OMCO: Online Multiple Coflow Scheduling in Optical Circuit Switch

Chao Xu*, Haisheng Tan*, Jiahui Hou†, Chi Zhang*, and Xiang-Yang Li*
*School of Computer Science and Technology, University of Science and Technology of China, China
†Department of Computer Science, Illinois Institute of Technology, USA

***Abstract*—*Coflow* is gradually prevalent as a new traffic structure in data centers, which allows applications to convey their application-level semantics into the network. Meanwhile, optical circuit switches (OCS) are increasingly deployed in data centers due to the superiority in hardware, such as high bandwidth and low energy consumption. However, few works in the literature considered both coflow scheduling with OCS. In this paper, we study the coflow scheduling problem in an OCS-based data center network, with an aim to minimize the coflow completion time (CCT). We derive an online algorithm called `OMCO` to solve this problem. `OMCO` can not only optimize the *circuit utilization* but also minimize the number of circuit reconfigurations. Extensive simulations with real-world data traces show that `OMCO` outperforms other existing solutions dramatically. Compared with a FIFO-based scheme and the shortest-coflow-first heuristic, `OMCO` reduces the average coflow completion time by up to 61% and 62%, respectively.**

I. Introduction

Recently, a new abstraction of structured data flows, which is called *Coflow* [1], is proposed to boost flow transmission in data center networks. In brief, a coflow is a collection of related parallel flows that occur between two stages of a multi-stage computing job, such as *shuffle* flows in MapReduce.

Based on the implementation of coflow, a range of works have been proposed to improve the efficiency of traffic transmission with the support of electronic packet switches in DCNs. For instance, Varys [2] proposed a heuristic algorithm to minimize average CCTs for multiple coflow scheduling. Then, Aalo [3] further studied this problem without the prior knowledge of coflows by the Discretized Coflow-Aware Least-Attained Service method. Upon that, CODA [4] observed the constraint of applicability in Varys and Aalo, and addressed existing systems need to recognize and generate coflow manually. Consequently, CODA [4] rendered an efficient and fault-tolerant scheme to enable systems to identify and generate coflow automatically, which is also the first work to combine machine learning with coflow. In addition to considering the practicality of coflow in systems, a number of works studied the theory of coflow. Qiu *et al*. [5] provided a theoretical guarantee for the multiple coflow scheduling problems, and presented a deterministic algorithm with a constant approximation ratio.

All the above-mentioned works built their network model on traditional electronic packet switches. Packet switches have its advantages on flow transmission, such as making forwarding decisions at the granularity of individual packets. However, the slow growth rate of bandwidth in packet switches is unable to meet the need of DCNs. Therefore, as the future data-rate proof, optical circuit switches (OCS) are increasingly deployed in the modern data center. Compared with traditional electrical packet switches, OCS have many advantages such as significantly higher data transfer rate and lower power consumption. Correspondingly, OCS have a limitation on transfer mode that each ingress or egress port can establish at most one circuit for data transmission at a time, *i.e., the port constraint*. Another non-negligible feature of OCS is that any reconfiguration of the circuit needs to suffer a fixed period of time δ (*i.e.,* the reconfiguration delay), which depends on the hardware techniques in OCS and ranges from hundreds of microseconds to dozens of milliseconds in general. Researchers have seized upon these features to propose hybrid circuit/packet data center architectures, such as Helios [6] and c-through [7]. Porter *et al.* [8] adopted the idea of improving the utilization of short-lived circuits and demonstrated a traffic matrix scheduling (TMS) to schedule circuits. Based on network models in [8], Liu *et al.* [9] also provided a remarkable algorithm (*i.e.,* Solstice) to deal with the flow scheduling in hybrid networks. Solstice improved the circuit utilization significantly and reduced the burdens caused by reconfiguration delay.

