a contribution to the first polynomial-time deterministic approximation algorithm for the multiple coflows scheduling problem. CODA [24] is the pioneer in detecting coflows in the individual flows with the help of machine learning techniques.

Rapier [19] proves that scheduling-only coflow strategies cannot optimize the application-level performance. Rapier takes both coflow routing and scheduling into consideration, and advances a heuristic solution for a single coflow with an approximation ratio. However, the theoretical bound is loose and they didn't include any performance-guaranteed solution for multiple coflows. When it comes to flow scheduling, before coflow abstraction is proposed, there are numerous scheduling methods for minimizing the average flow completion time (FCT) and meeting flow deadlines. DCTCP [26], D2TCP [27], and L2DCT [28] focus on improving flow completion time by modifying the default behavior of the TCP at end-hosts, and thus do not require a modification of switches or host hardwares. They rely on Explicit Congestion Notification (ECN) for congestion notification. PDQ [29] and pFabric [30] tag priorities on the packets in order to minimize the FCT and reduce congestion, but need to modify their hardware. Yu et al. [17] propose a rounding-based randomized approximation algorithm to minimize the average CCT. Its approximation ratio has a probability constraint. However, the time complexity of their proposed solution is easily out of control because of solving a linear programming problem. Additionally, its approximation ratio is linearly proportional to the number of coflows, which could be relatively large when a large amount of coflows are included. [31] proposes a near-optimal network design for coflows that can be implemented on top on any transport layer (for flows) that supports priority scheduling. However, the priority of each coflow as well as the ordering of all flows belonging to one

coflow needs to be known in advance. We propose a performance-guaranteed solution and prove it has a better approximation ratio in a special network topology, compared to [17]. The single coflow routing and scheduling problem for minimizing the CCT has been proven NP-hard [32]. With multiple coflows, the inter-coflows' and intra-coflows' paths will overlap and flows will compete for the same link resources. In addition, cluster computing frameworks are dynamic in providing enough prior knowledge, which demands an online approach. As a result, we reduce the problem to a specific data center topology, the two-tier Leaf-Spine structure. Additionally, we generate a performance-guaranteed solution for multiple coflows with intuitive insights.

3. Framework

3.1. Motivation and Problem

This paper studies the multiple coflows' online routing and scheduling problem in order to minimize their average CCTs, which has been a popular objective in several works [29; 30; 20; 21]. The coflow routing and scheduling problem is challenging and current methods do not get good enough theoretical results. It has been proven that even a single coflow trying to minimize the CCT is NP-hard [32], not to mention multiple coflows. This conceptual difficulty of the NP-hard problem arises from combining packing constraints due to the existence of capacities with the path selection under an arbitrary topology. What's more, the multiple coexisting inter-coflows and intra-coflows will overlap on paths with others, which will result in a competition for the same network resources, such as link bandwidth. We are motivated by the fact that none of a single routing, or a single scheduling, or a separate inter- and intra- coflow routing-and-scheduling strategy is sufficient enough to optimize the average CCT. Furthermore, in this paper we focus on a special and simple two-tier Clos topology, the Leaf-Spine structure [33; 34], which is quite popular in today's data centers. We aim to propose performance-guaranteed routing and scheduling strategies in the Leaf-Spine topology.

We use Fig. 3 to further explain the definition of the coflow and some key observations. In this example, we assume at time t, there are only two coflows: coflows a and b. Coflow a in the yellow solid line has two flows, $f_1^{(a)}$ from h_1 to h_3 and $f_2^{(a)}$ from h_2 to h_3, both with a workload size of $1Mb$; coflow b in the dotted blue line has two flows, $f_1^{(b)}$ from h_1 to h_2 and $f_2^{(b)}$ from h_2 to h_3, both with a size of $3Mb$. They are generated at the same

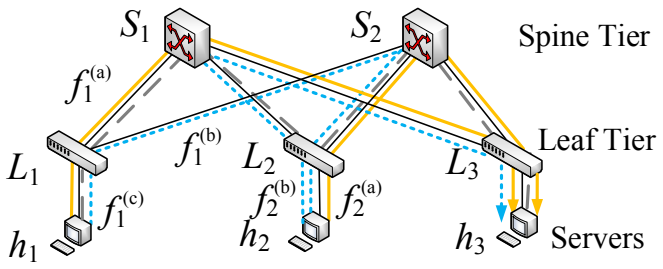


Figure 3: Coflow example.

time and all the links have a $1Mbps$ bandwidth. A possible path assignment plan is shown in Fig. 3. The flows' paths are all one-hop paths, which pass through only one spine switch. The optimal scheduling plan arranges both flows of coflow a with a bandwidth $0.25Mbps$ and both flows of coflow b with a bandwidth of $0.75Mbps$. After $1s$, a new coflow c including only one flow $f_1^{(c)}$ is generated from the source h_1 to the destination h_3. All the one-hop paths from h_1 to h_3 are busy and the biggest available bandwidth for all the paths is $0.25Mbps$, which passes by L_1, S_2, and L_3. Under this circumstance, the optimal solution's average CCT for these three coflows is $4.75s$. However, we notice that there is a detour two-hop path with an available bandwidth of $0.75Mbps$, whose path is the grey line through L_1, S_1, L_2, S_2, and L_3. Applying this path to $f_1^{(c)}$, we get that the average CCT is $4.25s$, which is $0.5s$ shorter than the former one-hop path. This result inspires us not to confine to only one-hop paths in the Leaf-Spine network. It is likely to obtain a better average CCT by trading with a longer path.

There are two key observations in our strategy as follows:

Observation 1. *Inter-coflow scheduling should apply the Minimum Remaining Time First strategy.*

This observation is inspired by the famous optimal job scheduling approach: Smallest Remaining Time First (SRTF) [35]. We are given a set of jobs with different processing times. In order to minimize the total completion time, SRTF selects the process with the smallest amount of time remaining until all jobs are finished. It's a preemptive online method. In this case, even if a coflow is occupying the bandwidth in the network, it will be preempted by the new-coming 'smaller' coflows. It is worth mentioning that our objective can be extended to minimizing the weighted average CCT. Its corresponding optimal scheduling approach is Weighted Smallest

Remaining Time First (WSRTF)

Observation 2. *Big intra-coflows should be scheduled first when we have an idle bandwidth.*

It is intuitive that it is always the big flows in a single coflow that are the last ones to finish their transmissions. As a result, in order to shorten the average CCT, if there is leisure bandwidth available for multiple flows, we should select the flows with a larger workload to execute first.

