Multi-hop Coflow Routing and Scheduling in 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. A useful abstraction, *coflow*, is proposed to express the communication requirements of prevalent data parallel paradigms. The multiple coflow routing and scheduling problem faces challenges when deriving a good theoretical performance ratio because coexisting coflows will compete for the same network resources such as link bandwidths. In this paper, we focus on the coflow problem in the most 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 a Multi-hop Coflow Routing and Scheduling strategy (MCRS) for inter-coflows and intra-coflows and prove that our method has a reasonably good competitive ratio. Extensive experiments show that MCRS outperforms other state-of-art schemes.**

***Index Terms*—Leaf-spine, coflow, routing, scheduling.**

I. Introduction

With the explosive growth of data-parallel computation frameworks such as MapReduce, Spark, Google Dataflow, etc., some applications only care about application-level information instead of an individual flows behaviour. An abstraction, *coflow*, is proposed to model such application-level information scenarios. A coflow is defined as a collection of parallel flows with a common performance goal [1]. In this paper, we focus on minimizing the average coflow completion time (CCT) in a most popular data center topology: Leaf-Spine [2]. For simplification, we assume that all flows within a coflow are generated at the same time. Every single flow can be routed towards only one path to avoid packet reorder costs.

We use a simple example in Fig. 1 to illustrate our motivation. Each link in the network has a 1 Mbps available bandwidth. At time t, two coflows: coflows a and b, are generated. 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 1 Mb; 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 3 Mb. A reasonable routing assignment is shown in Fig. 1. The flows' paths are all one-hop paths that pass through only one spine switch. The optimal scheduling plan allocates both flows of coflow a 0.25 Mbps bandwidth and both flows of coflow b 0.75 Mbps bandwidth. After 1s, a new coflow c in a dashed red line is generated with only one flow, $f_1^{(c)}$, from h_1 to h_3. The biggest available bandwidth among all one-hop paths for $f_1^{(c)}$ is 0.25 Mbps. This strategy's average CCT is 4.75s. However, we notice that there is a detour two-hop path

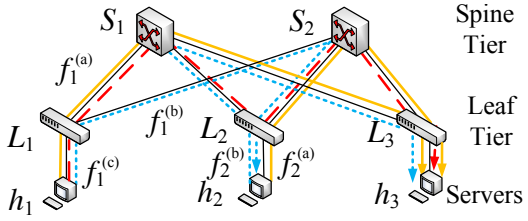


Fig. 1: Coflow example.

with 0.75 Mbps bandwidth, which passes L_1, S_1, L_2, S_2, and L_3. Its average CCT is 4.25s, which is 0.5s shorter. It inspires us to obtain a shorter average CCT by trading a longer path.

In this paper, we dive into the online coflow routing and scheduling problem with an objective of minimizing the average CCT in the Leaf-Spine topology. We start with the routing issue on the one-hop and two-hop path selection and give a blocking probability analysis of multi-hop paths. To minimize the average CCT, we present both inter-coflow and intra-coflow routing and scheduling algorithms based on some crucial observations. A sound competitive performance ratio is given with a detailed proof. We also apply work conservation to refine our algorithms. Current load balancing strategies intended to use one-hop routing paths for all flows. However, our previous work [3] showed that multi-hop paths can improve bandwidth utilization. Extensive simulations demonstrate the efficiency of our strategies against some state-of art works.

The remainder of the paper is organized as follows. Section II surveys the related work. Section III states the problem formulation and studies the routing issues. Section IV focuses on the intra- and inter-coflow routing and scheduling solutions. Section V provides a theoretical analysis of our scheme and proves it has a reasonably competitive ratio. Section VI includes the experiments. The paper concludes in Section VI.

II. Related Work

Though coflow management is a relatively novel topic, a growing body of recent work [4–8] has demonstrated that using coflows can significantly improve the communication performance of distributed data-parallel applications. The coflow abstraction was first proposed as "a networking abstraction to express the communication requirements of prevalent data parallel programming paradigms" [9], which studied the communication requirements of diverse cluster computing applications and proposed to use grouped data flows as the network model for data parallel jobs. Originally, some papers [6, 7] studied coflow scheduling only. However,

Rapier [4] proved that scheduling-only coflow strategies could not optimize the application-level performance. Rapier took both coflow routing and scheduling into consideration and advanced a heuristic solution for the general topology. Yu et al. [8] presented a rounding-based randomized approximation algorithm for single coflow routing and scheduling, which was the first online theoretical performance-guaranteed solution for multi-coflows. Nevertheless, the competitive ratio is loose because the single coflow routing and scheduling problem for minimizing the CCT is already NP-hard [10]. With multiple coflows, the inter-coflows' and intra-coflows' paths will overlap and flows will compete for the same link resources. What's more, cluster computing frameworks are dynamic in providing enough prior knowledge, which demands an online approach. Consequently, we relax the problem into a the Leaf-Spine network architecture.

