Dagger: Efficient and Fast RPCs for Cloud Microservices with Near-Memory Reconfigurable NICs
Extended Abstract

Nikita Lazarev Shaojie Xiang Neil Adit Zhiru Zhang Christina Delimitrou

Cornell University

1. Motivation

The shift of cloud services from monolithic applications to microservices and the increasing interest in interactive applications with strict latency constraints, are changing the requirements of datacenter networks. Typical microservices are not computationally intense and very short; in the order of a few hundred millisecond to a few seconds [12, 24], therefore a substantial fraction of their latency comes from networking. Remote Procedure Calls (RPC) are one of the most common communication primitives in microservices. Unfortunately, commercially-available RPC frameworks, such as Apache Thrift [3], gRPC [2], Finagle [1] are all based on commodity networking systems, like the Linux kernel stack, which introduce considerably overheads when it comes to microservices [14, 22, 24]. Reducing these overheads and improving the overall performance of cloud networking is one of the key challenges towards enabling more widespread adoption of interactive microservices.

2. Limitations of the State of the Art

We can classify current solutions on low-latency datacenter networks into three groups. The first class is based on algorithmic optimizations of networking stacks. This includes proposals for new transport layers [5, 6, 13, 19], with optimized congestion control, flow scheduling, connection management, etc. Some proposals move network processing from kernel to user space [8, 17] therefore eliminating the inefficiency of crossing the kernel-user boundary. Even though these solutions improve the overall performance of networking systems, they are subject to two issues: (1) software-based implementations of network protocols add CPU-related overheads, and consume processor resources, which is especially problematic, given the highly-concurrent nature of microservices, and (2) almost all commercially-available NICs are peripheral devices seen by processors over PCIe interfaces, which has been shown to be inefficient for fine-grained workloads [11, 15, 20].

To address the first issue of software-based networking stacks, another line of research [9, 16, 25] proposes to leverage specialized adapters such as RDMA NICs to run the networking stack in hardware, and make use of the remote memory abstraction to implement efficient high-level communication primitives, such as RPCs, on top of it. However, these proposals (1) still inherit the PCIe-related overheads of software-based solutions [15], since all commercial RDMA adapters are based on PCIe, and (2) they do not offload the whole communication stack to hardware, keeping the execution of the RPC layer on the host CPU.

Both these limitations are receiving increased attention today. To this end, the recent proposals have presented solutions for integrating NICs into the host processor (soNUMA [21]) or memory (NetDIMM [4]), and discussed the implementation of hardware-offloaded RPC stacks on top of them [22, 26]. Unfortunately, these proposals require taping out custom hardware, which requires considerably investments at cloud scale, and is not well-suited for frequently-changing applications. For instance, NetDIMM requires designing custom memory chips that integrate networking interfaces, while NeBuLa [26] proposes to push packets all the way to the L1 caches, which requires redesigning the processor memory subsystem. Despite these issues, the idea of closely-coupling processors with NICs is both attractive and promising. In this work, we discuss how closely-coupled NICs can be made practical by leveraging FPGAs attached to the CPUs over coherent memory interconnects (i.e., NUMA) as networking devices.

3. Key Insights

Dagger provides the following three key insights regarding efficient datacenter networks for interactive microservices.

1. Fully offloading the RPC stack in hardware: We profile commodity networking stacks for microservices, and find that both the RPC layer and the networking layer, i.e., transport, NICs, wiring, account for a sizeable portion of application latency. This shows that offloading/accelerating the networking layer alone is not sufficient. For this reason, Dagger offloads the entire networking stack up to, and including, the RPC layer to hardware, leaving only a small portion of computation on the processor, corresponding to the RPC API and connection set-up routines.

2. Leveraging memory interconnects as CPU-NIC interfaces: PCIe has been the de facto interface between the CPU and NIC for a long time. However, the PCIe messaging model is not efficient for transferring small, fine-grained requests, which dominate network traffic in microservices. In contrast, memory interconnects provide a different messaging scheme,

which does not require explicit notifications to be sent from the processor to notify NICs about new requests: the data transfer is instead handled by the interconnect's state machine and is implemented entirely in hardware. This improves the CPU efficiency of transferring small requests, resulting in higher throughput and lower latency. The current version of Dagger is based on the Intel UPI NUMA interconnect, available in commercial server Intel Xeon processors.