Above OCS-enabled works are all coflow-agnostic, which means that they could not capture the semantics of communication in networks. To our best knowledge, Sunflow [10] is the first work to take into account the features of both OCS and coflow. Sunflow rendered an excellent approximation algorithm for a single coflow scheduling and a heuristic method to handle multiple coflow scheduling in OCS. In Sunflow, they considered coflow scheduling problem in off-line circumstance, which means all coflows arrive in the network at the initial time. However, generally coflows are generating in the DCNs from time to time, which is more practical to consider the coflow scheduling in online circumstance. As a summary, the previous works mentioned above are compared in Table I.

In this paper, we study the problem of online multiple

Haisheng Tan is the Corresponding Author (email: hstan@ustc.edu.cn).

This work is supported partly by China National Funds for Distinguished Young Scientists No. 61625205, NSFC Grants 61772489, 61502201, 61520106007, Key Research Program of Frontier Sciences (CAS) No. QYZDY-SSW-JSC002, NSF ECCS-1247944, NSF CNS 1526638, and the Fundamental Research Funds for the Central Universities at China.

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

TABLE I: Comparison of Research Outputs Related to Coflow and OCS

Solutions	Coflow-aware	OCS-enable	Online
Varys [2], Aalo [3], CODA [11], RAIPER [12], Qiu et al. [13], Chen et al. [14], Shafiee et al. [15], and etc.	✓	✗	✓
Heilos [6], c-through [7], Porter et al. [8], Liu et al. [9], and etc.	✗	✓	✓
Sunflow [10]	✓	✓	✗
OMCO [this work]	✓	✓	✓

coflow scheduling in OCS-based DCN. Essentially, the major challenges of this problem are as follows:

- Single coflow scheduling in OCS has been proved to be NP-hard [16]. In our model, we also need to consider the competition among multiple coflows, which increases the problem difficulty significantly.
- The information of upcoming coflow is unpredictable, which means we could not predict the influence of present decisions on subsequent transmission. Therefore, how to narrow the gap between our local operation and the global optimum is a hard nut to crack.

We propose a heuristic algorithm OMCO to handle this problem. Specifically, we have several contributions as follows:

- OMCO is based on a classical matrix decomposition method called BvN, and combines with our effective advancement, which makes it more effective and easier for deployment in real systems. To the best of our knowledge, OMCO is the first online algorithm for multiple coflow scheduling in OCS.
- Extensive simulations based on realistic workloads are conducted to compare OMCO with other existing algorithms. The results demonstrate that OMCO can handle multiple coflow scheduling efficiently in different scenarios, and improve the performance by up to 61% in reducing the coflow completion time.

The rest of paper is organized as follows. We define the switch model and formulate the problem in Sec. II. In Sec. III, we introduce the methodology in this paper and derive our online algorithm OMCO to handle coflow scheduling in OCS. The performance evaluation is presented in Sec. IV, and the whole work is concluded in Sec. V.

II. Model and Problem Definitions

A. Switch model

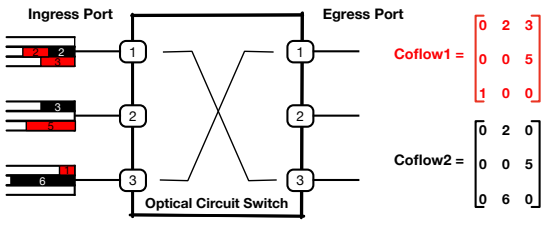


Fig. 1: Switch model illustration.

Optical Circuit Switch: Similar to [5], [8], [10], the fabric of data center network is modeled as a non-blocking optical circuit switch with N-port. Any pair of ports can establish a circuit to do traffic transmission. In the context of data centers, each switch port is connected to individual servers or ToR (Top of Rack) switches, and each ToR is connected to a group of end hosts. The data flows are buffered at sender machines and wait to be transferred by switches. The primary feature of OCS is its inherent limitation on ports, which means no input port is allowed to connect to multiple output ports in optical circuit switch, and no output port is allowed to connect to multiple input ports in a single circuit configuration. The alteration of circuit configuration incurs a fixed time delay δ (*i.e.,* reconfiguration delay), and the ports involved in the altered circuit are disabled during the period of δ. However, there exist two contrary opinions about the availability of the unaltered circuits during a circuit reconfiguration. Most previous works hold a pessimistic view that all of the circuits are torn down during the reconfiguration (*i.e., all-stop* model) [6], [7], [9], [17]. In contrast, the Sunflow [10] proposed *not-all-top* model, and considered the unchanged circuits, which are not affected and may keep available during the reconfiguration. In this paper, we are in line with the former opinion. In fact, according to the experiences from industry community, the commodity optical circuit switch could not achieve *not-all-stop* model precisely due to the limitation of hardware technology. Therefore, it is reasonable to consider the *all-stop* model as a worst case of *not-all-stop*.