III. Framework

A. Model and Formulation

A Leaf-Spine network is modelled 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 m coflows in total during the whole process. Note coflow C_i $(1 \leq i \leq m)$, which arrives at time T_i and contains w_i individual flows. A flow j $(1 \leq j \leq w_i)$ within a coflow C_i is defined by a 3-tuple $(s_j, t_j, v_j) \in C_i$, where s_j and $t_j \in V$ are the source and destination nodes, and $v_j > 0$ is the flow volume. The path and the bandwidth for flow j in coflow C_i are denoted as $p_j^{(i)}$ and $b_j^{(i)}$, respectively.

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, similar to [8]. At time $x \geq T_i$, flow $(s_j, t_j, v_j) \in C_i$ 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, which means that this flow is waiting for transmission. Because we relax our problem into the special Leaf-Spine topology, the path 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. A one-hop path passes only one spine switch during the flow's transmission and so on. Furthermore, the number of paths between any pair of nodes is bounded by $poly(n)$ because there are $|S|$ one-hop paths and $|S|^2 * (|L| - 2)$ two-hop paths and we have $O(|S| + |S|^2 * (|L| - 2)) = O(|S|^2 * (|L| - 2)) = O(n^3) = O(poly(n))$. A routing and scheduling strategy for a coflow, C_i, is defined as $S_i := \{p_j^{(i)}(x),\ b_j^{(i)}(x)\}_{j=1}^{w_i}$ A time-slotted system is considered. We then define the CCT t_i for coflow C_i to be the minimum time, such that $\sum_{x=T_i}^{T_i+t_i} b_j^{(i)}(x) \geq v_j^{(i)}, \forall j \in [1, w_i]$

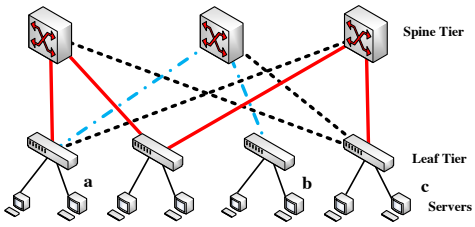


Fig. 2: Unbalanced traffic situation.

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, we assume each flow's path can only be decided once. A valid strategy is 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 $C_i := \{(s_j, t_j, v_j)\}_{j=1}^{w_i}$, 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.

B. Routing Path Block Probability Analysis

We assume that the routing path is either one-hop or two-hops long under the Leaf-Spine topology. It is inspired by a recently proposed distributed routing mechanism for the Leaf-Spine topology, called CONGA. CONGA scheduled paths based on a real-time fabric congestion situation, which was obtained by feedback from the remote switches. CONGA limited packets to a one-hop routing path. However, traffic is likely to be heavily unbalanced in data centers because of bursts in a few congested links, shown as the black dotted lines in Fig. 2. The red and blue 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 red line path [3], which detours at the second leaf switch, is able to avoid the congested links. The 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 $\rho_b = \prod_{k \in S}[1 - p(i, k) * p(k, j)]$ So, the non-blocking possibility is $\rho_n = 1 - \rho_b = 1 - \prod_{k \in S}[1 - p(i, k) * p(k, j)]$ 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 $\rho_n' = \rho_n + \sum_{k,k' \in S, m \in L'}[p(i, k) * p(k, m) * p(m, k') * p(k', j)]$. So,

the difference between ρ'_n and ρ_n is calculated as $\rho'_n - \rho_n = \sum_{k,k' \in S, m \in L'} [p(i,k) * p(k,m) * p(m,k') * p(k',j)]$. 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. The traffic spike and an unbalanced bursty traffic condition can make our routing strategy superior [3].

IV. Coflow Routing and Scheduling Strategy

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). In order to minimize the total completion time, SRTF selects the process with the smallest amount of time remaining until completion to execute. It's a preemptive method. In this case, even if a coflow is occupying the bandwidth in the network, it will be preempted by the new 'smaller' coflows.

