Scheduling Coflows in Datacenter Networks: Improved Bound for Total Weighted Completion Time

Mehrnoosh Shafiee and Javad Ghaderi
Department of Electrical Engineering
Columbia University

ABSTRACT

Coflow is a recently proposed networking abstraction to capture communication patterns in data-parallel computing frameworks. We consider the problem of efficiently scheduling coflows with release dates in a shared datacenter network so as to minimize the total weighted completion time of coflows. Specifically, we propose a randomized algorithm with approximation ratio of $3e \approx 8.155$, which improves the prior best known ratio of $9 + 16\sqrt{2}/3 \approx 16.542$. For the special case when all coflows are released at time zero, we obtain a randomized algorithm with approximation ratio of $2e \approx 5.436$ which improves the prior best known ratio of $3 + 2\sqrt{2} \approx 5.828$. Simulation result using a real traffic trace is presented that shows improvement over the prior approaches.

KEYWORDS

Scheduling Algorithms, Approximation Algorithms, Coflow, Datacenter Network

ACM Reference format:
Mehrnoosh Shafiee and Javad Ghaderi. 2017. Scheduling Coflows in Datacenter Networks: Improved Bound for Total Weighted Completion Time. In *Proceedings of SIGMETRICS '17, June 5–9, 2017, Urbana-Champaign, IL, USA,* , 2 pages.
DOI: http://dx.doi.org/10.1145/3078505.3078548

1 INTRODUCTION

Many data-parallel computing applications (e.g. MapReduce [3]) consist of multiple computation stages. Intermediate parallel data is often produced at various stages which needs to be transferred among servers in the datacenter network. A stage often cannot start or be completed unless all the required data pieces from the preceding stage are received. Hence, the collective effect of the data flflows between the stages is more important for the performance (e.g. latency, job completion time) than that of any of the individual flows. Recently Chowdhury and Stoica [1] have introduced the *coflow* abstraction to capture these communication patters. *A coflow is defined as a collection of parallel flows whose completion time is determined by the completion time of the last flow in the collection.*
A Smallest-Effective-Bottleneck-First heuristic was introduced in Varys [2] for the problem of coflow scheduling in order to minimize the weighted sum of completion times of coflows in the system.

This work is supported by NSF Grants CNS-1652115 and CNS-1565774.

SIGMETRICS '17, June 5–9, 2017, Urbana-Champaign, IL, USA

DOI: http://dx.doi.org/10.1145/3078505.3078548

Table 1: Performance Guarantees (Approximation Ratios)

Case	Best known	This paper
Deterministic, without release dates	8 [5]	8
Deterministic, with release dates	12 [5]	12
Randomized, without release dates	$3 + 2\sqrt{2}$ [5]	$2e$
Randomized, with release dates	$9 + 16\sqrt{2}/3$ [6]	$3e$

This problem has been shown [5, 6] to be NP-complete through its connection with the concurrent open shop problem [2, 6]. Table 1 summarizes the best known approximation ratio results for this problem [5, 6]. We propose a deterministic algorithm that gives the same bound as the prior best known bound, however, our approach improves the best known bound for the randomized algorithm in both cases of with and without release dates.

2 MODEL AND PROBLEM STATEMENT

Similar to [2, 6], we abstract out the datacenter network as one giant $N \times N$ non-blocking switch, with N input and N output links connected to N source and N destination servers, respectively. We assume all the link capacities are equal and normalized to one.
We assume there is a set of K coflows denoted by $\mathcal{K}$. Coflow $k \in \mathcal{K}$ can be denoted as an $N \times N$ matrix $D^{(k)}$ which is released at time r_k that means it can only be scheduled after time r_k. d_{ij}^k, the (i, j)-th element of the matrix $D^{(k)}$ is the size of flow (i, j, k). For a source node i and a coflow $k \in \mathcal{K}$, we define $d_i^k = \sum_{1 \le j \le N} d_{ij}^k$, which is the aggregate flow that node i needs to transmit for coflow k. d_j^k is defined similarly for destination node $j \in \mathcal{J}$ and coflow $k \in \mathcal{K}$. Moreover, $W(k) = \max\{\max_{1 \le i \le N} d_i^k, \max_{1 \le j \le N} d_j^k\}$ is defined as the effective size of coflow k.
We use f_k to denote the completion time of coflow k, which, by definition of coflow, is the time when all its flows have finished processing. In other words, for every coflow $k \in \mathcal{K}$, $f_k = \max_{1 \le i,j \le N} f_{ij}^k$, where f_{ij}^k is the completion time of flow (i, j, k).
For given positive weights w_k, $k \in \mathcal{K}$, the goal is to minimize the weighted sum of coflow completion times, i.e., $\sum_{k \in \mathcal{K}} w_k f_k$, subject to capacity and release dates constraints.

