Pas de deux: Shape the Circuits, and Shape the Apps too!

Hong Zhang[1], Kai Chen[1], Mosharaf Chowdhury[2]

[1]SING Lab, Hong Kong University of Science and Technology [2]University of Michigan

Abstract

Despite continued efforts toward building high bandwidth, low cost datacenter networks with reconfigurable optical fabrics, the impact of optical networks on datacenter applications has received little attention. Given the constraints of optical networks and the semantics of datacenter applications, we believe the network-application intersection to be the next innovation hotspot. In this paper, we specifically focus on data-parallel applications for two primary reasons: they are a natural fit to exploit high bandwidth optical fabrics, and they often form structured communication patterns or coflows.

We show that configuring circuits in reaction to changing traffic patterns is not enough. Efficient scheduling of even a single coflow in optical networks should be a *"Pas de deux"*[1] – *a joint shaping of not only the underlying circuit, but also the application's traffic demand.* Our preliminary evaluation with a production trace shows that joint shaping is on average within 1.18× of the optimal and performs 30% better than solutions that configure circuits in application-agnostic fashions. We further extend our analysis to inter-coflow scheduling and propose a layered solution that jointly considers circuit reconfiguration, coflow prioritization, as well as flow rate and route assignments.

CCS Concepts

• **Networks → Traffic engineering algorithms**; **Network architectures**;

Keywords

Data-intensive applications; optical switching; data center

ACM Reference format:
Hong Zhang, Kai Chen, Mosharaf Chowdhury. 2018. Pas de deux: Shape the Circuits, and Shape the Apps too!. In *Proceedings of 2nd Asia-Pacific Workshop on Networking, Beijing, China, August 2–3, 2018 (APNet '18),* 7 pages.
https://doi.org/10.1145/3232565.3232568

[1]A pas de deux is a dance duet in which two dancers perform ballet steps together.

1 Introduction

In recent years, a growing body of datacenter architecture designs [3, 4, 10, 11, 15] have proposed interconnecting Top-of-Rack (ToR) switches with optical fabrics. Although these designs provide high bisection bandwidth, low power consumption, and reduced cabling complexity, they suffer from high circuit reconfiguration delay of up to tens of milliseconds [4]. Yet, it is still unclear how they affect the end-to-end performance of datacenter applications.

Scheduling applications on the state-of-the-art optical architecture [4, 11] differs from previous circuit scheduling problems [2, 12, 13] in the following three aspects:

- **Application semantics**: Most existing circuit scheduling algorithms aim to maximize the throughput of an aggregated traffic matrix. However, this may not directly benefit application-level communication performance [7] because each application can have its own traffic matrix and its own performance objective. The coflow abstraction [5] captures such semantics of communication between two groups of machines in successive computation stages.
- **Traffic pattern**: Most existing circuit scheduling designs assume that the traffic demand matrix is *sparse* [2, 15]. However, many coflows involve wide-spread communication among thousands of machines [7].
- **Connectivity:** Most previous work adopt a network model, where one ToR can only establish one exclusive circuit connection to another ToR. However, to support wide-spread communication patterns, recent optical datacenter architecture designs [4, 11] allow increased connectivity among ToRs. For example, one ToR can simultaneously establish tens (ProjecToR [11]) to hundreds (MegaSwitch [4]) of circuit connections to other ToRs.

Overall, the presence of coflows in data-parallel applications and the advances in optical designs greatly complicate the circuit scheduling problem. Minimizing the average coflow completion time (CCT) requires coordinated scheduling of flows within each coflow as well as prioritized scheduling among multiple coflows. The rich connectivity among ToRs further increases the solution space.

In this paper, we start by investigating an important special case that involves serving a single coflow on optical fabrics. We observe that although the rich connectivity brings new opportunity to serve coflows without reconfiguration, obtaining a circuit configuration that well matches a coflow's traffic pattern is not always possible. Such mismatch between the coflow and circuit structure greatly increases the CCT (§3.1).

To alleviate this mismatch, we instead ask the following question: *can we reshape the coflow traffic demand matrix to match the circuit configuration?* In particular, we notice that the demand matrix can be effectively reshaped by rerouting traffic via other ToRs as relays. Multi-hop rerouting is not new for optical networks. However, unlike previous work [3] that employ multi-hop routing to provide overall connectivity, we leverage a fine-grained, coflow-specific multi-hop routing scheme to redistribute the original coflow traffic demand among ToRs (§3.2). However, while reshaping coflows can help in mitigating the mismatch and improving CCT, its effectiveness depends on underlying circuit configurations.