***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 CCT, if there is leisure bandwidth available for multiple flows, we should select the flows with a larger workload to execute first.

MCRS optimized the average CCT in the data-intensive Leaf-Spine topology DCNs by considering both the routing and scheduling of the coflows. MCRS mainly focuses on large coflows in the DCNs, while deadline-driven individual flows or small coflows are routed directly as background traffic. 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. We provide two algorithms for inter-coflows and intra-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, 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. We need to schedule unfinished coflows to utilize the bandwidths.

Alg. 1 solves the single coflow routing and scheduling problem with a minimum CCT. It first solves a linear programming problem. Then, it schedules each flow to its one-hop path with the maximum bandwidth and calculates 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 arranged flows to ensure that they have the same completion time and update the residual bandwidth. Alg. 2 describes the inter-coflow arrangement method of MCRS. We sort the coflows in the order of their remaining completion times and execute them one-by-one.

Algorithm 1 Smallest Remaining Coflow First Inter-coflow Algorithm

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

1: Use the Intra-coflow Algorithm to calculate each remaining coflows' completion time t_i;
2: Sort these coflows' completion time t_i non-increasingly according to their transmission time;
3: Apply the allocation to the coflow with the smallest t_i.

V. Theoretical performance analysis

A. Single Coflow Routing and Scheduling

We first consider the single coflow case [4]: $C_i := \{(s_j^i, t_j^i, v_j^i)\}_{j=1}^{w_i}$. According to the above setting, for each flow $f_j \in C_i$, the source-destination pair and the volume are given. The path set, P_j^i, for each flow $f_j \in C_i$ 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.

$$\min \ t_i \tag{1}$$

subject to

$$v_j^i = b_j^i * t_i, \qquad 1 \le j \le w_i \tag{2}$$

$$\sum_{j=1}^{w_i} \sum_{e \in p} b_j^i x_{j,p}^i \le R_e, \qquad e \in E \tag{3}$$

$$\sum_{p \in P_j^i} x_{j,p}^i = 1, \qquad 1 \le j \le w_i \tag{4}$$

$$x_{j,p}^i = \{0,1\}, \qquad 1 \le j \le w_i \tag{5}$$

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 $x_{j,p}^i$ 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 [10]. 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., $b_j^i = \alpha_i * v_j^i$. As for the binary variable, we relax the binary constraint into a continuous one. Thus, we have $\alpha_i = \frac{1}{t_i}$ and Program A can be modified as:

$$\max \ \alpha_i \tag{6}$$

subject to

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

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

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

Algorithm 2 The Intra-coflow Algorithm

In: Coflow $C_i = \{(s_j, t_j, v_j)_{j=1}^{w^i}\}$;
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 $\{y_{j,p}^i,\ p \in P_j^i\}_{j=1}^{w_i}$ for the Linear Program B.
2: **for** each flow f_j in C' **do**
3: Choose $p \in P_j^i$ with $\max\{y_{j,p}^i\}$ 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 $x_{j,p}^i$ in its above first constraint makes the problem a concave problem that is hard to handle. So, we bring in new variables $y_{j,p}^i$ to replace the product, i.e., $y_{j,p}^i = \alpha_i x_{j,p}^i$. According to Eq. 14, we get $\alpha_i = \sum_{p \in P_j^i} y_{j,p}^i = \frac{1}{w_i} \sum_{j=1}^{w_i} \sum_{p \in P_j^i} y_{j,p}^i$. Consequently, our problem can be transformed into the linear Program B as follows:

$$\max \ \frac{1}{w_i} \sum_{j=1}^{w_i} \sum_{p \in P_j^i} y_{j,p}^i \tag{10}$$

subject to

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

$$y_{j,p}^i \ge 0, \qquad 1 \le j \le w_i \tag{12}$$

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 of the relaxed LP can be obtained in polynomial time using standard solvers.

***Theorem** 1:* 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 $\{y'^i_{j,p}\}$. From Alg. 1, we route each flow to the path with the maximum $\{y_{j,p}^i\}$ and hence, we have $y_{j,p}^i \ge y'^i_{j,p}/|S|^2(|L|-2)$, 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.

B. Multiple Coflow Routing and Scheduling

From above, we know that the competitive ratio ρ in 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) strategy.