3.2. Model and Formulation

A Leaf-Spine network is modeled as a directed graph, $G = (V, E)$, where E is the edge set and V is the node set. The network size is denoted as $n = |V|$. In data center networks (DCNs), each node, $v \in V$, can be a server or a switch. Each edge, $e \in E$, has a capacity of R_e. There are two layers of switches: leaf switches and spine switches. A series of leaf switches, L, form the access layer. These switches are fully meshed to a series of spine switches, S. A coflow is a collection of related parallel flows with a common performance goal (e.g. to minimize the average CCT in this paper). Assume there are totally m coflows during the whole process. Here we need to mention that some flows may have a different performance evaluation metric, for example, delay-sensitive flows with a high priority. In this paper, we do not include them in the input of our problem. Usually, the network will reserve some bandwidth for these special flows [36]. Denote the i_{th} coflow as C_i $(1 \leq i \leq m)$, which arrives at time T_i and contains w_i individual flows. Here we define the arrival time T_i of the coflow C_i as the earliest arrival time of all individual flows within the coflow C_i. A flow j $(1 \leq j \leq w_i)$ within a coflow C_i is defined by a 3-tuple $(s_j, d_j, v_j) \in C_i$, where s_j and $d_j \in V$ are the source and destination nodes, and $v_j > 0$ is the flow volume. The available path set and the bandwidth for flow j in coflow C_i are denoted as $P_j^{(i)}$ and $b_j^{(i)}$, respectively. For the ease of reference, we summarize notations in Tab. 1.

Without loss of generality, we assume that a coflow C_i has all the information about its flows and starts to transmit when it arrives at the network at time T_i, which is similar to [17]. At time $x \geq T_i$, a flow $(s_j, d_j, v_j) \in C_i$ is then forced to be routed on $p_j^{(i)}$ with a rate of $b_j^{(i)}$. Note that $b_j^{(i)}$ can be zero for some time x's, which means that this flow is waiting for transmission. Because we relax our problem on the special Leaf-Spine topology, the path

Table 1: Symbols and Definitions.

Symbols	**Definitions**
V, E, L, S, C	set of nodes, edges, leaf, spine switches and coflows
m, n	number of coflows and nodes
v, e, C_i	a node, an edge, and the i_{th} coflow
T_i, w_i	arrival time and number of flows in coflow C_i
s_j, d_j, v_j	source, destination, and volume of flow j
$P_j^{(i)}, b_j^{(i)}$	path set and bandwidth of flow j in coflow C_i

for each flow is simplified. In our paper, we only allow one-hop and two-hop paths, which limits the path length to be less than 4 units (1 unit is one link). A one-hop path passes only one spine switch during the flow's transmission, and so on. The reason is introduced in the next section. Furthermore, the number of paths between any pair of nodes is bounded by $poly(n)$, because there are $|S|$ one-hop paths and $|S|^2 \times (|L| - 2)$ two-hop paths and we have $O(|S| + |S|^2 \times (|L| - 2)) = O(|S|^2 \times (|L| - 2)) = O(n^3) = O(poly(n))$. A routing and scheduling strategy for a coflow C_i is defined as

$$S_i := \{p_j^{(i)}(x),\ b_j^{(i)}(x)\}_{j=1}^{w_i} \tag{1}$$

A time-slotted system is considered. We then define the CCT t_i for coflow C_i to be the minimum time such that

$$\sum_{x=T_i}^{T_i+t_i} b_j^{(i)}(x) \geq v_j^{(i)}, for\ all\ 1 \leq j \leq w_i \tag{2}$$

which is the earliest time that all the flows in C_i finish transmitting their data. We further define the total CCT for all the coflows as

$$t = \sum_{i=1}^{m} t_i$$

Since frequent flow rerouting will cause a severe coordination overhead, which is not desirable in practice, in our paper, each flow's path can only be decided once. A valid strategy guarantees that each link's bandwidth is no less than the sum of the assigned bandwidth for all the flows. Information about the future coflows is not known. With the above settings, we define the online multiple coflow routing and scheduling problem as follows:

Problem 1. *In a network, m coflows* $C_1,\ C_2, ...,\ C_m$ *arrive at time* $T_1,\ T_2, ...,\ T_m$*. The information of every coflow* $C_i := \{(s_j, d_j, v_j)\}_{j=1}^{w_i}$ *is given at its arrival, which includes the corresponding source-destination pair, and its volume. The available path set for the flow* $j \in C_i$ *is* P_j^i *consisting of its one-hop and two-hop paths. The problem is designing an algorithm to find a valid routing and scheduling strategy,* $\{S_i\}_{i=1}^m$*, for each coflow so that the average completion time of the coflows,* $\frac{t}{m}$ *, is minimized.*

4. Routing Path Block Probability Analysis

In our paper, we assume that the routing path is either one-hop or two-hops long under the Leaf-Spine topology. It was inspired by a recently proposed distributed routing mechanism for Leaf-Spine topology, called CONGA [36]. It schedules paths based on a real-time fabric congestion situation, which is obtained through feedbacks from the remote switches. CONGA limits packets to a one-hop routing path. However, traffic is likely to be heavily unbalanced in data centers because of bursts in just a few congested links, shown as the grey dotted lines in Fig. 4. Other colored lines are the current idle links. In this example, when Server a wants to transfer a flow to Server b, the blue one-hop path is chosen. But if the destination is Server c, there are no available congestion-free one-hop paths, and CONGA will suffer severe transmission delays. However, we notice that a two-hop yellow line path [18], which detours at the second leaf switch, is able to avoid the congested links. Consequently, in order to further balance the loads, we provide an alternative to a two-hop routing path by leveraging relative leisure links to relieve local traffic pressure when all direct one-hop paths are heavily-loaded. The specific theoretical analysis is as follows:

Suppose there are $|S|$ spine tier switches and $|L|$ leaf tier switches. We use ρ_b and ρ_n to denote the blocking and non-blocking probability of its paths. $p(i,k)$ is the non-blocking probability of the link from switch i to switch k. As a result, the blocking probability of the one-hop path is calculated as

$$\rho_b(i,j) = \prod_{k \in S} [1 - p(i,k) \times p(k,j)] \tag{3}$$

So, the non-blocking possibility is

$$\rho_n(i,j) = 1 - \rho_b(i,j) = 1 - \prod_{k \in S} [1 - p(i,k) \times p(k,j)] \tag{4}$$

Suppose L' is the set of all the leaf switches except the flow's source switch i and destination switch j. The non-blocking probability of our path set is calculated as

$$\rho'_n(i,j) = \rho_n(i,j) + \sum_{k,k' \in S, m \in L'} [p(i,k) \times p(k,m) \times p(m,k') \times p(k',j)]$$

The difference between $\rho'_n(i,j)$ and $\rho_n(i,j)$ is calculated as

$$\rho'_n(i,j) - \rho_n(i,j) = \sum_{k,k' \in S, m \in L'} [p(i,k) \times p(k,m) \times p(m,k') \times p(k',j)]$$

Intuitively, the bigger the non-blocking probabilities of the links from the source to the idle leaf switch and from the idle switch to the destination are, the larger the difference of $\rho'_n - \rho_n$ is. That means our strategy's advantage is more obvious. The traffic spike between one pair of nodes and an unbalanced bursty traffic condition can make our routing strategy superior [18].