Given that neither coflow nor circuit (re)shaping alone efficiently minimizes CCT, we reformulate the problem as *a joint shaping of both circuit configuration and coflow traffic demand*. We show through examples that such joint scheduling brings extra flexility to find a good matching, and effectively minimizes the CCT (§3.3). Moreover, we design a heuristic that effectively calculates the joint scheduling by iteratively: (i) releasing under-utilized circuits by reshaping the coflow demand; (ii) configuring the released circuits to serve the bottleneck source-destination ToR pair. Preliminary evaluation with a production trace shows that joint shaping is on average within 1.18× of the optimal and performs 30% better in comparison to algorithms that only configure circuits (§4).

We conclude by extending our analysis to the general case of inter-coflow scheduling (§5). Specifically, we find each of the following factors: circuit configuration, routing, rate allocation, and coflow prioritization to play important roles, leading to a joint scheduling problem with exponential complexity. To perform efficient scheduling with reasonable complexity, we design a framework that effectively decouples the scheduling into three successive steps – circuit configuration, coflow permutation, and rate/routing. Each step is further transformed into the single coflow scheduling problem we have already formulated. We leave the in-depth evaluation under the general case as future work.

2 Background

2.1 The Network Model

We consider a network model where the entire datacenter fabric is abstracted out as one non-blocking core interconnecting all the n ToRs. Each ToR has k sending and receiving ports. ToRs can send and receive through these k different ports simultaneously by establishing a directed inter-port connection to any other ToRs, with each inter-port connection carries a fixed capacity of B. A *feasible* circuit configuration can be presented as a $n \times n$ matrix M, where $M_{ij} \in \{0, 1, \ldots k\}$ is the number of inter-port connections from ToR i to ToR j. Note that the configuration is also constrained by the sending and receiving port number k per ToR, where we have $\sum_{j=1}^{n} M_{ij} \leq k$ and $\sum_{i=1}^{n} M_{ij} \leq k$ for all i and j. We denote the set of all *feasible configurations* as $\{\mathbf{M}\}$. Moreover, the inter-port connection can be reconfigured with a reconfiguration delay δ during which the switch cannot carry any traffic. Such δ can range from tens of microseconds to few milliseconds depending on the technology [2].

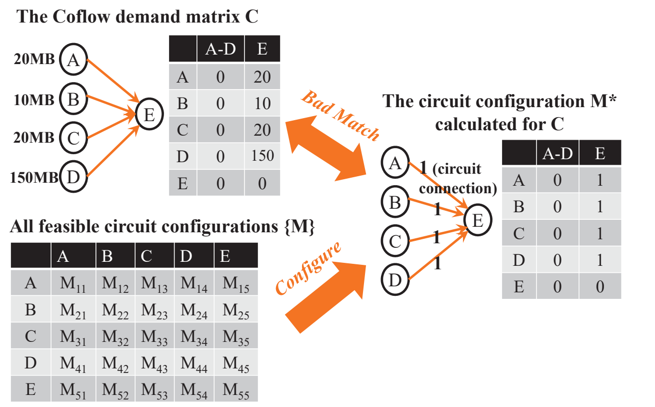
One major difference with previous models lies in the choice of k. Most prior work assumes $k = 1$, which means that one ToR can only establish one exclusive circuit connection to another ToR. However, to support wide-spread communication patterns, optical architecture designs are evolving with richer connectivity among ToRs. For example, ProjecToR [11] supports at least tens of lasers and photodetectors (corresponds to sending and receiving ports respectively) under each rack to handle fan-in/fan-out traffic. MegaSwitch [4] supports all-to-all communications among more than 30 ToRs (due to WSS port number limitation) simultaneously with k as large as 192. Consequently, in our model k can be a large number comparable to n.

2.2 The Traffic Model

In this paper, we focus on the coflow traffic model. Unlike the traditional flow abstraction, a coflow captures a collection of flows between two groups of machines in successive computation stages, where the communication stage finishes only after *all* the flows have completed [7]. A typical example of coflow is the shuffle between mappers and reducers in MapReduce [9]. More specifically, a coflow can be represented as an $n \times n$ traffic demand matrix C, where each element C_{ij} indicates the demand from ToR i to ToR j.

3 The Single Coflow Case

This section investigates the single coflow case. We simplify our discussion by restricting to only one circuit configuration for each coflow. Such assumption is reasonable in many cases because (1). optical fabric with rich connectivity usually have very large reconfiguration delays (i.e., ~20ms in [4]); (2). the rich connectivity allows serving a coflow with no reconfiguration. We extend our analysis to multiple coflows and remove the no-reconfiguration assumption in Section 5.