***Theorem** 2:* Suppose the competitive ratio of Alg. 1 is ρ. Then, Alg. 2 is $\frac{(m+1)}{2} * \rho$-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 times have the relationship of $\text{OPT}_1 \le \text{OPT}_2 \le ... \ \text{OPT}_m$. 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

$$t' = t_1 + t_2 + ... + t_m = (OPT_1) + (OPT_1 + OPT_2) + ... + (OPT_1 + OPT_2 + ... + OPT_i) = \sum_{i=1}^{m} (m - i + 1) * OPT_i$$

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

$$\sum_{i=1}^{m} (m - i + 1) * OPT_i \le \frac{1}{m} (\sum_{i=1}^{m} OPT_i)(\sum_{i=1}^{m} i)$$
$$= \frac{1}{m} (\sum_{i=1}^{m} OPT_i) \frac{m(m+1)}{2} = \frac{(m+1)}{2} (\sum_{i=1}^{m} OPT_i)$$

Thus, we have $t \le t' \le \frac{(m+1)}{2} (\sum_{i=1}^{m} OPT_i)$. Therefore, our theorem is proven. □

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.

Compared to the performance bound in [8], our algorithm, MCRS, has a much tighter competitive ratio.

C. 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. 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 coflows, the coflows 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 transmitting 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.

VI. Performance Evaluation

We evaluate our MCRS's performance by packet-level simulations. The input is a suite of production traces in the Coflow-

TABLE I: Coflow categories by length and width

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%

Benchmark [11]. These traces are synthesized from the one-hour workload collected from Facebook. We compare MCRS with the following state-of-the-art methods: 1) Routing-only: all the individual flows are routed by ECMP [12], in which all flows' bandwidths are assigned by the max-min fairness strategy; 2) Scheduling-only (baseline): MCRS with only one-hop path; 3) Heuristic: the barrier-aware strategy (BAS) [13] routes and schedules inter-coflows fairly and intra-coflows to finish at the same time.

A. 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 flow it contains; and 3) size: the total amount of data in megabytes. Similar to [6, 8], we divide non-zero coflows into 4 categories as shown in Table I. 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). Our leaf-spine topology includes 4 leaf and 4 spine switches. Each leaf switch is connected to 8 servers. Each link has a bisection bandwidth of 10 Gbps. We measure the improvement, η, in the average CCT when comparing two schemes. Take MCRS with the baseline Scheduling-only as an example. It can be calculated as $\eta(\%) = \frac{ave\ CCT(Baseline) - ave\ CCT(MCRS)}{ave\ CCT(Baseline)}$.

B. Performance Comparison

Fig. 3 illustrates our results in three aspects: the average CCT, the maximum CCT, and the number of the concurrent coflows. Fig. 3a shows that all four strategies' average CCTs increase with the growth of the traffic load. With more coflows, the competition for bandwidth becomes more severe and more coflows have to wait a much longer time to transmit. We observe that MCRS always outperforms all the other methods. It can reduce the average CCT by up to 74.3% compared to the Routing-only strategy, while Scheduling-only and Heuristic can only reduce it by 51.4% and 35.8%. In this case, the coflow scheduling will contribute more than the coflow routing when minimizing the CCTs. Since Scheduling-only, Heuristic, and MCRS consider coflow routing, they have a better perfomance than Routing-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. 3b. 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 these four approaches. 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. 3a, the results still show that MCRS is in an advantageous position.

We further investigate the number of coflows. Fig. 3c illustrates that MCRS has less concurrent coflows on average. Compared to the Routing-only approach, MCRS has only 50.1% of its coflows 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 to transmit the smallest number of coflows at the same time.

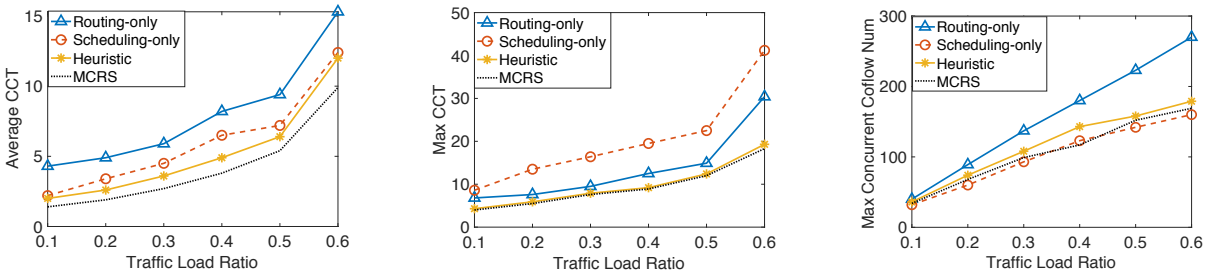
C. Impact of Coflow Parameters

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 strategy, as the baseline and traffic load ratio is about 20%-45%. Generally, in Fig. 4, MCRS has the performance improvement at least 41.8% for low traffic loads.