Here it is worth to mention that multiple-hop (more than 2) paths can also be applied to route flows, especially when the traffic distribution is extremely unbalanced. However, a two-hop path uses double bandwidth resource of a one-hop path, a three-hop path uses triple and a k-hop path uses k times, indicating the extra bandwidth consumption increasing with the path length. Additionally, a longer path needs a longer transmission time and more forwarding entries in forwarding tables of the switches. However, Katta et al. [16] point out the insufficient memories of the forwarding tables. As a result, we only allow one-hop and two-hop paths in this paper. We also do the simulations to indicate the suitability of allowing no more than two-hop paths. Additionally, in order to improve the performance, the turning point from a one-hop path to a two-hop path is based on an empirical threshold ratio. When the completion time of a single flow in a one-hop path is larger than the threshold ratio times its completion time in a two-hop path, we apply the two-hop path; otherwise, we still use its one-hop path. This is because applying two-hop paths utilizes more bandwidth, which may degrade the network performance afterwards. Only when the two-hop paths can save more time, we use the two-hop paths. We test the turning point effect in our simulations as well.

5. Coflow Routing and Scheduling Strategy

We design our algorithms in order to satisfy several important properties introduced in [19]:

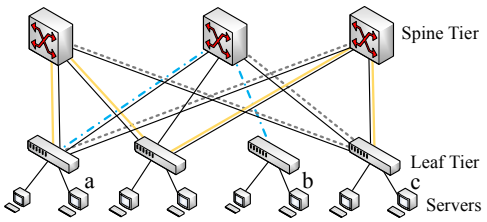


Figure 4: Unbalanced traffic situation.

- *Scalability*: Our scheme is necessarily an online system. Upon a new incoming coflow, the algorithms must be able to quickly and efficiently decide the routing paths, rates, and scheduling orders for all individual flows in the coflow. For this purpose, it must run in real-time with a low time complexity.

- *Starvation-free*: As our algorithm allows bandwidth preemption, we must ensure that any coflow should not starve for an arbitrarily long period, though this might worsen the average CCT in the network.

- *Work-conserving*: Work-conservation means that the network resource sits idle only if there is no traffic demand in the network. We require the algorithm to be work-conserving to fully utilize the network capacity and to minimize the average CCT.

- *Readily deployable*: The system should be readily implementable with existing commodity switches and easy to deploy without modifying any network devices.

Our following algorithm, Multi-hop Coflow Routing and Scheduling strategy (MCRS), optimizes the average CCT in the data-intensive Leaf-Spine topology DCNs by considering both the routing and scheduling of the coflows. In order to obtain scalability and ensure coexistence, MCRS mainly focuses on large coflows in the DCNs. We use a site broker to periodically measure the usage of background traffic in each link, and update the available bandwidth for the large coflows we need to schedule.

5.1. Single Coflow Routing and Scheduling

First, we consider the single coflow case: $C_i := \{(s_j^i, d_j^i, v_j^i)\}_{j=1}^{w_i}$. According to the above setting, for each flow $j \in C_i$, the source-destination pair and the volume are given. The path set P_j^i for each flow $j \in C_i$ only includes

one-hop and two-hop paths in order to limit the number of the paths for each flow within $O(n^3)$. The capacity for each edge e is R_e. Our goal is to find a valid strategy for C with the minimum average CCT when it monopolizes the network. We use Program A to describe the initial problem as follows:

$$\min \ t_i \tag{5}$$

subject to

$$v_j^i = b_j^i \times t_i, \qquad 1 \le j \le w_i \tag{6}$$

$$\sum_{j=1}^{w_i} \sum_{e \in p} b_j x_{j,p} \le R_e, \qquad e \in E \tag{7}$$

$$\sum_{p \in P_j^i} x_{j,p} = 1, \qquad 1 \le j \le w_i \tag{8}$$

$$x_{j,p} = \{0, 1\}, \qquad 1 \le j \le w_i \tag{9}$$

In this program, variable t_i denotes the CCT of C. Variable b_j^i denotes the average bandwidth of the j-th flow, and variable $x_{j,p}$ denotes whether or not we choose path p for the j-th flow, which has an integer value of 0 or 1. It is impractical to find the optimal solution of Program A, because it is not only nonlinear, but also has binary variables. This problem is an integer multi-commodity flow problem that is proven to be NP-hard [32]. Therefore, we do some equivalence transformations to Program A. Based on the first constraint, we know that the rate of each flow is directly proportional to its volume, i.e., $b_j^i = \alpha_i \times v_j^i$. As for the binary variable, we relax the binary constraint to a continuous one. Thus, we have $\alpha_i = \frac{1}{t_i}$ and Program A can be modified as:

$$\max \ \alpha_i \tag{10}$$

subject to

$$\sum_{j=1}^{w_i} \sum_{e \in p} \alpha_i v_j^i x_{j,p} \le R_e, \qquad e \in E \tag{11}$$

$$\sum_{p \in P_j^i} x_{j,p} = 1, \qquad 1 \le j \le w_i \tag{12}$$

$$0 \le x_{j,p} \le 1, \qquad 1 \le j \le w_i \tag{13}$$

Algorithm 1 The Intra-coflow Algorithm

In: Coflow $C_i = \{(s_j, d_j, v_j)_{j=1}^{w}\}$;
Out: t_i for C_i and $\{p_j^i, b_j^i\}$ for each flow belongs to C_i;

1: Calculate the optimal solution $\{y_{j,p},\ p \in P_j^i\}_{j=1}^{w}$ for the Linear Program B.
2: **for** each flow f_j in C_i **do**
3: Choose the one-hop or two-hop path $p \in P_j^i$ with $\max\{y_{j,p}\}$ as the route p_j^i for f_j.
4: Find the link e^* with the maximum $(\sum_{e \in p_j^i} v_j^i)/R_e$.
5: $t_i = (\sum_{e^* \in p_j^i} v_j^i)/R_{e^*}$.
6: **for** each flow f_i in C **do**
7: $b_j^i = v_j^i/t_i$.
8: **return** t_i and $\{p_j^i, b_j^i\}$.

However, the product of the two variables α_i and $\{x_{j,p}\}$ in its above first constraint makes the problem a concave problem that is hard to handle. So, we bring in new variables $\{y_{j,p}\}$ to replace the product, i.e., $y_{j,p} = \alpha_i x_{j,p}$. According to Eq. 14, we get $\alpha_i = \sum_{p \in P_j^i} y_{j,p} = \frac{1}{w_i} \sum_{j=1}^{w_i} \sum_{p \in P_j^i} y_{j,p}$. Consequently, our problem can be transformed to the linear Program B as follows:

$$\max \ \frac{1}{w_i} \sum_{j=1}^{w_i} \sum_{p \in P_j^i} y_{j,p} \tag{14}$$

subject to

$$\sum_{j=1}^{w_i} \sum_{e \in p} v_j y_{j,p} \leq R_e, \qquad e \in E \tag{15}$$

$$y_{j,p} \geq 0, \qquad 1 \leq j \leq w_i \tag{16}$$

Program B is a linear programming problem that has $\sum_{j=1}^{w_i} |P_j^i|$ variables and $|E| + \sum_{j=1}^{w_i} |P_j^i|$ constraints. The optimal fractional solutions, $\{y_{j,p}\}$, of the relaxed LP can be obtained in polynomial time using standard solvers.