The Coflow demand matrix C
20MB
10MB
20MB
150MB
A
B
C
D
E
Bad Match
The circuit configuration M* calculated for C
1 (circuit connection)
1
1
1
All feasible circuit configurations {M}
Configure

	A-D	E
A	0	20
B	0	10
C	0	20
D	0	150
E	0	0

	A	B	C	D	E
A	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
B	M_{21}	M_{22}	M_{23}	M_{24}	M_{25}
C	M_{31}	M_{32}	M_{33}	M_{34}	M_{35}
D	M_{41}	M_{42}	M_{43}	M_{44}	M_{45}
E	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}

	A-D	E
A	0	1
B	0	1
C	0	1
D	0	1
E	0	0

Figure 1: [Example 1] Circuit configuration M^* calculated via circuit shaping (Problem 1) cannot well match the coflow demand matrix C. We can see that ToR pair (D, E) has 150MB demand while it is only assigned 1 circuit. Such bottleneck pair leads to a CCT of 150ms (calculated by Eq (1)), which is 3× the optimal CCT in packet switched networks (calculated by Eq (2)).

3.1 Shaping the Circuit

Similar to most existing circuit scheduling algorithms, we first try to efficiently configure the circuit to match the coflow traffic pattern. This directly translates to the following circuit shaping problem:

Problem 1. *(The circuit shaping problem) Given a coflow demand C and a set of feasible circuit configurations $\{\mathbf{M}\}$, find a configuration $M^* \in \{\mathbf{M}\}$ that minimizes the CCT.*

Note that the CCT can be calculated as

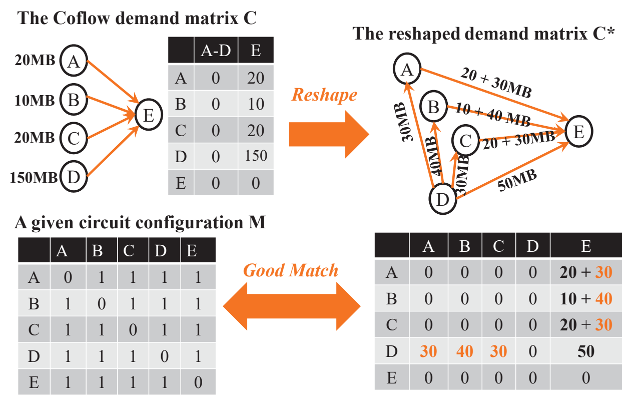
$$CCT = \max_{i,j}\left(\frac{C_{ij}}{M_{ij} \cdot B}\right) \qquad (1)$$

The optimal solution to Problem 1 can be obtained by first connecting all pairs (i, j) with $C_{ij} > 0$, and then iteratively adding connections to the pair (i^*, j^*) with the longest completion time.

However, we find that even with a rich port count, the optimal CCT achieved via circuit shaping can be far from the optimal CCT achievable in packet switched networks. To illustrate this, consider the example in Figure 1. The optical network contains 5 ToRs, each with 4 sending/receiving ports. Each inter-port connection carries a fixed capacity of 1GBps.The coflow demand C follows a many-to-one pattern.

We first calculate the optimal CCT achievable in packet switched networks with the same ingress/egress capacity per ToR. Note that this serves as the lower bound of the CCT achievable in the optical network. In such case, the CCT is constrained only by the ToR with the largest ingress/egress data volume, more specifically we have

$$CCT^{OPT} = \max\left(\max_i \frac{\sum_j C_{ij}}{B \cdot k}, \max_j \frac{\sum_i C_{ij}}{B \cdot k}\right) \qquad (2)$$


The Coflow demand matrix C
20MB
10MB
20MB
150MB
A
B
C
D
E
Reshape
The reshaped demand matrix C*
20 + 30MB
10 + 40 MB
20 + 30MB
30MB
40MB
30MB
50MB
A given circuit configuration M
Good Match

	A-D	E
A	0	20
B	0	10
C	0	20
D	0	150
E	0	0

	A	B	C	D	E
A	0	1	1	1	1
B	1	0	1	1	1
C	1	1	0	1	1
D	1	1	1	0	1
E	1	1	1	1	0

	A	B	C	D	E
A	0	0	0	0	20 + 30
B	0	0	0	0	10 + 40
C	0	0	0	0	20 + 30
D	30	40	30	0	50
E	0	0	0	0	0

Figure 2: [Example 2] Given a full-mesh circuit configuration M, we can reshape the coflow demand C to C^* to well match the configuration via coflow shaping (Problem 2). We achieve a CCT of 50ms following this schedule (calculated by Eq (1)), which is identical to the optimal CCT in packet switched networks.

where $B \cdot k$ is the total egress/ingress capacity per ToR. For coflow C in Figure 1, a CCT of 50ms can be achieved by allocating the 4GBps ingress capacity of ToR E among senders A to D proportionally to their demand.