3 LINEAR PROGRAMING RELAXATION

In this section, we use *linear ordering variables* (see, e.g., [4]) to present a relaxed linear program of coflow scheduling problem. For any two coflows k, k', we introduce a binary variable $\delta_{kk'} \in \{0, 1\}$ such that $\delta_{kk'} = 1$ if coflow k is finished before coflow k', and it is 0 otherwise. In the linear program relaxation, we allow the ordering

variables to be fractional. This yields the following relaxed LP

$$\text{(LP)} \ \min \ \sum_{k=1}^{K} w_k f_k \tag{1a}$$

$$\text{subject to: } f_k \geq d_i^k + \sum_{k' \in \mathcal{K}} d_i^{k'} \delta_{k'k} \ \ 1 \leq i \leq N, k \in \mathcal{K} \tag{1b}$$

$$f_k \geq d_j^k + \sum_{k' \in \mathcal{K}} d_j^{k'} \delta_{k'k} \ \ 1 \leq j \leq N, k \in \mathcal{K} \tag{1c}$$

$$f_k \geq W(k) + r_k \ \ k \in \mathcal{K} \tag{1d}$$

$$\delta_{kk'} + \delta_{k'k} = 1 \ \ k, k' \in \mathcal{K} \tag{1e}$$

$$\delta_{kk'} \in [0,1] \ \ k, k' \in \mathcal{K}. \tag{1f}$$

The constraint (1b) (similarly (1c)) follows from the definition of ordering variables and the fact that flows incident to a source node i (a destination node j) are processed by a single link of unit capacity. The fact that each coflow cannot be completed before its release date plus its effective size is captured by constraint (1d). The next constraint (1e) indicates that for each two incident coflows, one precedes the other. We denote by $\tilde{f}_k$ the optimal solution to the (LP) for completion time of coflow $k \in \mathcal{K}$.

4 ALGORITHM DESCRIPTION AND MAIN RESULTS

Both deterministic and randomized algorithms have three steps: (i) solve the relaxed LP (1), (ii) use the solution of the relaxed LP to partition coflows into disjoint subsets $C_0, C_1, \cdots, C_L$ for some L to be determined, and (iii) treat each subset C_l, $l = 0, \cdots, L$, as a single coflow and schedule its flows in a way that optimizes its completion time.

Partition Rule: We define $\gamma = \min_{i,j,k} d_{ij}^k$ and $T = \max_k r_k + \sum_k \sum_i \sum_j d_{ij}^k$. Given $\beta \geq 1$ (to be optimized), we choose L to be the smallest integer such that $\gamma \beta^{L+\alpha} \geq T$, where in the deterministic algorithm, $\alpha = 0$ and in the randomized algorithm α is a number chosen uniformly at random from $[0, 1)$. Consequently, define $a_l = \gamma \beta^l$, for $l = -1, 0, 1, \cdots, L$. Then the l-th partition is defined as the interval $(a_{l-1}, a_l]$ in both algorithms and the set C_l is defined as the subset of coflows whose completion times $\tilde{f}_k$ fall within the l-th partition, i.e., $C_l = \{k \in \mathcal{K} : \tilde{f}_k \in (a_{l-1}, a_l]\}$; $l = 0, 1, \cdots, L$.

Scheduling Each Subset: To schedule coflows of each subset C_l, $l = 1, \cdots, L$, we construct a single coflow by aggregating all the coflows of C_l, and then schedule its flows to optimize the completion time of this aggregate coflow.