We provide two algorithms for intra- and inter- coflows respectively in Alg. 1 and Alg. 2. The algorithms are invoked whenever a new coflow comes or an existing coflow finishes. More specifically, when a new coflow arrives,

Algorithm 2 Smallest Remaining Coflow First Inter-coflow Algorithm

In: All the coflows C's information $C_i := \{(s_j, d_j, v_j)\}_{j=1}^{w_i}$;
Out: The coflow executing order;

1: Use the Intra-coflow Algorithm to calculate all the remaining coflows' completion time t_i;
2: Sort these coflows' completion time t_i non-increasingly according to their transmission time;
3: **while** some coflows are not finished **do**
4: Apply the allocation to coflow with the smallest t_i.
5: Schedule biggest flows if there are remaining bandwidths.

Alg. 1 is used to compute the path and bandwidth arrangement for each single flow of it. Alg. 2 is used to sort the order of coflows that are waiting to be transmitted. When an existing coflow finishes, the bandwidth of all the links that are occupied by its flows will be released. As a result, we need to decide which coflows should take up the released bandwidth. The details of these two algorithms are as follows:

Alg. 1 is used to solve the single coflow routing and scheduling problem with a minimum CCT. It first solves the linear program B in Line 1. Then lines 2-3 schedule each flow to its one-hop path with the maximum bandwidth and calculate its completion time. If it is less than the current coflow completion time, we will try to check whether its two-hop paths have a path with a larger bandwidth. After choosing the path with the shortest completion time, we scale up all the arranged flows in order to make sure they have the same completion time in lines 4-5 and lines 6-7 allocate the bandwidths for each flow. Line 8 returns the completion time of the coflow C_i as well as the paths and allocated bandwidths for all flows of C_i.

Alg. 2 describes the inter-coflow arrangement method of MCRS. It is easy to understand. Line 1 calls Alg. 1 to calculate the completion time of each coflow. We sort the coflows in the order of their remaining completion times in line 2 and execute them one-by-one in order in lines 3-4. Line 4 improves the performance by utilizing the leisure bandwidth resources.

Preliminaries: When we say an algorithm for the problem is ρ-competitive, we mean that for all k ($1 \leq k \leq m$), the online algorithm returns a valid strategy for each coflow in the set $\{(C_i, T_i)\}_{i=1}^{k}$ that has a completion time of at most ρ times its minimum completion time under the offline setting. The

offline setting means that the information of the posterior coflows is known in advance (the knowledge of (C_i, T_i) for $j > i$).

Theorem 1. *The Alg. 1 is $|S|^2(|L|-2)$-competitive, where $|S|$ and $|L|$ are the numbers of spine and leaf switches.*

Proof: Suppose the solution of the Program B is $\{y'_{j,p}\}$. From Alg. 1, we route each flow to the path with the maximum $\{y_{j,p}\}$ and hence, we have $y_{j,p} \geq y'_{j,p}/|S|^2(|L|-2)$, where $|S|^2(|L|-2)$ is the maximum number of the candidate paths for the flow j. This is a loose bound, because in practice, the number of paths used in the optimal solution of Program B is much smaller than the maximum number of paths. ■

5.2. Multiple Coflow Routing and Scheduling

From Thereom 1, we know that the competitive ratio ρ of Alg. 1 executes a single coflow. Then, we apply Alg. 2 to multiple coflows. The underlying scheduling policy is based on the well-known minimum remaining first (MRTF).

Theorem 2. *Suppose the competitive ratio of Alg. 1 is ρ. Then, Alg. 2 is $(m+1)/2 \times \rho$-competitive for the online multiple coflow routing and scheduling problem.*

Proof: Suppose OPT is the offline minimum completion time for $\{(C_i, T_i)\}_{i=1}^m$ and t is the completion time of our strategy. Denote OPT_i as the optimal completion time for coflow C_i. After applying Alg. 2, the coflows are renumbered as $C'_1, C'_2, ..., C'_m$, whose completion time has the relationship of $\text{OPT}_1 \leq \text{OPT}_2 \leq ... \text{OPT}_m$. If our strategy is without work conservation, we only allow one coflow to transmit after the last coflow finishes, which means every coflow will monopolize the network. The time of this situation is t' and we have $t < t'$. Trivially, the completion time of the coflow i is

$$t_i = OPT_1 + OPT_2 + ... + OPT_i$$

Then, t' can be represented as

$$\begin{aligned} t' &= t_1 + t_2 + ... + t_m \\ &= (OPT_1) + (OPT_1 + OPT_2) + ...+ \\ &\quad (OPT_1 + OPT_2 + ... + OPT_i) \\ &= \sum_{i=1}^{m} (m - i + 1) \times OPT_i \end{aligned} \tag{17}$$

Because of $\mathrm{OPT}_1 \leq \mathrm{OPT}_2 \leq ... \ \mathrm{OPT}_m$, apply the Chebyshev's inequality, and we get

$$\begin{aligned}\sum_{i=1}^{m}(m-i+1)\times OPT_i &\leq \frac{1}{m}(\sum_{i=1}^{m}OPT_i)(\sum_{i=1}^{m}i) \\ &= \frac{1}{m}(\sum_{i=1}^{m}OPT_i)\frac{m(m+1)}{2} \\ &= \frac{(m+1)}{2}(\sum_{i=1}^{m}OPT_i)\end{aligned} \tag{18}$$

Thus, we have

$$t \leq t' \leq \frac{(m+1)}{2}(\sum_{i=1}^{m}OPT_i) \tag{19}$$

Therefore, our theorem is proven. ■

Theorem 2 shows that our MUSR can achieve a reasonable performance bound. Furthermore, the experiments in the next section show that our strategy has a more superior performance than the current methods. Here we need to mention that the approximation ratio of our proposed solution is still linear to the number of coflows, but it is almost half of that of the one in [17]. Additionally, with a special network topology, we can derive the approximation ratio more specifically. We can combine the results of Theorem 1 and Theorem 2 to get the following corollary.