Compared to the rate allocation that can take any fractional value, the circuit configuration M_{ij} can only be integers. For example, a sender can arbitrarily share its egress capacity among different receivers in packet switched networks; in optical networks, however, there is no way to split one sending port into two halves and connecting them to two receiving port simultaneously. Such integrality constraints result in mismatches between coflow demands and circuit configurations. On the one hand, ToR pairs with small traffic demand are over-provisioned and can hardly utilize the capacity assigned to them. In Figure 1, although ToR pair (A, E) only has a 20MB traffic demand, we still have to establish a circuit connection for it. As a result, the traffic demand of (A, E) is satisfied after 20ms, and the circuit then becomes idle for the rest of the time.[2] On the other hand, ToR pairs with high traffic demand suffer from under-provisioning, which greatly increases the CCT. For example, ToR pair (D, E) has a 150MB demand, but it is only assigned with one connection and takes 150ms to complete. Because *a coflow finishes only after all the traffic demand is served*, (D, E) becomes the bottleneck and increases the CCT to 150ms.

3.2 Shaping the Coflow

To address such mismatch, we explore in the reverse direction and try to reshape the traffic demand matrix to match the circuit configuration. Following this idea, we leverage multi-hop

[2]The connection can be reconfigured after the demand is satisfied if we relax the no-reconfiguration model. However, such reconfiguration still leads to low circuit utilization by introducing large reconfiguration delay (i.e., ∼20ms in [4]).

routing to reshape the traffic demand. Unlike previous work [3] that employs multi-hop routing to provide overall connectivity among ToRs, we leverage multi-hop routing to reshape the traffic demands to better match the circuit configuration. Also, note that the multi-hop routing is done in a cut-through way via source routing, rather than in a store-and-forward manner [2]. Doing so avoids circuit reconfiguration and extra buffer demand. Specifically, the reshaping is composed of a set of deliberate routing schemes to redistribute the original coflow traffic demand among ToRs.

Figure 2 illustrates the effectiveness of such reshaping. Given a circuit configuration M which forms a full-mesh connection, we can reshape the coflow C to match the configuration M. More specifically, consider the bottleneck ToR pair (D, E) with 150MB demand. We can effectively resolve it by rerouting these traffic through relaying nodes A, B and C. By doing so, we are actually reshaping the traffic demand from C to C^*. We can see that the schedule in Figure 2 results in a CCT of 50ms, which is identical to the best solution in packet switched networks.

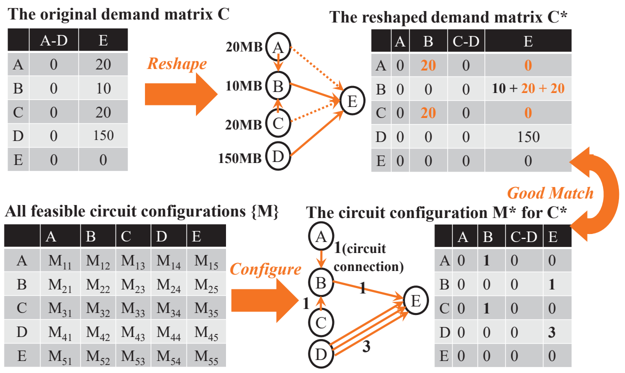
Formally, we define coflow shaping as follows:

Problem 2. *(The coflow shaping problem) Given a coflow demand C and a fixed circuit configuration M, determine the C^* which can be reshaped from C by multihop routing, so that the CCT (by Eq (1)) is minimized.*

We further translate this problem into the multi-commodity flow problem [8], where each ToR pair (i, j) corresponds to a commodity in the multi-commodity flow problem with demand C_{ij}. The goal is to maximize z, so that at least z percent of each demand is transferred within a unit time. Note that CCT equals $1/z$, and the problem can be solved in polynomial time using LP or fast approximation algorithms.

However, coflow reshaping also suffer from major drawbacks. First, it is not always possible to approach the optimal CCT via coflow reshaping given an arbitrary configuration. For example, if the circuit connection forms two closed cycles: $(A \rightarrow B, B \rightarrow C, C \rightarrow A)$ and $(D \rightarrow E, E \rightarrow D)$, then there is no way to satisfy the coflow demand via coflow reshaping. As a result, the effectiveness highly depends on the underlying circuit configuration. Moreover, notice that multi-hop connection may also result in an inefficient use of the circuit. As we see in Figure 2, 100MB traffic goes through two-hop paths. As a result, these traffic takes network resource which can potentially serve 200MB traffic demand via direct connection. Such low efficiency may result in reduced goodput and enlarged CCT in the multi-coflow case.

