Md Majid Jahangir | SBU ID - 112077145
AMS 560 - Big Data Systems, Algorithms and Networks

Assignment 1 (Paper Reading) - Efficient Coflow Scheduling Without Prior Knowledge

Motivation : In a data parallel clusters, application-level communication performance is improved by inter-coflow scheduling as decreasing a coflow's completion time (CCT) leads to faster completion of jobs hence increasing application's performance. Existing efficient schedulers that require a prior coflow information, ignore important cluster dynamics like pipelining, task failures, multi-wave scheduling, etc, because of which such characteristics of coflow cannot be determined until the coflow has completed, thus limiting their applicability to several use-cases. Besides, schedulers without prior knowledge compromise on performance by multiplexing coflows to avoid head-of-line blocking. Thus, this paper presents a coordinated inter-coflow scheduler that does not compromise on performance and minimizes the average CCT without any assumption of prior knowledge about coflow characteristics.

Prior Solutions / Differences from prior work : Prior works includes non-clairvoyant schedulers like Baraat[3] and Orchestra[2]. Baraat is a FIFO based fully decentralized scheduler that considers only local observations which does not give accurate information about CCTs of large coflows, while Orchestra, again a FIFO based scheduler, multiplexes coflows, and both does not assume any prior knowledge about coflow characteristics. The proposed solution Aalo, also a non-clairvoyant scheduler, uses global information unlike Baraat and avoids head-of-line blocking unlike Orchestra to outperform both existing schedulers.
Another clairvoyant scheduler Varys[5] uses heuristics like smallest-bottleneck-first to schedule coflows, they have complete global information over all the coflows in a system and hence performs better for larger coflows. However, Aalo is a non-clairvoyant scheduler that schedule coflows without any prior knowledge making it practical in the presence of task failures, multi-wave scheduling, pipelining, dependencies in DAG, etc, that Varys ignores and hence makes it impractical in many scenarios. Aalo performs better than Varys in tiny coflows by avoiding coordination and complete information, however Varys marginally outperforms Aalo for larger coflows by exploiting complete information. Hedera[4], a traditional flow scheduling is used for point to point flows of data communication than Aalo which generalizes it to capture multipoint to multipoint.

Key Ideas of Proposed Solution / How does the solution work : The proposed solution is Aalo that employs Discretized Coflow-Aware Least-Attained Service (D-CLAS) and supports coflow dependencies and pipelines and works well for cluster dynamics like multi-wave scheduling. It requires no prior knowledge about coflow characteristics like coflow size, number of flows in coflow, etc. However, it tracks the total number of bytes sent by all the flows of a coflow and assigns a priority to each coflow and the priority decreases with the total number of bytes that the coflow has already sent. Thus, average CCT is reduced as smaller coflows have higher priorities. For heavy tailed distribution of coflow sizes, Aalo approximates smallest-total-size-first heuristic, whereas, for light tailed distribution, it discretizes coflow priorities by decreasing priority only when number of bytes sent exceeds some predefined thresholds. D-CLAS is implemented using a multi-level scheduler, where each queue maintains all coflows with same priority. Within the same queue, FIFO is followed whereas, across queues, weighted fair queuing is used to weigh the queue based on its coflow priority thus guaranteeing starvation freedom.

Properties : Since, smallest-bottleneck-first heuristic is known only after the coflow has completed**,** a non-clairvoyant scheduler must schedule a coflow based on an attribute that -

- Can approximate its clairvoyant counterpart using current observations.
- Involves all the flows to avoid drawbacks from the lack of coordination.

Also, apart from minimizing average CCT time, a non-clairvoyant scheduler must -

- Guarantee starvation freedom for bounded CCTs.
- Decrease coordination requirements for scalability.
- Ensure work conservation to increase utilization.

Performance : Aalo uses global information unlike Baraat and avoids head-of-line blocking unlike Orchestra to outperform both existing schedulers. It improved CCTs upto 2.25 times with respect to per flow sharing for communication dominated jobs. Aalo's average improvements were within 12% of Varys for single-stage, single-wave coflows, and it outperformed Varys for multi-stage, multi-wave coflows by up to 3.7 times through dependency-aware scheduling. In trace-driven simulations, Aalo performed 2.7 times better than per-flow fair sharing and up to 16 better than fully decentralized solutions that suffer significantly due to the lack of coordination. Aalo coordinator can also scale to O(10000) daemons with minimal performance loss.