Corollary 1. *Our algorithm is $|S|^2(|L|-2)(m+1)/2$-competitive, where m is the number of coflows.*

Proof: According to Theorem 1, $\rho = |S|^2(|L|-2)$. Then from Theorem 2, the approximation ratio of our algorithm is calculated as $(m+1)/2 \times \rho = (m+1)/2 \times |S|^2(|L|-2)$. Compared to the performance bound in [17], our algorithm MCRS has a much tighter competitive ratio. ■

5.3. Work-Conservation Improvement

From the above two algorithms, we know that each coflow will monopolize the network when it is transmitting. However, some idle bandwidth will be wasted, which should be used to execute more flows. We pursue the work-conserving property by distributing the remaining bandwidth to flows to further increase the overall system performance.

Figure 5: The front and back views of our testbed.

The challenge in distributing the remaining bandwidth is determining the preempting order of flows. At first, for the coflows that have already been scheduled, their CCTs cannot be improved by allocating more bandwidth to their intra-flows. Therefore, among all the coflows, those that have not been scheduled should have a higher priority in using the remaining bandwidth. This also helps prevent coflow starvation. Within a coflow, we prefer to allocate more bandwidth to the larger flows than to the smaller ones. This is because the flows with a larger traffic volume are more likely to be the bottleneck of a coflow, i.e., complete the transmission last if all the flows are served by a best-effort delivery. Based on this observation, when there is free bandwidth in the network, we allow the "elephant" flows to utilize the resources first.

6. Testbed Evaluation

We evaluate MCRS using testbed experiments in this section and using large-scale simulations in the next section.

6.1. Configurations

We do realistic transmission experiments on the testbed of our lab, whose front and back photos are shown in Figure 5. The testbed contains two Cisco switches (8 ports) and three Pica switches (48 ports). Each Pica switch connects with two 64-bit Dell Power Edge R210 servers. Each server has 2.4 GHz CPU and 4 GB memory and is accessible via the connections offered by the switches. *Grnlntrn* is the controller, which is constructed by a Dell

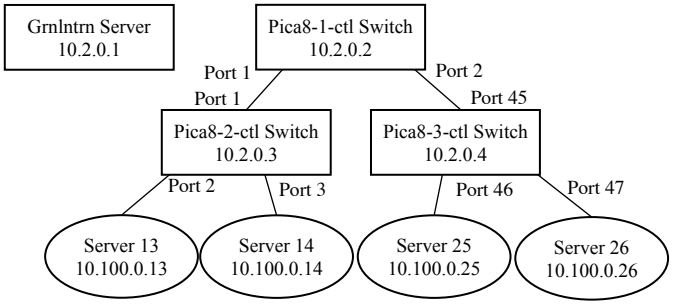
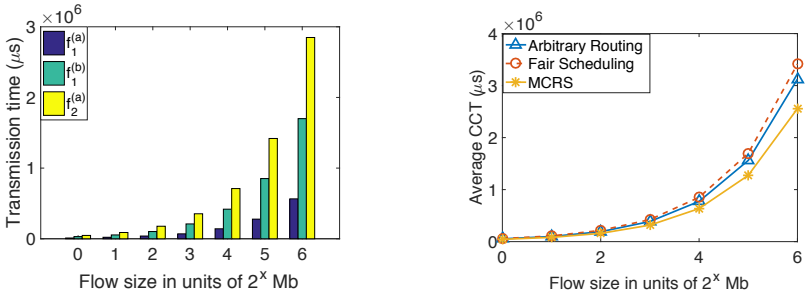


Figure 6: Topology of the test bed.

Power Edge R210 server. The capacity of all the physical links in our testbed is 1Gbps.

The topology we use is shown in Figure 6. We implement the configuration of our toy example in Fig. 2. Due to the number of hardwares in our lab, the testbed has a relatively small size. However, this paper tends to focus on proposing some theoretical results instead of just heuristic ones. We hope our performance-guaranteed solution in this paper can inspire more following works. We utilize the virtual port transmission function of the Pica8 SDN switch, which can serve as two independent switches when we generate two virtual ports. Additionally, the virtual port can limit the transmission rate. As a result, in Figure 6, Pica8-1 works as two spine switches in the example and Pica8-2 and Pica8-3 work as leaf switches, respectively. Thus, we are given two coflows: coflow a has two flows, f_1^a and f_2^a, with the size of xMb and $3xMb$ respectively; coflow b has only one flow, f_1^b, with a size of $5xMb$ ($x \in 2^n Mb, n \in \{0, 1, 2, 3, 4, 5, 6\}$). The source of f_1^a and f_1^b is Server 13, while the source of f_2^a is Server 14. The destination of all flows is Server 25. All flows are generated at the same time. There are two comparison transmission plans. The routing of the first comparison plan, shown in Fig. 2 (a), is an arbitrary routing while its scheduling is optimal corresponding to its routing, shown in Fig. 2 (c). The routing of the first comparison plan, shown in Fig. 2 (b), is an optimal routing while its scheduling is the fair allocation scheme, shown in Fig. 2 (d). We use the average CCT to evaluate the performance of our algorithm over the other two. The completion time of each flow is measured as the end-to-end delay of transmitting the flow load [37].



(a) Single flow transmission time. (b) A possible routing case.

Figure 7: Results of the testbed.

6.2. Evaluation Results

The results are shown in Fig. 7. Fig. 7(a) illustrates the transmission time of two coflows including three flows in total. The time is measured when there is only one flow transmitting through its path. It reveals the shortest completion time of each flow with full support of bandwidth. We test three flows with the traffic load starting at 1 Mb, 3 Mb and 5 Mb. Each of them doubles their loads in the next round of transmission. So we have tested 15 transmission times of distinct flows, shown in 7(a). The tendency of each color-bar is similarly increasing. For each color-bar, when the load of a flow doubles, its transmission time is also changed to nearly two times. For example, the average tested transmission times of a 10 Mb flow and a 20 Mb flow are 88570 microseconds and 177211 microseconds. The time almost doubles, which proves the correctness and generality of our testbed. We also find that the ratio between the transmission time of a flow with a load 2x Mb and that with a load x Mb is decreasing slightly when x is increasing. This is because with the load increasing, the transmission time increases; the other processing times, such as queueing delay and propagation delay, are roughly constants, which has less influence on the flow's completion time.

Fig. 7(b) shows the average CCT of the three algorithms. It is obvious that our MCRS always has the least average CCT. The differences between MCRS and other algorithms are larger when the traffic loads of flows are heavier. When the loads are 1 Mb, 3 Mb and 5 Mb, the average CCT ratios between MCRS and other two algorithms are 81.84% and 76.17%. When the loads are $64Mb, 192Mb$ and 320 Mb, the average CCT ratios between MCRS and other two algorithms are 81.76% and 74.90%, both of which are smaller than the ratios with lighter traffic loads. It indicates that the

longer transmission time is, the more outstanding our MCRS is compared with the other two. Additionally, the Arbitrary Routing algorithm has a better performance than the Fair Scheduling algorithm. It demonstrates that scheduling is more important than routing to some extent when the objective is to minimize the average CCT. Obviously, the joint routing and scheduling strategy is better than either of the single influence factor algorithms since both routing and scheduling factors effect the coflow transmission time.

