Efficient Coflow Scheduling with Varys[1]

Reviewed by Han Zhang and Ryan Marcotte

Summary

This work explores the problem of inter-coflow scheduling with the dual objectives of minimizing average Coflow Completion Time (CCT) and meeting temporal deadlines for each coflow. Efficient coflow scheduling is challenging in large part due to the significant variations that exist in the characteristics (length, width, size, skew) of real-world coflows. The authors propose an ordering heuristic (SEBF) that schedules a coflow based on the completion time of its bottleneck as well as an algorithm (MADD) that allocates rates to individual flows, minimizing overall bandwidth usage of the coflow. They incorporate these elements into a fully operational coflow scheduling system (Varys). In experiments driven by traffic traces from large-scale datacenters, this work shows an up to 3.16 times reduction in CCT while allowing 2 times more coflows to meet their deadlines, comparing against the performance of per-flow mechanisms.

Novelty/Contribution

1. Demonstrating the necessity of study with real world data

The authors continually motivate the utility of their proposed methods, laying out through their exposition a strong case for why existing schedulers are not well-suited for inter-coflow scheduling. More significant than the arguments they make for this, though, are the data presented on the subject in Section 4. In this section, the authors use actual data to back up their claim of inter-coflow scheduling being challenging due to the widely varying characteristics of production coflows. For example, the data shows that wide coflows (>1 million flows) coexist with a significant number of narrow coflows (60 percent have at most 50 flows). We believe that this is a very important part of this work as it helps demonstrate the difficulty of the problem of

[1] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with Varys," in *SIGCOMM*, 2014, vol. 44, no. 4, pp. 443–454.

scheduling such heterogeneous coflows. Furthermore, it convinces the reader of the value of the proposed method if it can provide an effective solution.

2. Categorization of Coflows into Bins

During the performance evaluation, the authors categorize coflow traces into 4 bins based on their characteristics. As mentioned above, coflows vary significantly in dimensions due to their different traffic patterns and application cases. In Section 7 the authors run performance analysis with respect to different bins and discover that Varys has divergent performances for each bin. It could have been the case that jumping directly to the conclusion that Varys improves the coflow scheduling based on the overall performance without looking into the performance measurement across different coflows. However, categorizing coflows into different bins and analyzing each separately help revealing the hidden values of Varys to each individual groups and it is interesting to find out that different bins could benefit variously.

3. “One Big Switch” Abstraction of Datacenter Networks

The paper leverages the abstraction of entire datacenter fabric as a big switch to simplify the network model. The elegance of this simplification immediately impressed us the first time we read these lines. As the paper has pointed out, this abstraction earns its practicality because of recent advances in datacenter networks development. It is noteworthy to point out, as the authors have already addressed in the context, that this abstraction is not too much of a novelty but rather an existing and reasonable approximation. We could definitely make use of the “one big switch” or similar useful abstraction in future research to simplify our modeling processes, as long as we could justify our argument with sufficient real world measurements and facts.

Possible Improvement or Extensions

1. Relaxation of Constraints

In Section 5, the authors make a rather curious argument about the desirable properties of a scheduler. They state that in addition to satisfying its primary objective (presumably this is minimizing CCT), the scheduler should prevent starvation and minimize usage of available resources. Intuitively, these properties do seem to correlate with an ideal scheduler. However, the authors describe these properties as constraints that should be satisfied, which we see as a mischaracterization. Absent any additional information about the inherent value of satisfying these properties in particular, these properties seem more like probable *byproducts* or *effects* of an ideal scheduler, rather than constraints. For example, unless there is an explicit cost incurred for using resources, minimizing resource utilization is likely only useful insofar as it helps reduce the primary objective of the scheduler, namely reducing CCT. In other words, it is possible that two schedulers with the same level of resource utilization could yield different CCT performance. If such cases do exist, it seems far better to focus optimization on reducing CCT rather than incorporating a constraint on resource utilization. It should be noted that the authors later make an additional vague statement that “letting resources idle . . . can hurt performance in the online case” of rate allocation, but they provide no further argument to back up this claim.

2. “Procrastination” of Scheduler

The authors argue that when seeking to guarantee coflow completion with a deadline, “completing . . . bottlenecks as fast as possible has no benefits.” While this assumption is important in the presented rate allocation algorithm (MADD), we question whether there can truly be “no benefits” to scheduling early completion of bottlenecks. Allocating the minimum rate needed to meet a deadline is certainly optimal in a deterministic setting, but what happens if there is some uncertainty in the system? Such optimal scheduling would not be robust in an uncertain or stochastic environment because it would leave no margin for error. In that case, scheduling so as to complete bottlenecks ahead of their deadline might have some benefit. The experimentation in this paper seems to support this idea, since a quarter of admitted coflows in the EC2 experiment fail to meet their deadlines. The authors actually cite uncertainties in estimating utilizations as a reason for these failures. All of this points to there being some possible benefits of early bottleneck completion, contrary to the statement of the authors.

3. Heterogeneous Coflow Scheduling

The authors explicitly state that they “do not use Varys for coflows with bottlenecks smaller than 25 MB” because the batching of control messages in large time intervals and coordination overheads impairs the performance of small coflows. In the performance evaluation the authors have shown that per-flow fairness outperforms Varys in “tiny, subsecond coflows.” This fact, along with the CDF of coflow sizes, raises a question in Varys’ methodology. According to the study, about 20% of total coflows are indeed less than 1 MB in size, which might benefit more from a per-flow fairness algorithm.

The authors justify their results by emphasizing volume-wise metrics, where large coflows dominate over 99% of all bytes communicated due to the heavy-tailed nature of coflow traffic. Although this argument stands on its own merits, we find it slightly concerning whether it’s a good call to sacrifice a significant number of coflows despite their relatively small traffic volume. It may sound a little bit philosophical when we try to argue about fairness here. Nevertheless we believe it would be a promising future work to design a heterogeneous coflow scheduling system such that it could leverage different scheduling algorithms depending on the coflow dimensions. For example, it could maintain a per-flow fairness for smaller coflows while enforcing Varys’ MADD for the majority large coflows.

4. Optimal Number for T and $\boldsymbol{\delta}$

During EC2 deployment the authors set the algorithm parameters T = 2 seconds and $\boldsymbol{\delta}$ = 200 milliseconds. They explain a little rationale behind these magic numbers in Section 6.2, where they discuss the drawbacks of setting $\boldsymbol{\delta}$ too small. In order to avoid erratic behavior of transport protocol, they “suggest” choosing $\boldsymbol{\delta}$ to be O(100) milliseconds and T to be O(1) seconds.
We believe that it might be worth showing the tuning process of the parameters and justifying the selection of final values on a factual basis. The current process of choosing values for experiment setups seems to be an arbitrary decision to us. As a matter of fact, these parameters actually play an important role in deciding the performance of algorithms. A change in the

parameters will likely affect the performance in all bins in similar or disparate ways. Therefore, we find it will be a promising future work to explore more in tuning these parameters and find out the correlation between different values and Varys' performances. In that case we can have facts about the ideal range of parameter values, and perhaps some optimal values.