Suppose a single coflow $D = \left(d_{ij}\right)_{i,j=1}^{N}$ is given. Let $W(D)$ denote its effective size. We assign transmission rate $x_{ij} = d_{ij}/W(D)$ to flow (i, j) until it is completed. Note that this rate assignment respects the capacity constraints, since for all source nodes $i \in \mathcal{I}$:

$$\sum_{1 \leq j \leq N} x_{ij}(t) = \sum_{1 \leq j \leq N} \frac{d_{ij}}{W(D)} \leq \frac{W(D)}{W(D)} = 1,$$

and similarly for all destination nodes $j \in \mathcal{J}$. Also, by this rate assignment, all flows of D finish in $W(D)$ amount of time.

Now we state the main results in the following theorems. Due to space constraint, we skip the proofs of these theorems and refer the reader to the technical report of this work [7].

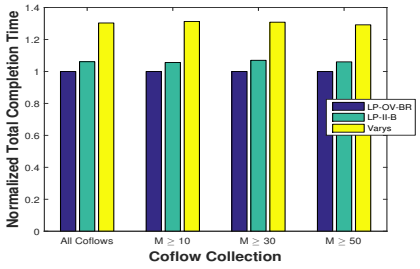

Figure 1: Performance of LP-OV-BR, LP-II-B, and Varys when all coflows release at time 0, normalized with the performance of LP-OV-BR, under real traffic trace.

Theorem 4.1. *When $\beta = 2$, the described deterministic algorithm is a 12-approximation algorithm for the problem of total weighted completion time minimization of coflows with release dates. Without release dates, it is an 8-approximation algorithm.*

Theorem 4.2. *When $\beta = e$, the described randomized algorithm is a $(3e)$-approximation algorithm. If all coflows are released at time 0, then it is a $(2e)$-approximation algorithm.*

5 EMPIRICAL EVALUATION

Now, we present our simulation result.

Workload: The workload is based on a Hive/MapReduce trace at Facebook that was collected from a 3000-machine cluster with 150 racks and was also used in [2, 6]. Similar to [6], we filter the coflows based on *the number of their non-zero flows which we denote by M*. Apart from considering all coflows ($M \geq 0$), we consider three coflow collections filtered by the conditions $M \geq 10$, $M \geq 30$, and $M \geq 50$. We assume that all coflows are released at time 0.

Algorithms: We combine backfilling strategy (see, e.g., [2, 6] with our algorithm and the algorithm in [6] and refer to them as 'LP-OV-BR' and 'LP-II-B', respectively. We also simulate Varys [2].

Figure 1 shows the performance of different algorithms for different collections of coflows when all coflows have equal weights. LP-OV-BR outperforms Varys by $28 - 32\%$ in different collections. It also constantly outperforms LP-II-B by $6 - 7\%$.

More simulation results are presented in the technical report [7].

REFERENCES

[1] Mosharaf Chowdhury and Ion Stoica. 2012. Coflow: A networking abstraction for cluster applications. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. ACM, 31–36.

[2] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. 2014. Efficient coflow scheduling with Varys. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 443–454.

[3] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (2008), 107–113.

[4] Rajiv Gandhi, Magnús M Halldórsson, Guy Kortsarz, and Hadas Shachnai. 2008. Improved bounds for scheduling conflicting jobs with minsum criteria. *ACM Transactions on Algorithms (TALG)* 4, 1 (2008), 11.

[5] Samir Khuller and Manish Purohit. 2016. Brief Announcement: Improved Approximation Algorithms for Scheduling Co-Flows. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*. ACM, 239–240.

[6] Zhen Qiu, Cliff Stein, and Yuan Zhong. 2015. Minimizing the total weighted completion time of coflows in datacenter networks. In *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*. ACM, 294–303.

[7] Mehrnoosh Shafiee and Java Ghaderi. 2017. *Scheduling Coflows in Datacenter Networks: Improved Bound for Total Weighted Completion Time*. Technical Report. http://www.columbia.edu/~ms4895/technical%20report/coflowsig.pdf.