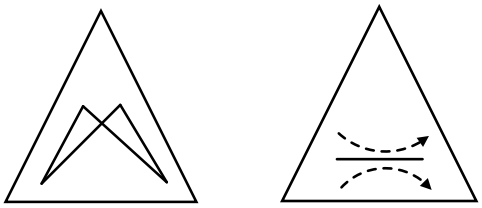
7. Performance Evaluation

In this section, we evaluate our proposed algorithm's, MCRS, performance by packet-level simulations. We use a suite of production traces in the Coflow-Benchmark as input in our simulations [38]. These traces are synthesized based on the one-hour workload collected from Facebook. We compare MCRS with the following state-of-the-art methods: 1) Scheduling-only: MCRS with only one-hop paths no matter the link congestion; 2) Routing-only (illustrated in Fig. 8(a)): all the individual flows (solid lines) are routed by ECMP [39], in which all flows' bandwidths are assigned by the max-min fairness strategy; 3) Heuristic (illustrated in Fig. 8(b)): all coflows (dashed line) equally share links (solid line) and bandwidth saving through alignment with the maximum completion time in a non-preemptive way [40]. Here, we highlight our key results:

- For both the average and maximum coflow completion time, MCRS outperforms the Routing-only, Scheduling-only, and Heuristic strategies.
- MCRS has a satisfying performance in the aspects of CCT and the maximum concurrent coflow number when applied to different traffic loads.
- MCRS consistently outperforms the baseline Scheduling-only strategy over a wide range of values for different coflow parameters, such as the number of coflows, the size, the width, and the inter-coflow arrival intervals.

7.1. Experimental Settings

Coflow Parameters: A coflow is measured by three main features: 1) width: the total number of individual flows; 2) length: the size of the largest



(a) Routing-only. (b) Heuristic.

Figure 8: Illustration of comparison algorithms.

flow it contains; and 3) size: the total amount of data in megabytes. Similar to [20; 17], we divide non-zero coflows into four categories as shown in Table 2. A coflow is W (wide) if it involves more than 50 flows and otherwise it is N (narrow); a coflow is L (long) if its length is greater than 5MB and otherwise it is S (short).

Network Topology: The network topology is shown in Figure 10, consisting of 64 leaf and 32 spine switches. Each leaf switch is connected to eight servers. Each link has a bisection bandwidth of ten Gbps. Our following experiments prove the efficiency and superiority of our model, MCRS, comprehensively. Our simulation is accomplished by using MATLAB.

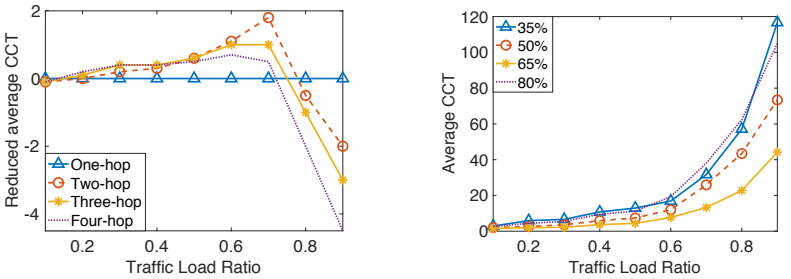
Metrics: To compare our algorithms with the other three algorithms, we measure the average CCT. The independent variable is the traffic load ratio, which is the ratio between the sum of all the allocated bandwidth and the total initial link capacity. To test the impact of coflow parameters, we measure the improvement, η, in the average CCT when comparing two schemes by changing the coflow parameters. Take Scheduling-only as the baseline. It can be calculated as

$$\eta(\%) = \frac{ave\ CCT(Baseline) - ave\ CCT(MCRS)}{ave\ CCT(Baseline)}$$

We run multiple times for each case and calculate the average as our final result.

7.2. Impact of Path Hops and Threshold

In this subsection, we test the relationship of the average CCT with the path hops (path length) and the threshold, respectively. We change the traffic load ratio from 0.1 to 0.9 with an interval of 0.1. In Fig. 9(a), each line represents the result of routing flows in no more than k-hop paths ($k =$



(a) Reduced average CCT. (b) Average CCT with different thresholds.

Figure 9: Impact of path hops and threshold.

Coflow type	1	2	3	4
Length	Short	Long	Short	Long
Width	Narrow	Narrow	Wide	Wide
Ratio of coflows	52%	16%	15%	17%
Ratio of bytes	0.01%	0.67%	0.22%	99.10%

Table 2: Coflow categories by length and width.

$\{1, 2, 3, 4\}$). We find that when the traffic load is heavy, the average CCT of multiple-hop paths increases rapidly. For example, when the ratio changes from 0.8 to 0.9, the average CCT of all four-hop paths becomes 40.2% longer. However, when the traffic load is light from 0.1 to 0.6, it is acceptable of two-hop paths since the average CCT is only 0.52 times on average larger than the one-hop paths. However, the average CCT of three-hop paths is 1.73 times larger than the one-hop paths. It indicates that applying two-hop paths does not increase the bandwidth consumption too much while three-hop and four-hop paths consume too much extra bandwidth leading to a much longer average CCT. In Fig. 9(b), the result shows that the impact of the threshold changing into two-hop paths on the average CCT. 65% has the minimum average CCT, indicating the suitability of our threshold value. 50% is also acceptable while 35% and 80% have relatively longer average CCTs. This is because either too many or too few two-hop paths are far from optimal scheduling and routing. Consequently, we allow one-hop and two-hops for routing flows and the turning point is 65% of the longest coflow completion time.

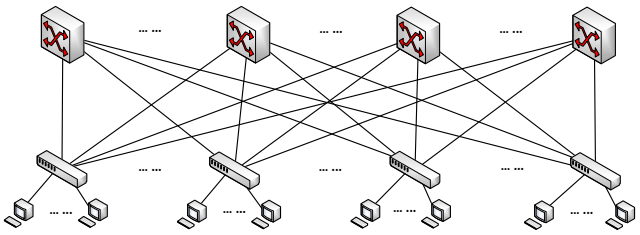
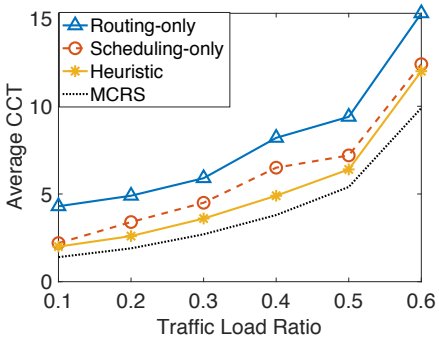
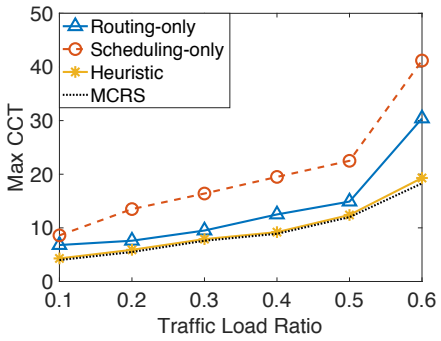


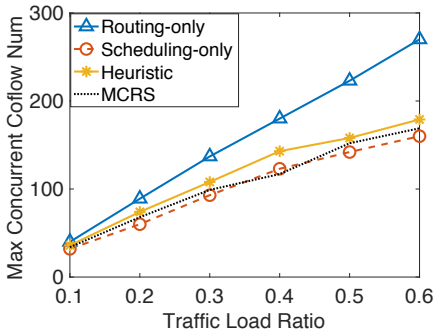
Figure 10: Network topology for simulations.



