Leveraging Network Programmability and In-network Computing for Streaming Analysis of Huge Data

Joaquin Chung[*], Zhengchun Liu, Tekin Bicer, Raj Kettimuthu, Ian Foster

Argonne National Laboratory, USA

Abstract — Scientists often wait long queue times to perform experiments, observations, and simulations on advanced scientific instruments. To utilize them efficiently, researchers often need to analyze huge data streaming from these instruments in near-real time, so that results from one experiment/simulation can guide selection of the next—or even influence the course of a single experiment/simulation. However, streaming analysis of huge data still faces challenges regarding network capacity between sites and redistribution of streaming data to multiple consumers. In this white paper we explore how the emerging trends of network programmability and in-network computing could aid the realization of streaming analysis of huge data for scientific applications. For instance, offloading computations to the network may reduce the volume of streaming data, while using network programmability may help to efficiently stream huge data to multiple consumers.

1 Streaming Analysis of Huge Data from Light Source Instruments

Light sources are crucial tools for addressing grand challenge problems in the life sciences, energy, climate change, and information technology [1]. For instance, the X-rays produced at a light source enable scientists to study internal morphology of materials and samples with very high spatial (atomic and molecular scale) and temporal (<100 ps) resolutions. These experiments can generate massive amounts of burst data. For example, tomographic imaging stations can collect 1,500 projections (images each with 2,048 x 2,048 pixels) in 9 seconds, generating data at a rate of >8 Gbps. Real-time streaming and analysis of these experimental data enable scientists (or the control software) to (1) make timely decisions that can significantly accelerate the execution of experiments, and (2) do smart experimentation, such as changing the parameters interactively or finalizing experimentation with only sufficient data.

We have developed an autonomous stream-processing system that allows data streamed from a light source instrument to be processed in real time on a remote compute system, with a control feedback loop used to make decisions during experimentation. This system can be used to analyze collected data and measure the output quality with low latency. The measured output quality is then used to make decisions for terminating the experiment, not only saving time but also saving sensitive samples (e.g., biological samples) from radiation damage. For example, for a real-world sample, this system showed that collecting 100 rather than 180 projections is sufficient to reconstruct relevant sample features (Figure 1). Collecting only the required amount of data resulted in a speedup of 1.8x for both data acquisition and analysis.

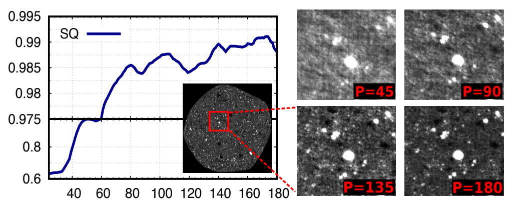

Figure 1: Real-time analysis and quality tests for data collected in a light source experiment. The x-axis is the number of projections processed, and the y-axis is the quality score.

Could we achieve higher speedups by offloading some projection normalization operations to the network? Normalization of projections requires simple arithmetic operations that can be easily executed by networking devices. Any switch in the network could execute the required operations, as we split operations along the path between source and destination. The advantage of pipelining data transfer and computation may save time compared to doing it after moving all the data. Furthermore, by executing these computations in the network, we can free cycles from supercomputers that could be used for more complex tasks.

Could we achieve more efficient streaming data distribution by leveraging network programmability? Some huge data streaming workflows may involve more that one consumer, and they may probably consume data for different purposes. For instance, one consumer could be a supercomputer performing streaming data analysis, while another consumer could be a visualization process. Instead of sending separate unicast flows,

[*]chungmiranda@anl.gov

we could leverage network programmability to apply multicast techniques to save bandwidth on the network. Furthermore, different consumers may use different transport protocols, i.e., protocol A for some subset of the flows and protocol B for the rest.

2 Network Programmability and In-network Computing

Network programmability is reaching higher levels with the separation between switch hardware and control software, and the availability of programmable network switches in which the data plane functionality can be programmed rather than being controlled by vendors. Software-defined networking (SDN) [2]—a networking paradigm in which the control and data planes of network devices are decoupled—enables global, agile network programmability, rapid innovation, and independent evolution of control and data planes. On the other hand, programmable data planes [3] are reconfigurable networking devices that allow programmers to define exactly how packets are processed.

In-network computing is the process of offloading operations from end hosts into networking devices (e.g., switches, routers, or smart NICs) [4]. Two emerging technologies have made in-network computing possible: programmable data planes and networked compute accelerators (e.g., GPUs and FPGAs) [5]. As networking devices are limited in memory (use of expensive TCAMs), set of actions (only arithmetic/boolean operations), and operations per packet (no loops), Sapio et at. [4] study what type of computations should be performed in-network. The authors identified applications that follow a partition/aggregate pattern as a plausible class of applications that could benefit from in-network computing. Examples of these applications are big data analytic and machine learning, graph processing, and streaming analysis.

3 Challenges

Breaking end-to-end TCP connections: By manipulating headers or changing the payload with in-network computing, we will change the checksum of TCP packets. Furthermore, IP packets that belong to the same TCP flow are not expected to take the same path. We need to obtain more understanding on the communication patterns of flows subject to in-network computing. For instance, would it be required to explore ways to disable TCP checksums on those flows, or would it be sufficient to use UDP transport? We suggest that circuit-switched approaches such as MPLS and Segment Routing combined with SDN may help keep packets from the same flow on the same path.

Data Segmentation/Partitioning: In the case of light source data streaming, the size of a typical projection is 1K by 1K or 1 MB, which is larger that a jumbo frame (9 KB). Fortunately, we do not need each projection to fit on a single packet. If the algorithm allows element-wise operation, i.e., operation to each independent pixel, we can certainly pack several pixels in one packet. However, applications that work with bigger chunks of data may require to develop novel approaches to data segmentation/partitioning.

Added Complexity: Using SDN combined with MPLS or Segment Routing to control and program networks increases complexity. Furthermore, for streaming applications to take advantage of network programmability and in-network computing, changes to the application will be needed. For instance, APIs for interacting with the network through SDN or knowing when data have been changed by lightweight operations on the network. We need to understand the tradeoffs of adding in-network computing into huge data streaming workflows by evaluating changes on network and application architectures.

References

[1] "The Report of the BES Advisory Subcommittee on Future X-ray Light Sources." `https://science.osti.gov/-/media/bes/besac/pdf/Reports/Future_Light_Sources_report_BESAC_approved_72513.pdf`, 2013. Accessed: 2020-01-06.

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, pp. 14–76, Jan 2015.

[3] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 99–110, ACM, 2013.

[4] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis, "In-network computation is a dumb idea whose time has come," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, (New York, NY, USA), p. 150–156, Association for Computing Machinery, 2017.

[5] A. Putnam, A. Caulfield, E. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, E. Peterson, A. Smith, J. Thong, P. Y. Xiao, D. Burger, J. Larus, G. P. Gopal, and S. Pope, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture (ISCA)*, pp. 13–24, IEEE Press, June 2014. Selected as an IEEE Micro TopPick.