Colfow: A coflow is defined as a collection of related parallel flows who transfer between two stages of a job and share a common performance goal. We denote $C_k (1 \leq k \leq m)$ as the coflow arrived at time T_k, which contains ω_k individual flows. The traffic demand of C_k is denoted as a $N \times N$ matrix D_k, where each element $d_{i,j}^k$ represents its flow demand between input port i and output port j, and the number of non-zero element is equal to ω_k. We set $f_{i,j}^k$ as the time when traffic demand between port i and j is finished. The time when the whole coflow is completed is defined as the maximum of $f_{i,j}^k$, denoted as $MAX(f_{i,j}^k)$. The *coflow completion time (CCT)* of C_k is the duration between its arrival time and completion time, denoted as CT_k, which is equal to $MAX(f_{i,j}^k) - T_k$. Without loss of generality, we assume that a coflow C_k embraces all the information about its flows and could start transmission when it arrives at the network at time T_k.

Scheduling: In OCS, a coflow scheduling is defined as a set of circuit configurations $P_1, P_2, ..., P_l$ while each reconfiguration is accompanied with a duration. Each configuration P_k is a

$N \times N$ binary matrix and also a permutation matrix.[1] The element $P_k(i,j) = 1$ if and only if the circuit between input port i and output port j is established during this configuration.

To elaborate problem background, we present Fig.1 as an illustration of our switch model.

B. Formalizing the problem

With the above settings, we define the *Online Multiple Coflow Scheduling in Optical Circuit Switch* as follows.

`Problem 1.` *In a network modeled as an $N \times N$ non-blocking OCS, m Coflows $C_1, C_2, ..., C_m$ arrives at time $T_1, T_2, ..., T_m$. The information of every coflow $C_k := \{d_{i,j}^k\}_{k=1}^{\omega_k}$ is given at its arrival. The problem is to find a feasible coflow scheduling such that the total coflow completion time, $\sum_{k=1}^{m} CT_k$, is minimized.*

III. Algorithm Design

A. Birkoff-von Neumann decomposition(scheduling)

The core method we apply to finish the traffic demand of coflow is the Birkoff-von Neumann (BvN) [18] decomposition. BvN decomposition is the extension of BvN theorem.

Given a non-negative square matrix $A = [a_{ij}] \in R^{n \times n}$, if $\forall i, j \in 1, 2, ..., n, a_{ij} \geq 0$, and $Ae = A^T e = \gamma e$ (e is a vector that each element equals to 1 and γ is a constant), then A is a γ-stochastic matrix. If the matrix A is γ-stochastic, based on the BvN theorem, there must exist $\alpha_1, \alpha_2, ..., \alpha_k \in (0,1)$ ($\sum_{i=1}^{k} \alpha_i = \gamma$) and a sequence of permutation matrix $P_1, P_2, ..., P_k$ could construct such equation: $A = \alpha_1 Q_1 + \alpha_2 P_2 + ... + \alpha_k P_k$. Besides, the equation is called the BvN decomposition of the matrix A. In the meantime, we can map BvN decomposition of a coflow demand matrix into OCS, where each permutation matrix is the OCS circuit reconfiguration status and coefficient of permutation matrix is the circuit duration time.

B. Tricks

***Padding*:** In general, the realistic demand matrices are not typically α-bistochastic. To implement BvN decomposition, we adopt a technology called *padding* to transform any demand matrix to be γ-bistochastic. Given a demand matrix D, *padding* approach is to traverse each $d_{i,j} \in D$ in order and increase its value until either the sum of row i or the sum of column j reaches γ, similar to [9]. After padding, we can acquire a feasible coflow scheduling by BvN decomposition.