How it handles lack of prior knowledge : Instead of using smallest-bottleneck-first heuristic which is not known until coflow completes, it uses smallest-total-size-first heuristic. This heuristic captures the how much bytes the flow has sent through the cluster and this attribute is monotonically increasing with each flow regardless of start time or endpoints. Thus without knowing about the start time or the endpoints or other characteristics of the coflow, using this heuristic a coflow could be prioritized where priority decreases with the current size of the coflow thus minimizing the average CCTs. Besides, the decrease in priorities could be discretized as explained in above sections to reduce too much coordination.

Difference between coflow scheduling and traditional flow scheduling? : A coflow is a collection of parallel flows with distributed endpoints, and it completes after all its flows have completed. Therefore, a coflow scheduling considers scheduling across a multipoint to multipoint aspect of data communications whereas a traditional flow scheduling considers scheduling across a point to point flow aspect of data communication.

If you are given the information (prior knowledge), can you do better? Yes, if i know certain information like smallest-bottleneck attributes, then scheduling a job with smallest bottleneck first will give a better result.

Strength :

1. Works well in systems involving cluster dynamics like pipelining, task failures, multiwave scheduling which is prevalent in practical scenarios.
2. Gives the best of both the existing schedulers - Baraat and Varys. Approximates both FIFO schedulers for light-tailed distribution of coflow sizes and approximates smallest-total-size-first heuristic for heavy tailed coflow distribution.
3. Uses loose coordination to efficiently schedule tiny coflows and performs comparable to a clairvoyant system like Varys that has complete information about the coflows characteristics.

Weakness :

1. Determining the number of queues and the corresponding threshold to discretize the decrease in priority is an open problem.
2. For in-network bottlenecks, designing, deploying and enforcing a distributed, coflow-aware routing and balancing is an unexplored domain.
3. Having a decentralized Aalo system where decentralizing D-CLAS depends on avoiding receiver side contentious without coordination and this is difficult since it depends on fast propagation of receiver feedbacks.

Follow Up Works :

1. **Siphon: expediting inter-datacenter coflows in wide-area data analytics[6]** - Shuhao Liu, Li Chen, Baochun Li : The paper discusses about novel intra-coflow and inter-coflow scheduling and routing strategies that have been implemented in Siphon. Siphon is a transport service that schedules the inter-datacenter traffic with the awareness of workload-level dependencies and performance.

2. **Trumpet: Timely and Precise Triggers in Data Centers[7]** - Masoud Moshref, Minlan Yu, Ramesh Govindan, Amin Vahdat : The paper discusses about an event monitoring system, Trumpet, that leverages CPU resources and end-host programmability, to

monitor every packet and report events at millisecond scale. Such properties determined at real time could help in coflow scheduling.

3. **Exploiting Inter-Flow Relationship for Coflow Placement in Datacenters[8] -** Xin Sunny Huang, T.S. Eugene Ng : The paper studies the coflow placement problem i.e. the endpoints of subflows within a coflow are preset, with considerations of the inter-flow relationship in coflows. It formulates the coflow placement problem and proposes a coflow placement algorithm.

References -

1. Efficient Coflow Scheduling Without Prior Knowledge - Mosharaf Chowdhury, Ion Stoica
2. M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. Managing data transfers in computer clusters with Orchestra. In SIGCOMM, 2011.
3. M. Chowdhury, Y. Zhong, and I. Stoica. Efficient coflow scheduling with Varys. In SIGCOMM, 2014.
4. M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In NSDI, 2010.
5. M. Chowdhury, Y. Zhong, and I. Stoica. Efficient coflow scheduling with Varys. In SIGCOMM, 2014
6. ACM Digital Library, https://dl.acm.org/citation.cfm?id=3277405
7. ACM Digital Library, https://dl.acm.org/citation.cfm?id=2934879
8. ACM Digital Library, https://dl.acm.org/citation.cfm?id=3107004