Implementation Feasibility Enabling fine-grained traffic reshaping requires flow-level routing enforcement. We find source routing a good candidate, which pre-installs all the routing rules needed on the ToR switches and does not require the costly dynamic routing table configuration. Specially, we


The original demand matrix C
Reshape
20MB
10MB
20MB
150MB
The reshaped demand matrix C*
Good Match
All feasible circuit configurations {M}
Configure
The circuit configuration M* for C*
1(circuit connection)
1
1
3

	A-D	E
A	0	20
B	0	10
C	0	20
D	0	150
E	0	0

	A	B	C-D	E
A	0	20	0	0
B	0	0	0	10 + 20 + 20
C	0	20	0	0
D	0	0	0	150
E	0	0	0	0

	A	B	C	D	E
A	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
B	M_{21}	M_{22}	M_{23}	M_{24}	M_{25}
C	M_{31}	M_{32}	M_{33}	M_{34}	M_{35}
D	M_{41}	M_{42}	M_{43}	M_{44}	M_{45}
E	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}

	A	B	C-D	E
A	0	1	0	0
B	0	0	0	1
C	0	1	0	0
D	0	0	0	3
E	0	0	0	0

Figure 3: [Example 3] In joint shaping (Problem 3) we shape both the circuit M^* and the coflow C^* to match each other. We achieve a CCT of 50ms following this schedule (calculated by Eq (1)), which is identical to the optimal CCT in packet switched networks. And it introduces much less multi-hop traffic compared to Example 2.

restrict the pre-installed paths to have a hop count less than a threshold p.[3] These paths are desirable because of their short lengths, and such restriction greatly reduces the number of routing entries needed. Note that the coflow demand between a ToR pair often consists of many TCP connections; thus we can distribute these flows over the desirable paths to match the coflow reshaping plan accordingly.

3.3 Shaping the Circuit and Coflow Together

Given that neither coflow nor circuit shaping alone can efficiently solve the problem, we focus on jointly shaping both circuit configuration and coflow traffic demand. More formally, we redefine the coflow scheduling problem as the following joint shaping problem:

Problem 3. *(The joint shaping problem) Given a coflow demand C and a set of feasible circuit configurations $\{\mathbf{M}\}$, determine the C^* which can be reshaped from C by multi-hop routing and the configuration $M^* \in \{\mathbf{M}\}$, so that the CCT (by Eq (1)) is minimized.*

As the example in Figure 3 shows, joint scheduling brings extra flexility to find a good match. Compared to circuit shaping, joint shaping is not constrained by the integrality constraint. Compared to coflow shaping, the effectiveness of joint shaping is not constrained by the given circuit configuration. Moreover, such flexibility also enables more efficient utilization of network resources. As we see in Figure 3, we achieve the optimal CCT of 50ms while introducing much less multi-hop traffic compare to coflow shaping (Figure 2).

However, solving the joint shaping problem is non-trivial because the reshaping of demand matrix C and circuit configuration M should be tightly coupled. We design a heuristic

[3]This constraint is also considered when calculating the coflow shaping.

Algorithm 1 Heuristic for the Joint Shaping Problem

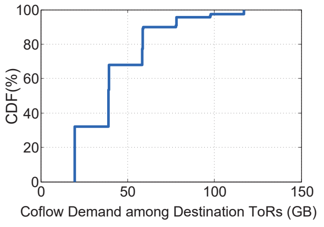
Input: M: the initial circuit configuration calculated by the circuit shaping; C: the initial coflow demand matrix;
Output: C^*: the reshaped coflow demand matrix (with the corresponding routing scheme); M^* the updated circuit configuration;

```
1:  procedure MAIN
2:      $M^* = M, C^* = C$;                                  ▷ Initialization;
3:      while true do                 ▷ Iteratively resolving the bottleneck;
4:          Calculate the CCT and Identify the bottleneck pair $(s, d)$.
5:          Release_from_S($C^*, M^*, CCT$): Release a sending port of $s$;
6:          Release_towards_D($C^*, M^*, CCT$): Release a receiving port of $d$;
7:          if both the above subroutines return true then
8:              $M^*_{sd} = M^*_{sd} + 1$;   ▷ Add one link for pair $(s, d)$ by connecting the released two ports;
9:              Reshape coflow demand matrix $C^*$ accordingly;
10:         else
11:             Return $M^*$ and $C^*$;
12: procedure RELEASE_FROM_S($C^*, M^*, CCT$)
13:     Select the currently least loaded link $(s, u)$ from ToR $s$;
14:     From all the destination ToRs from $s$:
15:     Select a relay node $r$ which minimizes $T = \max\big((C_{sr} + C_{su})/(M_{sr} \cdot k), (C_{ru} + C_{su})/(M_{ru} \cdot k)\big)$ ▷ Shift the load of $(s, u)$ to $\big((s, r), (r, u)\big)$ via the least loaded relay node $r$;
16:     if $T < CCT$ then          ▷ Determine if reshaping reduces CCT
17:         Return true;
18: procedure RELEASE_TOWARDS_D($C^*, M^*, CCT$)
19:     Done in a similarly way as Release_from_S().
```