***Aligning*:** It seems that BvN decomposition is the most effective to finish the demand matrix, in fact, it is indeed perfect if the reconfiguration delay is zero (*i.e.,* δ=0). Nevertheless, with non-zero reconfiguration delay, the naive BvN decomposition might have a terrible performance on coflow scheduling [9]. The reason is that when confronted with some trivial residual demands, primitive BvN decomposition always generate scheduling with plenty of configurations, and each configuration is in company with a tiny duration (generally on the same order of reconfiguration delay, δ). Hence, the frequent and noneffective reconfigurations greatly lower the efficiency and prolong the CCT seriously. To overcome this hurdle, we propose a technology called *aligning*. Aligning means that given a demand matrix D, we traverse each entry $d_{i,j} \in D$ and align it to $\lceil \frac{d_{i,j}}{\delta} \rceil \cdot \delta$. On the other hand, aligning could greatly reduce the reconfiguration frequency while suffering a fraction of the cost of configuration duration time. We will present that the cost is negligible in CCT through simulations in section IV.

[1] a permutation matrix has exactly one 1 in each row/column.

C. `OMCO` algorithm

The applicability of BvN decomposition provides some inspirations to our algorithm design, and we remedy low efficiency of BvN through two tricks: *padding* and *aligning*. Based on the improved BvN decomposition, we now present the `OMCO` algorithm. `OMCO` is designed to cope with multiple coflow scheduling, so it is composed of two steps: select coflow to do scheduling from arrived coflows according to a specific selecting strategy and finish the traffic demands of selected coflow regarding our scheduling strategy.

Selecting Strategy: Our selecting strategy is motivated by Shortest Coflow First (SCF). As we all know, SCF always allocates the highest priority to the coflow whose size is minimum (*i.e.,* the sum of traffic demands), while we allocate top priority to coflow who has minimal *traffic threshold*. Given a demand matrix D, and calculating the sum of each row/column, then traffic threshold is defined as the maximum within these values. In fact, the value of variable γ we used in *padding* procedure is equal to traffic threshold. Coflow's traffic threshold represents its minimum traffic that needs to be transmitted, which means that traffic threshold is an excellent characteristic to reflect the size of coflows.

Scheduling Strategy: Due to the incapability about future coflows, we attach importance to utilize existing resources as much as possible (*i.e.,* take full advantage of established circuits). Given the traffic demand matrix D_k of selected coflow C_k, we get a new demand matrix D_k'' after the implementation of aligning and padding procedure. We can observe that actually there exists many artificial demands in matrix D_k'', whereas the circuit configurations will consider the artificial demands. Furthermore, sometimes the artificial demands will cause some circuits become idle and decrease the utilization of circuits. Therefore, our goal is to reduce the circuit idle time while satisfying the traffic demand of coflow.

The detail of `OMCO` is elaborated in Algorithm III-C.

At first, we maintain a variable T_g to indicate the time for the completion of the last operation, and we initialize it to the arrival time of the first coflow (Line 3). In the meantime, `OMCO` algorithm would not stop as long as there is an uncompleted coflow (Line 4). We denote CA as the collection of coflows, which are waiting for scheduling in the network, and update their traffic threshold. We select the eligible coflow C_s based on our selecting strategy and call the function Single-Coflow decomposition (Line 5 to Line 8).

Initially, the selected coflow will undergo two preprocessing steps: aligning and padding (Line 3 to Line 4). We

Algorithm 1: OMCO

1 **Input** $\{(C_k, T_k)\}_{k=1}^{m}$, reconfiguration delay δ;
2 **Output** Valid scheduling sequence for$\{(C_k, T_k)\}_{k=1}^{m}$;
3 $T_g \leftarrow$ the arrival time of the first coflow;
4 $CA \leftarrow$ uncompleted coflows;
5 **while** *there exists coflows being completed* **do**
6 **for** *coflows in* CA **do**
7 update coflow's traffic threshold;
8 $C_s \leftarrow$Select coflow according to selecting strategy;
9 Call **Single-Coflow Decomposition** (C_s, T_g);

