JumpGate: Towards In-Network Data Processing

Craig Mustard (Student Author)
craigm@ece.ubc.ca
University of British Columbia

Alexandra Fedorova
sasha@ece.ubc.ca
University of British Columbia

Ivan Beschastnikh
bestchai@cs.ubc.ca
University of British Columbia

Today, it is common to store data in long-term storage systems and to load data into the memory of systems for further processing. Therefore, the network path from storage to main-memory is a natural place to perform early processing on data being loaded. Programmable packet processors are emerging that can efficiently process a large number of packets using specialized hardware designs [8, 9, 13, 20, 26] or optimized software on general purpose processors [22, 24]. We are studying the feasibility and potential performance gains of JumpGate, a system that uses packet processors to perform common database operations (filter, projection, shuffle, aggregation) on data as it moves from storage to compute nodes.

In-network compared to near-storage processing Near-storage [10, 14, 16, 17, 18] and in-memory processors [3, 5, 27] have high throughput designs that efficiently process large amounts of data. However, coupling compute to storage can leave powerful processors and expensive storage mediums under-utilized when data is not accessed often. Packet processing systems can avoid this utilization problem because they are not tied to storage. Since packet processors can be quickly allocated and released, they may be complimentary to existing near-storage compute systems that use general purpose processors, such as [1, 25].

JumpGate Design. Figure 1 shows the JumpGate architecture and illustrates how a query would be executed. A query processor (e.g., Apache Spark [4]) would compile applicable parts of a query to a high-level packet-processing program and send the program to a controller that allocates available packet processors and compiles the program for them (steps 1-4). The storage cluster (e.g., HDFS) would send (properly formatted) data towards the packet processors (step 5-6), which process data as it arrives, forwarding the processed packets to the query processor's nodes to finish the remainder of the query (step 7). JumpGate could target a variety of packet processors, such as programmable switches [8], NICs [13] or general purpose processors by compiling queries to P4 [7], eBPF [6] or C using the DPDK [24].

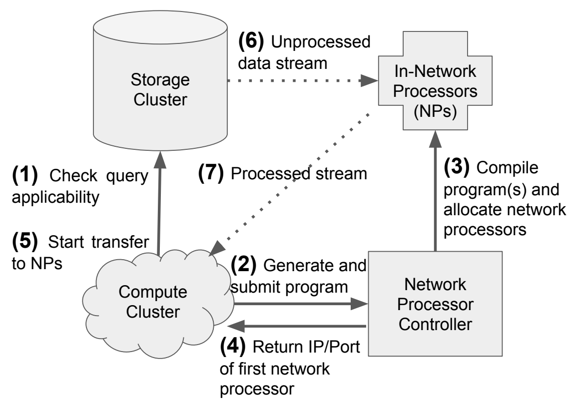

Figure 1: JumpGate architecture.

Potential Contributions We are encouraged by recent successes on compiling Spark queries to native code [11, 12], compiling P4 to eBPF/XDP [2], processing network telemetry operators in packet processors [15, 19], and performing raw filtering before parsing [21]. Potential contributions would be the compilation techniques, fault tolerant network protocols and operator implementations neccessary to achieve beneficial in-network processing on data stored in typical storage systems and formats.

A full JumpGate implementation would require extensive development effort and changes to multiple systems. To motivate such efforts, we are first studying feasibility and potential performance improvements of JumpGate by evaluating transport protocols and operators on packet traces modeled on real schemas. For example, we are investigating: (1) performing early filtering on JSON records where each packet holds at least one complete record, and (2) performing in-network shuffles by directing/cloning packets to their assigned host(s). The results of this study will be transport protocols and operator implementations we can use for evaluations on relevant benchmark suites, such as TPC-DS [23].

We hope to present more details on JumpGate and early results of our study in the poster session at OSDI.

References

[1] Amazon S3 Select. https://aws.amazon.com/s3/features/#s3-select.

[2] p4c-xdp. https://github.com/vmware/p4c-xdp.

[3] S. R. Agrawal, S. Idicula, A. Raghavan, E. Vlachos, V. Govindaraju, V. Varadarajan, C. Balkesen, G. Giannikis, C. Roth, N. Agarwal, and E. Sedlar. A many-core architecture for in-memory data processing. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pages 245–258, New York, NY, USA, 2017. ACM.

[4] Apache Software Foundation. Apache Spark. http://spark.apache.org/. [Online; accessed 10-April-2017].

[5] C. Balkesen, N. Kunal, G. Giannikis, P. Fender, S. Sundara, F. Schmidt, J. Wen, S. Agrawal, A. Raghavan, V. Varadarajan, A. Viswanathan, B. Chandrasekaran, S. Idicula, N. Agarwal, and E. Sedlar. Rapid: In-memory analytical query processing engine with extreme performance per watt. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 1407–1419, New York, NY, USA, 2018. ACM.