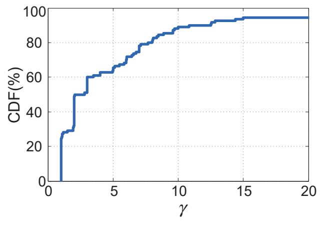
that effectively calculates the joint scheduling by iteratively: (i) releasing under-utilized circuits by reshaping the coflow demand; and (ii) configuring the released circuits to serve the bottleneck source-destination ToR pair. We now describe the main idea by using Example 3. We start from the configuration resulted from the circuit shaping problem (M in Figure 1) and identify the bottleneck ToR pair that determines the CCT ((D, E) in Figure 1). We then try to release the under-utilized connections which share the same sender/receiver with the bottleneck ToR pair (connection $A \longrightarrow E$ in Figure 1). This can be done by rerouting the traffic on these under-utilized connections via multi-hop paths composed of other under-utilized links (reshape the demand (A, E) to (A, B) and (B, E) as shown in Figure 3). After that, we assign the released ports to the bottleneck ToR pair (add an extra connection from D to E) so that the CCT is reduced. We run the above procedure iteratively until the CCT cannot be reduced anymore. Algorithm 1 describes this in more detail.

4 Preliminary Evaluation

Evaluation Settings We use a realistic workload based on a one-hour Hive/MapReduce trace collected from a Facebook production cluster [1]. The trace contains more than 500



(a) Traffic distribution within one coflow



(b) CDF of γ

Figure 4: [Evaluation] Traffic pattern within a coflow can be very biased.

Coflows observed in a datacenter with 150 ToRs. For the network model, we assume that each ToR has 192 available ports (i.e., $k = 192$), which is the case for MegaSwitch [4]. We set each inter-port connection to carry 1Gbps capacity.

Understanding the Coflow Structure We first study how the traffic is distributed within a coflow in the Facebook trace. Note that such coflow structure is directly related to the performance of the scheduling algorithm. For example, in Example 1 if all the flows of coflow C have the same size, then circuit shaping also achieves the optimal CCT. In particular, the mismatch in circuit shaping tends to be more severe when the flow size within a coflow follows a biased distribution.

Observation from the Facebook trace validates that the traffic pattern within a coflow can be very biased. To show this, we first select a large coflow with widespread communication among more than 120 ToRs and inspect its structure. Figure 4a demonstrates how the traffic is distributed among destination ToRs — 25% of the ToRs receive less than 2GB data, while another 25% receives more than 6GB.

To show that such a coflow is no corner case, we define γ as the ratio between the maximum and minimum traffic demand within a coflow among all destination ToRs. Figure 4b shows the distribution of γ for all coflows with wide-spread communication pattern[4]. We see that for 40% of the coflows, some ToR receives 3× data compared to some others. And 20% of the coflows experience a larger bias of 8×.

Effectiveness of Joint Shaping We implement a flow-level simulator to evaluate the effectiveness of the joint shaping heuristic in the simplified single coflow case. We evaluate three algorithms: the optimal solution (calculated by Equation 2), the circuit shaping and the joint shaping. For easy comparison, we normalize the achieved CCT of circuit shaping and joint shaping over the optimal solution, i.e.,

$$\text{Normalized Comp. Time} = \frac{\text{Compared CCT}}{\text{Optimal CCT}}$$

[4] Coflows which contain more than 100 subflows with different source and destination.

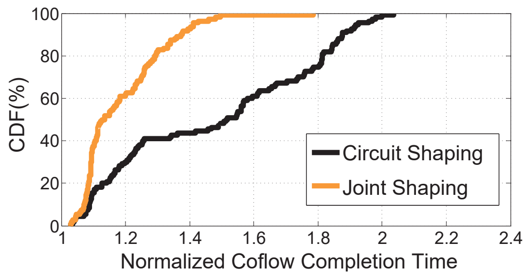


Figure 5: [Evaluation] CDF of the normalized CCT. We see that joint shaping significantly outperforms the circuit shaping algorithm.

Smaller values indicate better performance, and a normalized completion time close to 1 indicates comparable performance to the optimal solution.

Figure 5 presents comparative CDFs of CCTs for all coflows with wide-spread communication patterns. We first notice that circuit shaping performs poorly compared to the optimal solution. In particular, 40% of the coflows finish 1.6× slower than the optimal. This is because under the biased coflow structure, the mismatch between the circuit configuration and the coflow structure becomes severe. Note that the performance of circuit shaping is also the best achievable performance for all algorithms with direct routing (e.g., Sunflow [12], Solstice [13]) under our evaluation settings.

Moreover, we observe that the simple heuristic (Algorithm 1) for joint shaping approaches the optimal solution, and significantly outperforms the circuit shaping algorithm. Specially, 40% of the coflows finish within 1.1× of the optimal, and 80% are within 1.3×. We also calculate the average normalized CCT. The result shows that joint shaping is within 1.18× to optimal, and outperforms circuit shaping by 30%.