Function 1: Single-Coflow Decomposition

1 **Input** C_s, T_g, reconfiguration delay δ;
2 **Output** Valid scheduling sequence of Coflow C_s;
3 $C'_s \leftarrow$ Aligning Procedure (C_s);
4 $C''_s \leftarrow$ Padding Procedure (C'_s);
5 $ID \leftarrow C''_s - C_s$;
6 **while** *Coflow* C''_s *is not completed* **do**
7 $e, PM \leftarrow$ Max-Min Macthing;
8 $C''_s \leftarrow C''_s - PM$;
9 $T_g \leftarrow T_g + e$;
10 **for** *each circuit* $i \rightarrow j$ *established in PM* **do**
11 $\lambda \leftarrow ID_{i,j}$;
12 $\tau \leftarrow T_g - ID_{i,j}$;
13 **for** *each coflow* C_x *in* S **do**
14 **if** $\lambda > 0$ **then**
15 $\mu \leftarrow T_g - T_s$;
16 $\rho \leftarrow C_x(i, j)$;
17 $\omega \leftarrow minimum(\mu, \rho, \lambda, \tau)$;
18 $\lambda \leftarrow \lambda - \omega$;
19 $\tau \leftarrow \tau + \omega$;

denote ID (Inserted Demands) to record how many artificial demands has been inserted into original traffic demands. Undoubtedly ID is also a $N \times N$ matrix (Line 5). After preprocessing, we enter second stages, which is conceptually similar to the main loop of BvN: to find the circuit configuration and its corresponding duration. In each loop, we adopt a more efficient approach call Max-Min Matching (MMM) to find configurations and avoid frequent reconfigurations simultaneously. For instance, given a demand matrix D, we define the element $D_{i,j}$ is *schedulable* when there exists enough elements larger than $D_{i,j}$ in D and the positions of these elements can construct a perfect matching (including position of $D_{i,j}$). The MMM scheme iteratively searches the largest *schedulable* element e and updates the demand matrix according to its corresponding perfect matching PM (Line 7 to Line 8). Sometimes a schedulable element might have several perfect matchings, whereas we arbitrarily choose one would not influence the final result. Obviously, the PM is the required circuit configuration, and e is its duration.

As we mentioned before, the circuits established by PM might exist idle time because of the inserted artificial demands. Therefore, we detect each circuit in PM and denote λ as its remaining idle time that can be utilized by other coflows. Given the circuit $i \rightarrow j$ in PM, we initialize λ to the value of $ID_{i,j}$. Then, we suppose S is the collection of coflows, which are uncompleted and their arrival time is earlier than T_g. We check each coflow in S in chronological order and investigate that whether they can reutilize the circuit idle time. This procedure is called ***Plugging***. Assume coflow C_x plans to utilize the idle space of circuit $i \rightarrow j$, we maintain several variables to record the changes in circuit idle time (Line 10 to Line 16): τ represents the terminal time after last coflow finish plugging procedure, μ represents the theoretic longest time C_x can spend on this circuit, ρ points C_x's real traffic demand on circuit $i \rightarrow j$, and ω is the minimum of λ, τ, ρ, μ, which indicates the real space that C_x occupies. When all the plugging procedures are finished, OMCO recall the MMM again to decompose C''_s.

Example 1 (Illustration of OMCO)**.** *Fig.2 describes the way to handle multiple coflow scheduling in OMCO. In this network, $\delta = 2$ and one unit traffic is transmitted per unit time. Coflow C_1 arrives at time $t = 0$ and coflow C_2 arrives at time $t = 5$, and their corresponding traffic demand matrix is D_1 and D_2. When C_1 arrives at network first, D_1 transforms to demand matrix D''_1 after aligning and padding procedure, and we could get the ID (Inserted Demand) matrix by using D''_1 subtract D_1. The bold italic numbers in ID represent the inserted artificial demand on each position. We adopt BvN decomposition to abstract the first configuration $conf_1$, and the circuit $1 \rightarrow 2$ and $2 \rightarrow 1$ will be established with 100 duration. The remaining demand matrix constructs the rem_1. However, according to the ID matrix, we know that the circuit $1 \rightarrow 2$ only transmits traffics from time 0 to time 35 and becomes idle in the remaining time. In the meantime, coflow C_2 has already arrived in the network at time 5. Hence, through plugging procedure, we measure how much idle time on circuit $1 \rightarrow 2$ could be utilized by C_2, and we*