[6] A. Begel, S. McCanne, and S. L. Graham. Bpf+: Exploiting global data-flow optimization in a generalized packet filter architecture. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '99, pages 123–134, New York, NY, USA, 1999. ACM.

[7] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.

[8] 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 *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 99–110, New York, NY, USA, 2013. ACM.

[9] Cavium. Liquidio ii network appliance smart nics. https://www.cavium.com/liquidio-II-network-appliance-adapters.html.

[10] J. Do, Y.-S. Kee, J. M. Patel, C. Park, K. Park, and D. J. DeWitt. Query processing on smart ssds: Opportunities and challenges. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1221–1230, New York, NY, USA, 2013. ACM.

[11] G. Essertel, R. Tahboub, J. Decker, K. Brown, K. Olukotun, and T. Rompf. Flare: Optimizing apache spark for scale-up architectures and medium-size data. In *Operating Systems Design and Implementation*. USENIX, 2018.

[12] G. M. Essertel, R. Y. Tahboub, J. M. Decker, K. J. Brown, K. Olukotun, and T. Rompf. Flare: Native compilation for heterogeneous workloads in apache spark. *CoRR*, abs/1703.08219, 2017.

[13] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H. Angepat, V. Bhanu, A. Caulfield, E. Chung, H. K. Chandrappa, S. Chaturmohta, M. Humphrey, J. Lavier, N. Lam, F. Liu, K. Ovtcharov, J. Padhye, G. Popuri, S. Raindel, T. Sapre, M. Shaw, G. Silva, M. Sivakumar, N. Srivastava, A. Verma, Q. Zuhair, D. Bansal, D. Burger, K. Vaid, D. A. Maltz, and A. Greenberg. Azure accelerated networking: Smartnics in the public cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 51–66, Renton, WA, 2018. USENIX Association.

[14] B. Gu, A. S. Yoon, D.-H. Bae, I. Jo, J. Lee, J. Yoon, J.-U. Kang, M. Kwon, C. Yoon, S. Cho, J. Jeong, and D. Chang. Biscuit: A framework for near-data processing of big data workloads. In *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16, pages 153–165, Piscataway, NJ, USA, 2016. IEEE Press.

[15] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger. Sonata: Query-driven streaming network telemetry. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 357–371, New York, NY, USA, 2018. ACM.

[16] Z. István, D. Sidler, and G. Alonso. Caribou: Intelligent distributed storage. *Proc. VLDB Endow.*, 10(11):1202–1213, Aug. 2017.

[17] I. Jo, D.-H. Bae, A. S. Yoon, J.-U. Kang, S. Cho, D. D. G. Lee, and J. Jeong. Yoursql: A high-performance database system leveraging in-storage computing. *Proc. VLDB Endow.*, 9(12):924–935, Aug. 2016.

[18] G. Koo, K. K. Matam, T. I, H. V. K. G. Narra, J. Li, H.-W. Tseng, S. Swanson, and M. Annavaram. Summarizer: Trading communication with computing near storage. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pages 219–231, New York, NY, USA, 2017. ACM.

[19] S. Narayana, A. Sivaraman, V. Nathan, P. Goyal, V. Arun, M. Alizadeh, V. Jeyakumar, and C. Kim. Language-directed hardware design for network performance monitoring. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 85–98, New York, NY, USA, 2017. ACM.

[20] Netronome. Smartnics overview. https://www.netronome.com/products/smartnic/overview/.

[21] S. Palkar, F. Abuzaid, P. Bailis, and M. Zaharia. Filter before you parse: Faster analytics on raw data with sparser. *Proceedings of the VLDB Endowment*, 11(11), 2018.

[22] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker. E2: A framework for nfv applications. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 121–136, New York, NY, USA, 2015. ACM.

[23] M. Poess, B. Smith, L. Kollar, and P. Larson. Tpc-ds, taking decision support benchmarking to the next level. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD '02, pages 582–587, New York, NY, USA, 2002. ACM.

[24] D. Project. Data plane development kit (dpdk). https://www.dpdk.org/.

[25] R. Weiss. *A Technical Overview of the Oracle Exadata Database Machine and Exadata Storage Server*. Oracle Corporation, 2012.

[26] Wikipedia. Tilera. https://en.wikipedia.org/wiki/Tilera.

[27] L. Wu, A. Lottarini, T. K. Paine, M. A. Kim, and K. A. Ross. Q100: The architecture and design of a database processing unit. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, pages 255–268, New York, NY, USA, 2014. ACM.