DPI: The Data Processing Interface for Modern Networks

Carsten Binnig[1]

Extended Abstract

The computer networks available in data centers and clusters are evolving rapidly, increasingly providing sophisticated capabilities such as RDMA (Remote Direct Memory Access), in-network processing, and customizable communication protocols. Once the province of specialized, expensive networks, the new functionality is becoming available in off-the-shelf networks as well. An example of how these advances can help with data intensive applications is RDMA, the ability to directly read or write the memory of remote machines without involving the remote CPU. RDMA makes data transfer more efficient, and it frees up computing capacity, which can lead to substantial performance gains [Ka16, Dr15, Dr14, Lo15, Za17, Ou11, Mi13, Ka14, Yo18, De05, Co17]. Unfortunately, using RDMA is complicated because it lacks higher-level abstractions [Dr17]. Recent work on using RDMA in relational databases has shown that the design involves many low-level, yet significant, decisions around connection management, memory allocation, and the choice of which RDMA operations to use [Bi16, Ba15].

This fragile dependency on low-level design aspects and lack of portability across networks is not unique to RDMA; it affects other technologies like smart NICs (Network Interface Cards) and programmable switches as well [Fi18]. This is concerning because modern networks are increasingly software-defined, and there is a growing need to tailor them to data processing, e.g., through load balancing and skew detection at the switch level, data partitioning on the NIC, and content based routing. Although recent results [Bl18, Sa17] have shown that smart NICs and programmable switches can improve the performance of distributed data processing systems, the hand-tuning of low level details remains a problem. Not only is the programming of the devices complex, it also creates resource management problems such as deciding when to offload computation into the network.

In this talk, I present the Data Processing Interface (DPI) as a way to address these problems. DPI's goal is to make it easier for applications to exploit these emerging capabilities of modern networks. Accordingly, DPI defines abstractions and interfaces suited to a broad class of data-intensive applications, yet simple enough for practical implementation with predictable performance and low overhead relative to "hand-tuned", ad hoc alternatives. In designing an interface tailored to data processing, we adopt the approach taken by other high-level interfaces, such as MPI (Message Passing Interface) [Gr14], which have been designed for other application domains and which, consequently, have seen only limited adoption for data processing [Ba17]. A detailed paper about DPI has recently been presented at the CIDR'19 conference [Al19].

[1] TU Darmstadt, Data Management Lab - Informatik, Germany, carsten.binnig@cs.tu-darmstadt.de

doi:10.18420/btw2019-ws-02

References

[Al19] Alonso, Gustavo et al.: DPI: The Data Processing Interface for Modern Networks. In: CIDR 2019. 2019.

[Ba15] Barthels, Claude et al.: Rack-Scale In-Memory Join Processing using RDMA. In: ACM SIGMOD. pp. 1463–1475, 2015.

[Ba17] Barthels, Claude et al.: Distributed Join Algorithms on Thousands of Cores. PVLDB, 10(5):517–528, 2017.

[Bi16] Binnig, Carsten et al.: The end of slow networks: It's time for a redesign. PVLDB, 9(7):528–539, 2016.

[Bl18] Blöcher, Marcel et al.: Boosting scalable data analytics with modern programmable networks. In: ACM DaMoN@SIGMOD. ACM, pp. 1:1–1:3, 2018.

[Co17] Chen, Haibo; other: Fast In-Memory Transaction Processing Using RDMA and HTM. ACM Trans. Comput. Syst., 35(1):3:1–3:37, 2017.

[De05] Devulapalli, Ananth et al.: Distributed Queue-Based Locking Using Advanced Network Features. In: ICPP. pp. 408–415, 2005.

[Dr14] Dragojević, Aleksandar et al.: FaRM: Fast remote memory. In: NSDI. pp. 401–414, 2014.

[Dr15] Dragojević, Aleksandar et al.: No compromises: distributed transactions with consistency, availability, and performance. In: OSDI. pp. 54–70, 2015.

[Dr17] Dragojevic, Aleksandar et al.: RDMA Reads: To Use or Not to Use? IEEE Data Eng. Bull., 40(1):3–14, 2017.

[Fi18] Firestone, Daniel et al.: Azure Accelerated Networking: SmartNICs in the Public Cloud. In: NSDI. pp. 51–66, 2018.

[Gr14] Gropp, William et al.: Using Advanced MPI: Modern Features of the Message-Passing Interface. The MIT Press, 2014.

[Ka14] Kalia, Anuj et al.: Using RDMA efficiently for key-value services. In: Proc. of ACM SIGCOMM. pp. 295–306, 2014.

[Ka16] Kalia, Anuj et al.: FaSST: fast, scalable and simple distributed transactions with two-sided (RDMA) datagram RPCs. In: Proc. of OSDI. pp. 185–201, 2016.

[Lo15] Loesing, Simon et al.: On the Design and Scalability of Distributed Shared-Data Databases. In: ACM SIGMOD. pp. 663–676, 2015.

[Mi13] Mitchell, Christopher et al.: Using One-Sided RDMA Reads to Build a Fast, CPU-Efficient Key-Value Store. In: Proc. of USENIX ATC. pp. 103–114, 2013.

[Ou11] Ousterhout, John et al.: The case for RAMCloud. Communications of the ACM, 54(7):121–130, 2011.

[Sa17] Sapio, Amedeo et al.: DAIET: a system for data aggregation inside the network. In: SoCC. ACM, p. 626, 2017.

[Yo18] Yoon, Dong Young et al.: Distributed Lock Management with RDMA: Decentralization without Starvation. In: ACM SIGMOD. pp. 1571–1586, 2018.

[Za17] Zamanian, Erfan et al.: The End of a Myth: Distributed Transaction Can Scale. PVLDB, 10(6):685–696, 2017.