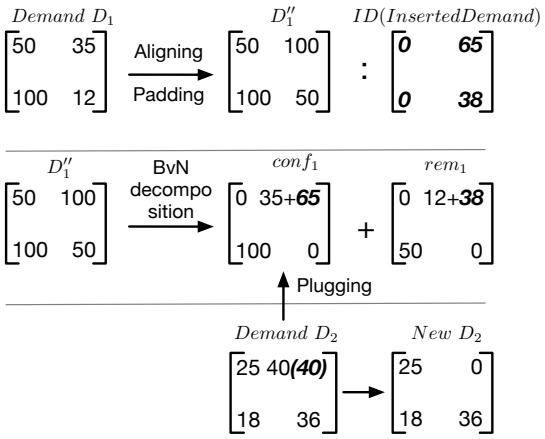


Fig. 2: An example for OMCO with 2 coflows in a 2×2 OCS.

use a bold italic number to record it in Fig. 2. After the plugging procedure, we recalculate the demand matrix D_2. When the first configuration is over, the plugging procedure will be executed before the next reconfiguration until C_1 is completed.

IV. Performance Evaluation

A. Simulation Settings

Workloads: Similar to [3], [11], [19], our workloads are based on a Hive/MapReduce [20] trace which was collected from Facebook production environment [21]. The trace consists of 526 coflow entries, and each contains several attributes about the coflow, such as the arrival time of each coflow, the number and the specific positions of mappers and reducers, and the traffic size of shuffled data.

Coflow: We measure a coflow by following parameters: 1) *volume:* the sum of all flow traffic demands in a coflow; 2) *density:* the proportion of non-zero elements in the traffic demand matrix, (*e.g.,* a 2×2 demand matrix has 2 non-zero elements, so its density is $2/4 = 0.5$.)

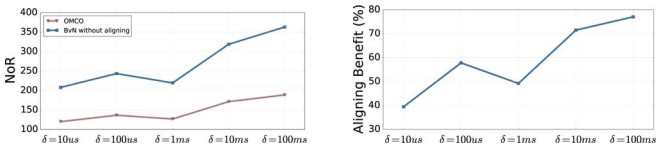
Baselines: Due to the lack of online coflow scheduling in optical circuit switch, we compare OMCO with the following classical schemes:

- *E-FIFO (Enhanced-First In First Out):* At first, E-FIFO schedules coflows in a FIFO way. Meanwhile, the scheme detects the port utilization in each circuit configuration. If there exist idle ports that could be utilized by some other coflows that have been arrived at the network, E-FIFO would establish the corresponding circuits on such idle port and schedule their corresponding flows.
- *E-SCF (Enhanced-Shortest Coflow First):* E-SCF is a similar scheme to Varys [2]. E-SCF allocate the priority according to the traffic threshold of coflow, which means the coflow with minimal traffic threshold will be allocated highest priority. Meanwhile, we also adopt the same method we used in E-FIFO to improve the port utilization in E-SCF.

Unless otherwise stated, the default full link bandwidth B is 10Gbps and the default reconfiguration delay δ is 1ms.

B. Impact of aligning

In section III, we propose a technology *aligning* to reduce the reconfiguration frequency of BvN while suffering a fraction of the cost of configuration duration. Therefore, we carry out some simulations to investigate the influence of aligning procedure to coflow completion time.



