DAIET: A System for Data Aggregation Inside the Network

Amedeo Sapio, Ibrahim Abdelaziz, Marco Canini, Panos Kalnis
KAUST

1 Context and Motivation

Many data center applications nowadays rely on distributed computation models like MapReduce and Bulk Synchronous Parallel (BSP) for data-intensive computation at scale [4]. These models scale by leveraging the partition/aggregate pattern where data and computations are distributed across many worker servers, each performing part of the computation. A communication phase is needed each time workers need to synchronize the computation and, at last, to produce the final output. In these applications, the network communication costs can be one of the dominant scalability bottlenecks especially in case of multi-stage or iterative computations [1].

The advent of flexible networking hardware and expressive data plane programming languages have produced networks that are deeply programmable [2]. This creates the opportunity to co-design distributed systems with their network layer, which can offer substantial performance benefits. A possible use of this emerging technology is to execute the logic traditionally associated with the application layer into the network itself. Given that in the above mentioned applications the intermediate results are necessarily exchanged through the network, it is desirable to offload to it part of the aggregation task to reduce the traffic and lessen the work of the servers. However, these programmable networking devices typically have very stringent constraints on the number and type of operations that can be performed at line rate. Moreover, packet processing at high speed requires a very fast memory, such as TCAMs or SRAM, which is expensive and usually available in small capacities.

2 The DAIET Approach

In this work, we propose DAIET, a system for data aggregation in network. DAIET leverages a programmable data plane to reduce the traffic as it is being forwarded towards the destination by opportunistically offloading the aggregation task to the network. In many distributed algorithms, the aggregation function is typically *commutative* and *associative*. Therefore, each network device can independently aggregate part of the data without affecting the correctness of the result. Moreover, the destination workers remain in charge of the portion of the aggregation task that is not handled by the network.

Since these applications typically exchange the intermediate results with *many-to-one* communications, DAIET models this pattern using several in-network aggregation trees, where the root is the destination, the leaves are the data sources and the network devices in the path are the intermediate nodes. A programmable network device aggregates the intermediate results for each tree it belongs to, reducing the traffic at each step in the forwarding path. This solution allows to reduce the network traffic significantly. It also reduces the amount of processing performed by the destination server, since most of the aggregation is performed by network devices at line rate.

We argue that the time has come to entrust network devices with part of the tasks typically executed by software, such as aggregation functions. The benefits provided by the progressive reduction of network traffic have also been proved in previous work that used middleboxes for data aggregation [3]. DAIET provides the same benefits but aims to exploit programmable data plane devices, while also freeing the CPU from part of the aggregation task. This does not affect correctness; in fact, when there is no programmable networking hardware available, the system can fallback to the software implementation.

We present a prototype implementation of DAIET, using P4 [2], for MapReduce-based applications. However, the techniques proposed by DAIET are general enough to be implemented on various programmable network devices, other network programming languages, and be applicable for other applications that follow the partition/aggregate pattern (e.g., graph processing, deep learning, and stream processing). Our results show a promising 88% median data reduction and a similar decrease in reduce computation time.

References

[1] M. Alizadeh et al. Data center tcp (dctcp). In ACM SIGCOMM CCR, 2010.

[2] P. Bosshart et al. P4: Programming protocol-independent packet processors. In ACM SIGCOMM CCR, 2014.

[3] L. Mai et al. NetAgg: Using Middleboxes for Application-specific On-path Aggregation in Data Centres. In CoNEXT, 2014.

[4] A. Roy et al. Inside the Social Network's (Datacenter) Network. In SIGCOMM, 2015.