3. FPGA implementation of reconfigurable NICs: All commercially successful RPC frameworks are modular and reconfigurable. For example, Thrift provides a flexible choice of serialization/de-serialization schemes and transport layers. This is very important for microservices, given the diversity in their characteristics, and the frequent cadence of their updates. We design Dagger following a similar principle: our NIC is based on a fully programmable FPGA. This allows the users or cloud providers to configure the hardware RPC pipeline based on the demands and traffic characteristics of their applications. In addition to our NUMA-based interface, Dagger also implements the commodity doorbell method, which is more efficient when networking objects are large. We characterize the networking footprint of microservices and observe that RPC sizes differ across microservices, even within the same application: some tiers never send RPCs larger than 64 Bytes while others only deal with large RPCs of few KBytes. Depending on the network footprint of a target microservice, Dagger's CPU-NIC interface can be differently configured.

4. Contributions & Main Artifacts

In this work, we show that memory interconnects provide a better alternative to CPU-NIC interfaces than PCIe busses and other forms of integrating NICs to CPUs [4, 21]. We show that this approach enables efficient implementation of hardware-offloaded RPC stacks, providing dramatic performance improvements to interactive microservices. Dagger is implemented in real FPGA hardware, therefore it captures all factors contributing to application performance, and supports a variety of interactive cloud services, from small microservices to key-value stores, like memcached.

We present the following three artifacts:

- **Characterization** of the networking footprint of representative interactive microservices, which highlights the dominance of fine-grained, small requests, and guides the design decisions in Dagger. Our study also demonstrates the diverse requirements of different microservices, underpining the need for the networking stack to be programmable.
- **Design** of Dagger on a real commercially-available platform based on a server Intel Xeon E5 processor, closely coupled with an FPGA over a UPI memory interconnect. Dagger is the first attempt to leverage *closely-coupled* FPGAs as *programmable networking devices*. In all previous proposals on leveraging FPGAs in datacenters [7, 10, 23], the FPGAs are viewed by the host CPU as peripheral devices connected over PCIe, which comes with substantial performance overheads. We offload the entire communication stack to the FPGA, including the RPC layer, transport, and data link layers. We make the design modular and programmable, so users can select the most suitable transport layers and CPU-NIC interfaces for their target microservices.
- **End-to-End Evaluation** of Dagger using stress testing with RPC requests, as well as widely-used datacenter applications, such as memcached, showing the system's performance benefits and practicality, without the requirement for taping out custom chips.

5. Key Results

We evaluate Dagger using both microbenchmarks and real cloud applications. First, we compare different CPU-NIC interfaces, and show the benefits of using memory interconnects compared to previous PCIe-based solutions. We show that data transfer over memory interconnects achieves 45% and 39% better median and tail end-to-end latency of small (up to 64 Bytes) RPCs compared to the standard doorbell [15] method based on a PCIe bus. We then show Dagger's scalability with the number of threads and CPU cores. Finally, we compare the performance of transferring end-to-end RPCs to previous work. We show that Dagger achieves $2.4-3.8\times$ higher per-core RPC throughput compared to software-optimized solutions leveraging user space networking and RDMA, while also achieving state-of-the-art request latency of 1 - 2 μs. The throughput of Dagger scales up to 84 Mrps with 8 threads on 4 CPU cores, which is 23% higher than the best reported throughput of the RDMA solution FaSST [16].

Our experiments with memcached show that Dagger can be easily integrated with third party datacenter applications, with minimal changes in their codebase. Running on top of Dagger, memcached shows $3.2-8$ us and $5.8-18$ us median and tail latency, which is $2.3\times$ lower in comparison with the highly-optimized key-value store MICA [18], when running over optimized DPDK-based user space networking.

6. Why ASPLOS?

Dagger is a software-hardware co-designed system that leverages low-level processor interconnects for efficient support of high-level cloud RPCs. Dagger is interdisciplinary, being in the intersection of cloud computing, reconfigurable hardware, computer architecture, and networking. Similar papers on network optimization for cloud systems have appeared in prior ASPLOS iterations many times.

7. Citation for Most Influential Paper Award

For proposing the integration of FPGA-based accelerators in cloud systems as first-class citizens using memory interconnects, rather than PCIe-attached peripheral devices, and for showing that offloading the entire RPC stack on hardware using closely-coupled NICs significantly improves the performance of interactive cloud services.

References

[1] Finagle rpc. accessed August, 2020. https://twitter.github.io/finagle/.

[2] gRPC. accessed May, 2020. https://grpc.io/.

[3] Thrift RPC. accessed May, 2020. https://thrift.apache.org/.

[4] Mohammad Alian and Nam Sung Kim. NetDIMM: Low-latency near-memory network interface architecture. *Int'l Symp. on Microarchitecture (MICRO)*, 2019.

[5] Mohammad Alizadeh, Adel Javanmard, and Balaji Prabhakar. Analysis of DCTCP: Stability, convergence, and fairness. *Int'l Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2011.

[6] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. Pfabric: Minimal near-optimal datacenter transport. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, page 435–446, New York, NY, USA, 2013. Association for Computing Machinery.