(a) Comparation of the number of reconfigurations

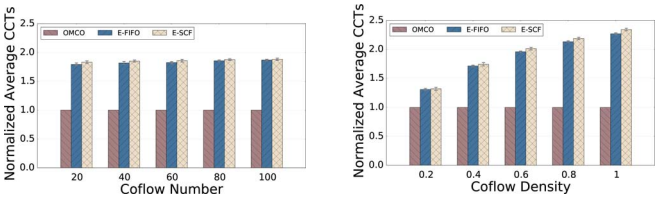
(b) Benefit of aligning

Fig. 3: Impact of aligning.

In simulations, we adjust δ from 100ms to 10us. In Fig. 3(a), the y-axis NoR represents the number of reconfigurations. We can observe that OMCO always requires fewer reconfiguration times than BvN without aligning in different δ. Meanwhile, in Fig. 3(b) we record increment of transmission time and decrement of CCT caused by aligning procedure, then we calculate the difference between two data as the aligning benefit from aligning. Obviously, the benefit provided by aligning is always larger than 40%, and up to 80%.

C. Performance of OMCO

In this section, we evaluate the performance of OMCO algorithm by comparing the average CCTs with other two different schemes. We pay attention to investigate the impact of some important parameters on the performance of OMCO, such as coflow numbers, coflow density, coflow size and average inter-coflow arrival interval. To demonstrate the simulation results more intuitively, we normalized average CCTs of E-FIFO and E-SCF according to OMCO's average CCTs.



(a) Impact of coflow number

(b) Impact of coflow density

Fig. 4: Impact of Coflow parameter.

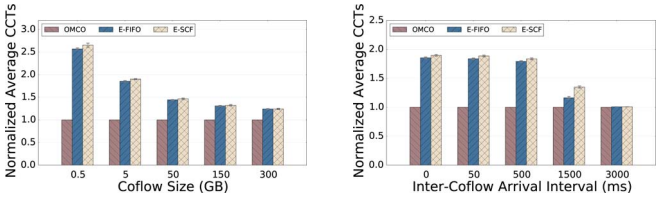
1) Coflow Number: Intuitively, the first parameter that affects the performance of coflow scheduling is the number of coflow. To evaluate its impact on different schemes, we remain the other parameter unchanged, *i.e.,* we set the coflow density, coflow size, and average inter-coflow arrival interval as 0.6, 1GB and 100ms, respectively. Then we change the number of coflows as the input of algorithm and calculate the average CCTs of OMCO, E-FIFO and E-SCF.

Fig.4(a) reflects that the advantage of OMCO is gradually expanding with the increase of coflow numbers. The reason for this phenomenon is that OMCO has more potential to reduce the circuit idle time, and improve the circuit utilization when more coflows were injected into the network. OMCO reduces the average CCTs by up to 45.8% and 46.9% when compared with E-FIFO and E-SCF.

2) Coflow Density: To investigate the influence of density on average CCTs, we fix the other related factors as follows. The number of coflows, the coflow size, mean inter-coflow arrival interval are set as 100, 1GB and 100ms, respectively. Then we inject coflows with different density into the network.

From the simulation results in Fig. 4(b), we can observe that the performance of OMCO is improved dramatically when increasing the coflow density. OMCO can reduce the average CCTs by up to 55.9% compared with E-FIFO, and up to 57.3% compared to E-SCF. The reason is that more dense coflows improve the chance to utilize the idle circuit, and always keep a high utilization of circuits. Besides, because higher density

means fewer unused ports for preemption, so E-FIFO and E-SCF suffer some performance loss with the increase of coflow density.



(a) Impact of coflow size (b) Impact of coflow arrival interval

Fig. 5: Impact of coflow parameter.

3) Coflow Size: In this experiment, we consider the impact of coflow size on different methods. We fixed the number of coflow, density of coflow and average inter-coflow arrival interval as 100, 0.6, and 100ms.

In Fig 5(a), OMCO always outperforms E-FIFO and E-SCF. OMCO reduces the average CCTs by up to 61.1% and 62.2% compared with E-FIFO and E-SCF. Differ from previous two parameters, the performance of OMCO gradually decreases with the increase of coflow size. This is because the benefit we receive from aligning and padding procedure is decreased. When the coflow size increases significantly, the flow size will increase accordingly. However, the reconfiguration delay δ is still unaltered, so the inserted artificial demands caused by aligning and padding is decreasing. In the meantime, less artificial demands mean less idle circuit for utilization. Consequently, OMCO could not give full play to its role. In fact, the small size coflow always account for a high proportion of the actual production environments, for example, the small size coflows account for 87.83% in [21].