(a) Average coflow completion time.



(b) Maximum coflow completion time.



(c) Maximum concurrent coflow number.

(d) Execution time.

Figure 11: Performance comparisons between MCRS and others.

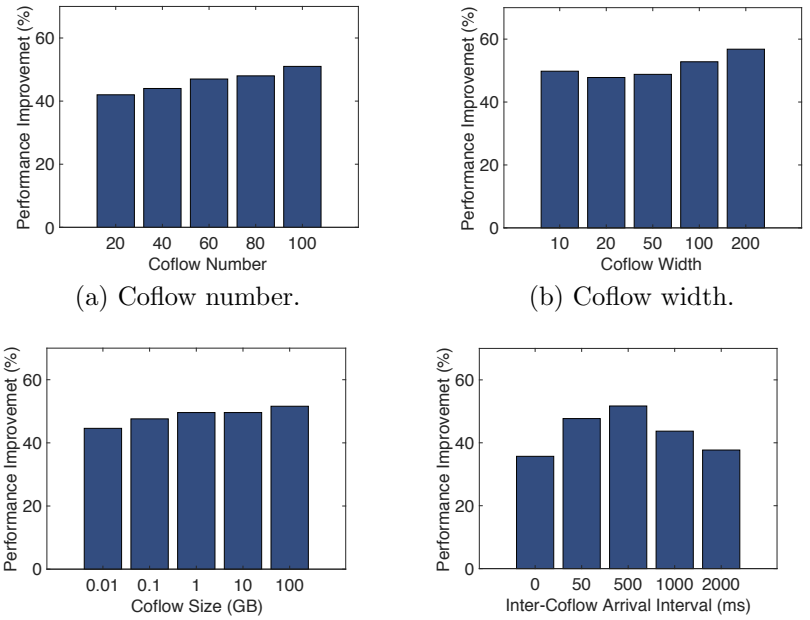
7.3. Performance Comparison

In this section, we evaluate the performance of our strategy, MCRS, against the Routing-only, Scheduling only, and Heuristic methods. Fig. 11 illustrates our results. We compare these methods in three aspects: the average CCT (Fig. 11a), the maximum CCT (Fig. 11b), the number of the concurrent coflows in the network (Fig. 11c) and the execution time of applying the algorithms (Fig. 11d).

Fig. 11a shows that all these four strategies' average CCTs increase with the growth of the traffic load. It can be explained that with more coflows, the competition for bandwidth becomes more severe and more coflows have to wait a much longer time to transmit. We can observe that MCRS always outperforms all the other methods. It can reduce the average CCT by up to 71.3% compared to the Routing-only strategy, while Scheduling-only and Heuristic can only reduce it by 50.0% and 32.2%. In this case, the coflow scheduling will contribute more than the coflow routing when minimizing the CCTs. Because Scheculing-only, Heuristic, and MCRS consider the coflow routing, they have a better performance than Routing-only. MCRS reduces by up to 31.7% compared to Scheduling-only. Moreover, we can see that MCRS performs even better than Heuristic, which illustrates the advantages of our intra and inter coflow routing and scheduling algorithms.

In the aspect of the maximum CCT among all the coflows, the tendency of each line is still increasing with larger traffic loads, shown in Fig. 11b. The reason is that with more coflows, the waiting time for coflows with a longer remaining time increases. We can see that MCRS always has the smallest CCT among all these four approaches, while Scheduling-only has the largest. Compared to the Scheduling-only strategy, MCRS reduces the maximum CCT by up to 69.2%, while Routing-only and Heuristc reduce it by 17.9% and 68.1%. Though the difference between MCRS and Heuristic is not so obvious, as in Fig. 11a, the results still show that MCRS is in an advantageous position.

Next we investigate the number of concurrent coflows while executing the schemes. Fig. 11c illustrates that both MCRS and Scheduling-only have a small number of concurrent coflows on average. Compared to the Routing-only approach, MCRS has only 50.1% of its coflows in the network when the traffic load ratio is 60%, while Scheduling-only and Heuristic have 45% and 48.3% of their coflows on average. This is because MCRS applies the shortest remaining time first algorithm, which transmits the smallest number of coflows at the same time.



(a) Coflow number. (b) Coflow width.

(c) Coflow size. (d) Inter-coflow arrival interval.

Figure 12: Performance improvement.

We further evaluate the execution times of running these four algorithms in Fig. 11d. When the traffic load ratio becomes larger, the speed of their execution times increases because more flows are included and more calculations are needed. The execution time of running our proposed algorithm, MCRS, is the smallest. This indicates the advantage of its low time complexity. Though Alg. Heuristic has the second best average CCT in Fig. 11a, its execution time is the largest and is even 43.1% larger than that of MCRS when the traffic load ratio is 0.8.

7.4. Impact of Coflow Parameters

We also study the impacts of different coflow parameters on the performance of MCRS, such as the total coflow number, the coflow width, the coflow size, and the inter-coflow arrival interval. We use the Scheduling-only algorithm as the baseline. The basic fix settings of the coflow parameters are: the coflow number is 100; the coflow width is 100; the coflow size is 500Mb; the inter-coflow arrival interval is 500ms. Generally, in Fig. 12, we can see that MCRS always outperforms the baseline under different scenarios.

Coflow Number: To evaluate the impact of the coflow number on the performance of the routing and scheduling schemes, we fix the other parameters, i.e., setting the coflow width, the coflow size, and the mean inter-coflow arrival interval as 100, 500MB, and 100ms, respectively [17]. We then inject different numbers of coflows into the network, and calculate the improvement of the average CCTs for MCRS compared with the baseline. Fig. 12a shows that the improvement in the average CCT increases with the growth of the coflow numbers. The reason is that more coflows would lead to a more severe competition for the network resources, and the efficient solutions will gain more benefits. MCRS can reduce the average CCT by up to 50.4% (see the case of 100 coflows) when compared to the baseline.