5 The General Case with Multiple Coflows

Coflow scheduling under the general case introduces new challenges. Specially, since different coflows may favor different circuit configurations, the optimal solution (which minimizes the average CCT) should be a sequence of schedules, where each schedule is characterized by a different circuit configuration, routing, rate allocation and duration. Compared to the case of coflow scheduling in packet switched networks (which is strongly NP-hard already [7]), the complexity of colfow scheduling in optical networks grows exponentially as we need to perform a joint scheduling of circuit configuration, routing/rate allocation, and coflow prioritization.

The key to perform efficient scheduling with reasonable complexity is to find a good way to decouple the problem. To this end, we break the scheduling problem into three successive steps – each determines the circuit configuration, permutation and rate/routing respectively. And each step is further transformed into the single coflow scheduling problems (i.e., Problem 1-3) formulated in Section 3. Similar to

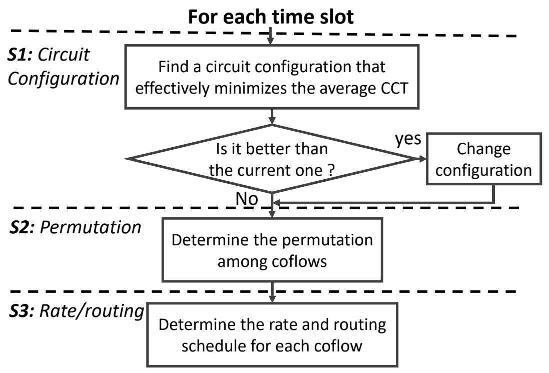

For each time slot
S1: Circuit Configuration
Find a circuit configuration that effectively minimizes the average CCT
Is it better than the current one ?
yes
Change configuration
No
S2: Permutation
Determine the permutation among coflows
S3: Rate/routing
Determine the rate and routing schedule for each coflow

Figure 6: Flow Chart of the Inter-Coflow Scheduling Framework.

prior work on coflow scheduling [6, 7, 16], we coordinate coflow information to the central controller at ∼100*ms* intervals, and run the inter-coflow scheduling framework (Figure 6) at the beginning of every scheduling interval. We briefly go through each step as follows.

Step 1: Circuit Configuration: Determine an appropriate circuit configuration for the current time slot is non-trivial. One intuitive idea is to calculate a circuit that best serves the aggregated demand of *all* current coflows. However, this is not necessarily a good idea, because the circuit can be reconfigured in the following timeslots. Since different coflows may favor different configurations, it is probably more efficient if we first configure the circuit to best serve the small coflows, and then reconfigure for the large coflows after the small coflows finish. As a result, there is no need to consider coflows which will be scheduled after a long time in the *current* circuit configuration. Another extreme is to always greedily optimize for the coflow with the highest priority. However doing so introduces too many reconfigurations, which also greatly enlarges the average CCT.

Considering the drawbacks of both extremes, we try to reach a sweet point in the middle – to optimize for the coflows that are likely to be scheduled in the next T timeslots. The choice of T serves as a flexible knob between the two extremes, and can be effectively tuned via evaluation. By doing so, we effectively excludes the impact of those far-away coflows, and at the same time reduces the reconfiguration frequency by jointly optimize for high priority coflows. To achieve this, we first find the m coflows that are likely to be scheduled in the next T timeslots. We then consider these m coflows as an aggregated coflow C_{agg}, and try to find a fixed configuration that well serves their aggregated demand. The problem is then transformed to the joint shaping problem (Problem 3), and can be solved using Algorithm 1.

Given the new configuration, we must decide whether to reconfigure or not regarding the reconfiguration cost. To do so, we estimate the CCT for each of the m coflows under the new/old configuration respectively[5]. Note that the CCT estimation can be described by the coflow shaping problem (Problem 2). We then make the decision based on the aggregated CCT of the m coflows.

Step 2: Permutation Given the optimality of the shortest-first policy in minimizing the average CCT, a natural choice would be directly adopt the Smallest-Effective-Bottleneck-First (SEBF) heuristic in Varys [7]. However, SEBF does not take into account the underlying circuit configuration when calculating the bottleneck. Note that this is no longer the case in optical networks, where the bottleneck always depends on the circuit configuration. Consequently, we extend the SEBF to CS (configuration-specific)-SEBF by further taking the circuit configuration and routing into consideration. More specifically, given the coflow demand C and the configuration M from Step 1, we calculate the minimum possible CCT Γ with coflow reshaping allowed. Note that this is again transformed to the coflow shaping problem (Problem 2). Coflows are then scheduled in the smallest-Γ-first order.