4) Inter-Coflow Arrival Interval: We pay attention to the average inter-coflow arrival interval. To carry out simulations more precisely, we modify the mean intervals based on the original data traces. We set the number of coflow, the density of coflow, the size of coflow as 100, 0.6, 1GB, respectively.

The simulation results are shown in Fig. 5(b),. We adjust mean arrival interval from 0ms to 3000ms. When interval equals to 0ms, our problem is transformed into an offline coflow scheduling. OMCO can reduce the average CCTs by up to 46.1% and 47.2% compared with E-FIFO and E-SCF. Then, the performance of OMCO degrades slightly when interval grows up to 50ms. However, when interval becomes extremely large, the advantage of OMCO starts to disappear significantly, *i.e.,* when interval equals to 3s, OMCO has nearly same performance as E-FIFO and E-SCF. The reason is that if the interval is too large, the next coflow is still on its way when the last coflow is finished. Therefore, any scheduling strategy would be disabled and to schedule coflow one by one is the only approach.

V. Conclusion

The combination of coflow and optical circuit switch is a new direction to improve the data transmission efficiency in data center networks. In this paper, we attach importance to the multiple coflows scheduling problems in OCS in an online circumstance, and we craft a heuristic algorithm OMCO to handle this problem effectively. Extensive simulations based on realistic workloads indicate that OMCO outperforms existing approaches. The theoretical analysis of the competitive ratio for OMCO is considered as our future work. In practice, the information of the flow is likely to be unknown unless its transmission is completed. Therefore, another interesting extension of this work is to study online scheduling of coflows without the knowledge of its flow size.

References

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

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

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

[4] H. Zhang, L. Chen, B. Yi, K. Chen, M. Chowdhury, and Y. Geng, "CODA: Toward Automatically Identifying and Scheduling Coflows in the Dark," in *ACM SIGCOMM*, 2016.

[5] Z. Qiu, C. Stein, and Y. Zhong, "Minimizing the Total Weighted Completion Time of Coflows in Datacenter Networks," in *ACM SPAA*, 2015.

[6] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *ACM SIGCOMM*, 2010.

[7] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," in *ACM SIGCOMM*, 2010.

[8] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," in *ACM SIGCOMM*, 2013.

[9] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky *et al.*, "Scheduling techniques for hybrid circuit/packet networks," in *ACM CoNEXT*, 2015.

[10] X. S. Huang, X. S. Sun, and T. E. Ng, "Sunflow: Efficient optical circuit scheduling for coflows," in *ACM CoNEXT*, 2016.

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

[12] Y. Zhao, K. Chen, W. Bai, C. Tian, Y. Geng, Y. Zhang, D. Li, and S. Wang, "RAPIER: Integrating Routing and Scheduling for Coflow-Aware Data Center Networks," in *Proc. of IEEE INFOCOM*, 2015.

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

[14] L. Chen, W. Cui, B. Li, and B. Li, "Optimizing coflow completion times with utility max-min fairness," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[15] M. Shafiee and J. Ghaderi, "An improved bound for minimizing the total weighted completion time of coflows in datacenters," *arXiv preprint arXiv:1704.08357*, 2017.

[16] I. Gopal and C. Wong, "Minimizing the number of switchings in an ss/tdma system," *IEEE Transactions on Communications*, vol. 33, no. 6, pp. 497–501, 1985.

[17] C.-H. Wang, T. Javidi, and G. Porter, "End-to-end scheduling for all-optical data centers," in *IEEE INFOCOM*, 2015.

[18] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of mathematics*, vol. 17, no. 3, pp. 449–467, 1965.

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

[20] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce online," in *Proc. of USENIX NSDI*, 2010.

[21] M. Chowdhury, "Coflow benchmark based on facebook traces," https://github.com/coflow/coflow-benchmark.