[7] Mina Tahmasbi Arashloo, Alexey Lavrov, Manya Ghobadi, Jennifer Rexford, David Walker, and David Wentzlaff. Enabling programmable transport protocols in high-speed NICs. *USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, 2020.

[8] Adam Belay, George Prekas, Ana Klimovic, Samuel Grossman, Christos Kozyrakis, and Edouard Bugnion. IX: A protected dataplane operating system for high throughput and low latency. *USENIX Symp. on Operating Systems Design and Implementation (OSDI)*.

[9] Aleksandar Dragojević, Dushyanth Narayanan, Miguel Castro, and Orion Hodson. FaRM: Fast remote memory. *USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, 2014.

[10] Daniel Firestone, Andrew Putnam, Sambhrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian Caulfield, Eric Chung, Harish Kumar Chandrappa, Somesh Chaturmohta, Matt Humphrey, Jack Lavier, Norman Lam, Fengfen Liu, Kalin Ovtcharov, Jitu Padhye, Gautham Popuri, Shachar Raindel, Tejas Sapre, Mark Shaw, Gabriel Silva, Madhan Sivakumar, Nisheeth Srivastava, Anshuman Verma, Qasim Zuhair, Deepak Bansal, Doug Burger, Kushagra Vaid, David A. Maltz, and Albert Greenberg. Azure accelerated networking: Smartnics in the public cloud. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, NSDI'18, page 51–64, USA, 2018. USENIX Association.

[11] Mario Flajslik and Mendel Rosenblum. Network interface design for low latency request-response protocols. *USENIX Annual Technical Conf. (ATC)*, 2013.

[12] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems. *International Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019.

[13] EunYoung Jeong, Shinae Wood, Muhammad Jamshed, Haewon Jeong, Sunghwan Ihm, Dongsu Han, and KyoungSoo Park. mTCP: a highly scalable user-level TCP stack for multicore systems. *USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, 2014.

[14] Anuj Kalia, Michael Kaminsky, and David Andersen. Datacenter RPCs can be general and fast. *USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, 2019.

[15] Anuj Kalia, Michael Kaminsky, and David G. Andersen. Design guidelines for high performance RDMA systems. *USENIX Annual Technical Conf. (ATC)*, 2016.

[16] Anuj Kalia, Michael Kaminsky, and David G. Andersen. FaSST: Fast, scalable and simple distributed transactions with two-sided (RDMA) datagram RPCs. *USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, 2016.

[17] Antoine Kaufmann, Tim Stamler, Simon Peter, Naveen Kr. Sharma, Arvind Krishnamurthy, and Thomas Anderson. Tas: Tcp acceleration as an os service. In *Proceedings of the Fourteenth EuroSys Conference 2019*, EuroSys '19, New York, NY, USA, 2019. Association for Computing Machinery.

[18] Hyeontaek Lim, Dongsu Han, David G. Andersen, and Michael Kaminsky. MICA: A holistic approach to fast in-memory key-value storage. *Symposium on Networked Systems Design and Implementation (NSDI)*, 2014.

[19] Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John Ousterhout. Homa: A receiver-driven low-latency transport protocol using network priorities. *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2018.

[20] Rolf Neugebauer, Gianni Antichi, José Fernando Zazo, Yury Audzevich, Sergio López-Buedo, and Andrew W. Moore. Understanding PCIe performance for end host networking. *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2018.

[21] Stanko Novakovic, Alexandros Daglis, Edouard Bugnion, Babak Falsafi, and Boris Grot. Scale-out NUMA. *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014.

[22] Arash Pourhabibi, Siddharth Gupta, Hussein Kassir, Mark Sutherland, Zilu Tian, Mario Paulo Drumond, Babak Falsafi, and Christoph Koch. Optimus prime: Accelerating data transformation in servers. *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[23] D. Sidler, Z. István, and G. Alonso. Low-latency TCP/IP stack for data center applications. *Int'l Conf. on Field Programmable Logic and Applications (FPL)*, 2016.

[24] Akshitha Sriraman and Abhishek Dhanotia. Accelerometer: Understanding acceleration opportunities for data center overheads at hyperscale. *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[25] Patrick Stuedi, Animesh Trivedi, Bernard Metzler, and Jonas Pfefferle. Darpc: Data center rpc. In *Proceedings of the ACM Symposium on Cloud Computing*, SOCC '14, page 1–13, New York, NY, USA, 2014. Association for Computing Machinery.

[26] Mark Sutherland, Siddharth Gupta, Babak Falsafi, Virendra Marathe, Dionisios Pnevmatikatos, and Alexandros Daglis. The NeBuLa RPC-optimized architecture. *Int'l Symp. on Computer Architecture (ISCA)*, 2020.