Coflow Width: Recall that the coflow width is the number of flows within it. In this experiment, we fix the coflow number, the size, and the mean inter-coflow arrival interval as 100, 500MB, and 100ms, respectively, and then we study the influence of the coflow width to the average CCTs. Fig. 12b shows that the larger coflow width, the more improvement on the average CCT is gained by each of the three schemes. The reason is still that more data communication leads to a more severe competition for the network resources. MCRS can reduce the average CCT by up to 53.8% compared to the baseline.

Coflow Size: Here, we fix the coflow number, the width, and the mean inter-coflow arrival interval to be 100, 100, and 100ms, respectively. In each experiment, we then send coflows of the same size into the network. For different coflow size, Fig. 12c shows that MCRS can reduce the average CCT by up to 48.1% when compared to the baseline. In addition, the improvement in the average CCTs basically increases with the growth of the coflow sizes. This is because under the online setting, a larger coflow size leads to more severe collisions among coflows. Compared to the baseline, MCRS can result in a much smaller number of concurrent coflows in the network. Compared with the baseline, our strategy also conducts scheduling, which contributes to the relief of the collisions. Therefore, the larger the coflow size is, the more improvement MCRS gains.

Inter-Coflow Arrival Interval: For each of the experiments in this part, the mean inter-coflow arrival interval is fixed at a value by modifying the intervals in the original setting. The other parameters are fixed as in the previous sections. We investigate the intervals starting from 0, which means that all coflows arrive at the same time (the same as the offline model). The results are shown in Fig. 12d.Moreover, we can observe that when

the interval becomes extremely large, the improvement of the schemes will decrease significantly, i.e., when the interval is 2s, little improvement can be gained. The reason is that if the interval is too large, most coflows will finish their transmissions during the interval and there will be little interaction among the coflows. In this case, the coflow routing will contribute more than the coflow scheduling in minimizing the CCTs.

Therefore, two-hop paths are suitable to be applied into the routing, while more hop paths incur too much bandwidth consumption. MCRS can transmit coflows much faster than the Routing-only, Scheduling-only, and Heuristic strategies and is more likely to be practical in real applications. For different coflow parameters, MCRS has the performance improvement of at least 41.8%. Coflow size improves performance least while coflow width improves most.

8. Conclusion

In this paper, we focus on the coflow routing and scheduling problem under the quite popular data center infrastructure: the Leaf-Spine topology. The single coflow's routing and scheduling problem has already been proven to be NP-hard, and multiple coflows surely make the problem more challenging. Our goal is to minimize the average CCT. First, we analyze the path blocking probability of this two-tier structure and propose to apply both one-hop and two-hop paths for all the flows. Then we propose two algorithms for inter-coflow and intra-coflow routing and scheduling, respectively, and theoretically prove that our strategy has a reasonably good competitive ratio. Extensive experiments show that our algorithms outperform the state-of-the-art schemes under the specific Leaf-Spine topology in different traffic scenarios.

9. Acknowledgment

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS-1651947, CNS 1564128.

10. References

References

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 2–2.

[3] "Google dataflow," www.google.com/events/io.

[4] S. Zhang, S. Zhang, X. Zhang, Z. Qian, M. Xiao, J. Wu, J. Ge, and X. Wang, "Far-sighted multi-stage aware coflow scheduling," in *GLOBECOM*, 2018.

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

[6] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *HotNets-XI*, 2012.

[7] M. Shafiee and J. Ghaderi, "Scheduling coflows in datacenter networks: Improved bound for total weighted completion time," *ACM SIGMETRICS Performance Evaluation Review*, vol. 45, no. 1, pp. 29–30, 2017.

[8] N. Kang, Z. Liu, J. Rexford, and D. Walker, "Optimizing the "one big switch" abstraction in software-defined networks," in *CoNEXT*, 2013.

[9] L. Luo, Y. Kong, M. Noormohammadpour, Z. Ye, G. Sun, H. Yu, and B. Li, "Deadline-aware fast one-to-many bulk transfers over inter-datacenter networks," *IEEE Transactions on Cloud Computing*, 2019.

[10] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *SIGMOD*, 2010.

[11] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *NSDI*, 2011.

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

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Computer Communication Revie*, vol. 38, no. 2, 2008.

[14] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao, and H. Zheng, "Packet-level telemetry in large datacenter networks," in *SIGCOMM*, 2015.

[15] T. Mizrahi and Y. Moses, "Time4: Time for sdn," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 433–446, 2016.

[16] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Cacheflow: Dependency-aware rule-caching for software-defined networks," in *SOSR*, 2016.

[17] Y. Li, S. H.-C. Jiang, H. Tan, C. Zhang, G. Chen, J. Zhou, and F. C. M. Lau, "Efficient online coflow routing and scheduling," in *MobiHoc*, 2016.

[18] Y. Chen and J. Wu, "High network utilization load balancing scheme for datacenters," in *GLOBECOM*, 2016.

[19] Y. Zhao, K. Chen, W. Bai, M. Yu, C. Tian, Y. Geng, Y. Zhang, D. Li, and S. Wang, "Rapier: Integrating routing and scheduling for coflow-aware data center networks," in *INFOCOM*, 2015.

[20] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *SIGCOMM*, 2014.

[21] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *SIGCOMM*, 2015.

[22] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, 2012.

[23] Z. Qiu, C. Stein, and Y. Zhong, "Minimizing the total weighted completion time of coflows in datacenter networks," in *SPAA*, 2015.

[24] H. Zhang, L. Chen, B. Yi, K. Chen, M. Chowdhury, and Y. Geng, "Coda: Toward automatically identifying and scheduling coflows in the dark," in *SIGCOMM*, 2016.

[25] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized task-aware scheduling for data center networks," in *SIGCOMM*, 2014.

[26] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *SIGCOMM*, 2010.

[27] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," in *SIGCOMM*, 2012.

[28] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan, "Minimizing flow completion times in data centers," in *INFOCOM*, 2013.

[29] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *SIGCOMM*, 2012.

[30] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *SIGCOMM*, 2013.

[31] S. Agarwal, S. Rajakrishnan, A. Narayan, R. Agarwal, D. Shmoys, and A. Vahdat, "Sincronia: Near-optimal network design for coflows," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 16–29.

[32] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 691–703, 1976.

[33] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, G. Varghese *et al.*, "Conga: Distributed congestion-aware load balancing for datacenters," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, 2014, pp. 503–514.

[34] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev *et al.*, "B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan," in *SIGCOMM*, 2018.

[35] L. Schrage, "Letter to the editor—a proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, no. 3, pp. 687–690, 1968.

[36] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "Conga: Distributed congestion-aware load balancing for datacenters," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 503–514, 2014.

[37] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach (6th Edition)*. Pearson, 2012.

[38] M. Chowdhury, "Coflow-benchmark," https://goo.gl/szsBQE, 2015.

[39] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," in *RFC 2992*, 2000.

[40] L. Chen, B. Li, and B. Li, "Barrier-aware max-min fair bandwidth sharing and path selection in datacenter networks," in *IC2E*, 2016.