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 to 100, 0.5GB, and 100ms, respectively [8]. 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. 4a 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 resources, and efficient solutions will gain more benefits. MCRS can reduce the average CCT by up to 44.1% (see the case of 100 coflows) against the baseline.

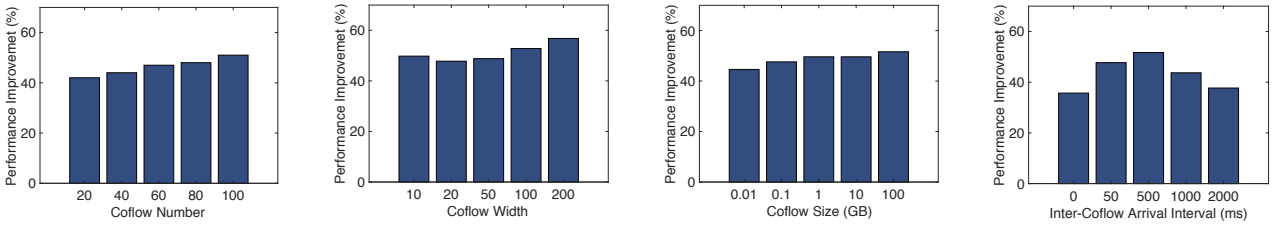
Coflow Width: Recall that the coflow width is the number of flows within it. We fix the coflow number, the size, and the mean inter-coflow arrival interval to 100, 500MB, and 100ms, respectively, and then, study the influence of the coflow width on the average CCTs. Fig. 4b shows that the larger coflow width, the more improvement on the average CCT is gained by each of the three schemes. The reason is still that more data communications lead to a more severe competition for the network resources. MCRS can reduce the average CCT by up to 55.7% compared to the baseline.

Coflow Size: We fix the coflow number, the width, and the mean inter-coflow arrival interval to be 100, 100, and 100ms, respectively. We send coflows of the same size into the network. For different coflow sizes, Fig. 4c shows that MCRS can reduce the average CCT by up to 50.2% compared to the baseline. 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 with baseline, MCRS can result in a much smaller number of concurrent



(a) Average coflow completion time. (b) Maximum coflow completion time. (c) Maximum concurrent coflow number.

Fig. 3: Performance comparisons between MCRS and others.



(a) Coflow number. (b) Coflow width. (c) Coflow size. (d) Inter-coflow arrival interval.

Fig. 4: Performance improvement.

coflows. Compared with baseline, our strategy also conducts scheduling, which contributes to the relief of the collisions.

Inter-Coflow Arrival Interval: The mean inter-coflow arrival interval is fixed to 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. 4d. We can see that MCRS reduces the average CCT by up to 51.0% when compared with the baseline. Moreover, we can observe that when the interval becomes extremely large, the improvements 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. The coflow routing will contribute more than the scheduling in minimizing the CCTs.

VII. Conclusion

We focus on the coflow routing and scheduling problem under 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 of multiple coflows. 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 prove that our strategy has a reasonably good competitive ratio. Extensive experiments show that our algorithms outperform comparison schemes.

VIII. Acknowledgment

This research was supported in part by NSF grants CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, and ECCS 1231461.

References

[1] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *SIGCOMM '11*.

[2] 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," in *SIGCOMM '14*.

[3] Y. Chen and J. Wu, "High network utilization load balancing scheme for datacenters," in *GLOBECOM '16*.

[4] 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 '15*.

[5] N. Wang and J. Wu, "Minimizing the subscription aggregation cost in the content-based pub/sub system," in *ICCCN '16*.

[6] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *SIGCOMM '14*.

[7] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *SIGCOMM '15*.

[8] 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 '16*.

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

[10] 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.

[11] M. Chowdhury, "Coflow-benchmark." [Online]. Available: https://goo.gl/szsBQE

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

[13] L. Chen, B. Li, and B. Li, "Barrier-aware max-min fair bandwidth sharing and path selection in datacenter networks," in *IC2E '16*.