Step 3: Routing and Rate Allocation Given the circuit configuration (Step 1) and coflow order (Step 2), we can iteratively calculate the routing and rate allocation. More specifically, for each coflow in the smallest-Γ-first order, we calculate the allocation by solving the coflow shaping problem (Problem 2) with the current circuit configuration and residual bandwidth. We then update the residual bandwidth according to the allocation, and calculate for the next coflow.

6 Related Work

Coflow Scheduling in Packet Switched Networks Most existing coflow schedulers are designed for packet switched networks [6, 7, 16], while the optical fabric brings two major challenges. First, the circuit configuration introduces the integrality constraint, which serves as the major cause of the mismatch between coflow demand and circuit configuration as we show in Section 3.1. Moreover, since reconfiguration introduces non-negligible delay in optical networks, the scheduler has to decide whether to reconfigure the circuit after coflow arrival/departure. As shown in Section 5, this greatly enlarges the solution space and complicates the problem.

Circuit Scheduling We discuss several most related work in circuit scheduling. Sunflow [12] first studies the coflow scheduling problem in optical networks. However, it adopts the assumption that one ToR can only establish one connection simultaneously, which no longer holds in many current optical architecture designs [4, 11]. Moreover, Sunflow only leverages circuit shaping, which is not sufficient with rich connection among ToRs. Third, preemption is not allowed in Sunflow once the circuit is established. However we notice that it plays an important role in minimizing CCT for multiple coflows (§5). Eclipse [2] also leverages multi-hop routing for circuit scheduling, however it does not consider application semantics such as coflow. Moreover, Eclipse requires the relaying ToR to buffer the traffic and transmit in a subsequent configuration for multi-hop routing. Doing so requires huge extra buffer and adds at least one extra reconfiguration delay. In one word, neither Eclipse nor Sunflow answers our motivating question, that how we can best serve the application traffic demand in start-of-the-art optical datacenter fabrics.

RotorNet [14] proposes a scalable and low-complexity optical datacenter network design with a fully decentralized control plane. Unfortunately, the decentralized design cannot be easily extended to perform efficient coflow scheduling. The spatial structure of coflow naturally requires coordinated scheduling, and decentralized coflow scheduling remains an open problem even in packet switched networks [6].

[5]The CCTs for the new configuration include an extra reconfiguration delay δ

References

[1] Coflow Benchmark Based on Facebook Traces. https://github.com/coflow/coflow-benchmark.
[2] Shaileshh Bojja, Mohammad Alizadeh, and Pramod Viswanath. 2015. Costly circuits, submodular schedules and approximate Carathéodory theorems. In *SIGMETRICS*.
[3] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. 2012. OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility. In *NSDI*.
[4] Li Chen, Kai Chen, Zhonghua Zhu, Minlan Yu, George Porter, Chunming Qiao, and Shan Zhong. 2016. Enabling Wide-Spread Communications on Optical Fabric with MegaSwitch. In *NSDI*.
[5] Mosharaf Chowdhury and Ion Stoica. 2012. Coflow: An application layer abstraction for cluster networking. In *Hotnets*.
[6] Mosharaf Chowdhury and Ion Stoica. 2015. Efficient Coflow Scheduling Without Prior Knowledge. In *SIGCOMM*.
[7] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. 2014. Efficient coflow scheduling with Varys. In *SIGCOMM*.
[8] Thomas H Cormen. 2009. *Introduction to algorithms*. MIT press.
[9] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*.
[10] Nathan Farrington, Alex Forencich, George Porter, Pang-Chen Sun, Joseph Ford, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2013. A Multiport Microsecond Optical Circuit Switch for Data Center Networking. In *IEEE Photonics Technology Letters*.
[11] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel Kilper. 2016. ProjecToR: Agile Reconfigurable Data Center Interconnect. In *SIGCOMM*.
[12] Xin Sunny Huang, Xiaoye Steven Sun, and TS Eugene Ng. 2016. Sunflow: Efficient Optical Circuit Scheduling for Coflows. In *CoNEXT*.
[13] He Liu, Matthew K Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M Voelker, David G Andersen, Michael Kaminsky, et al. 2015. Scheduling techniques for hybrid circuit/packet networks. In *CoNEXT*.
[14] William M Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C Snoeren, and George Porter. 2017. RotorNet: A scalable, low-complexity, optical datacenter network. In *SIGCOMM*.
[15] Guohui Wang, David Andersen, Michael Kaminsky, Konstantina Papagiannaki, T. Ng, Michael Kozuch, and Michael Ryan. 2010. c-Through: Part-time Optics in Data Centers. In *SIGCOMM*.
[16] Hong Zhang, Li Chen, Bairen Yi, Kai Chen, Mosharaf Chowdhury, and Yanhui Geng. 2016. CODA: Toward automatically identifying and scheduling coflows in the dark. In *SIGCOMM*.