COIN
Internet-Draft
Intended status: Informational
Expires: April 14, 2019

J. He
R. Chen
Huawei
M. Montpetit, Ed.
Triangle Video

October 11, 2018

In-Network Data-Center Computing
draft-he-coin-datacenter-00

Abstract

This draft wants to review the existing research and the open issues that relate to the addition of data plane programmability in Data Center. While some of the research hypotheses that are at the center of in-network-computing have been investigated since the time of active networking, recent developments in software defined networking, virtualization programmable switches and new network programming languages like P4 have generated a new enthusiasm in the research community and a flourish of new projects in systems and applications alike. This is what this draft is addressing.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 14, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

It is now a given in the computing and networking world that traditional approaches to cloud and client-server architectures lead to complexity and scalability issues. New solutions are necessary to address the growth of next generation network operation (in data centers and edge devices alike) including automation, self-management, orchestration across components and federation across network nodes to enable emerging services and applications.

Mobility, social network and big data and AI/ML as well as emerging content application in the XR (virtual, augmented and mixed reality) require more scalable, available and reliable solution not only in real time, anywhere and over a wide variety of end devices. While these solutions involve edge resources for computing, rendering and distribution, this paper focuses on the data center what are the current research approaches to create more flexible solutions. We must define what we understand by data centers. In this draft, we are not going to limit them to single location cloud resources but add multiple locations as well as interwork with edge resources to enable the network programmability that is central to next generation DCs in term of supported services and dynamic resilience. This leads to innovative research opportunities, including but not limited to:

- Software defined networking (SDN) in distributed environments.

- Security and trust models.

- Data plane programmability for consensus and key-value operations.

- High Level abstractions as in network computing should focus on primitives, which can be widely re-used in a class of applications and workloads, and identify those high level abstractions to promote deployment.

- Machine Learning (ML) and Artificial Intelligence (AI) to detect faults and failures and allow rapid responses as well as implement network control and analytics.

- New services for mixed reality (XR) deployment with in-network optimization and advanced data structures and rendering for interactivity, security and resiliency.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. In Network Computing and Data Centers

As DC hardware components becoming interchangeable, the advent of software-defined technologies suggests that a change is underway. In the next-generation data center, an increasing percentage of critical business and management functions will be activated in the software layer rather than the underlying hardware. This will allow organizations to move away from the current manual configurations to handle more dynamic, rules-based configurations. Hence, virtualization and cloud computing have redefined the datacenter (DC) boundaries beyond the traditional hardware-centric view [SAPIO]. Servers, storage, monitoring and connectivity are becoming one. The network is more and more the computer.

Hence, there is now a number of distributed networking and computing systems which are the basis of big-data and AI-related applications in DCs in particular. They include Distributed file system (e.g. the Hadoop Distributed File System or HDFS [HADOOP]), distributed memory database (e.g. MemCached [MEM]), distributed computing system (e.g. mapReduce from Hadoop, [HADOOP], Tensorflow [TENSOR], and Spark GraphX [SPARK]), as well as distributed trust systems on the blockchain, such as hyperledgers and smart contracts.

In parallel the emergence of the P4 language [P4] and programmable switches facilitates innovation and triggers new research. For example, the latest programmable switches make the concept of the totally programmable and dynamically reconfigurable network closer to reality. And, as distributed systems are increasingly based on memory instead of hard disks, distributed system-based application performance is increasingly constrained by network resources not computing.

However, there are some challenges when introducing in-network computing and caching:

- Limited memory size: tor example, the SRAM size [TOFINO] can be as small as tens of MBs.

- Limited instruction sets: the operations are mainly simple arithmetic, data (packet) manipulation and hash operation. Some switches can provide limited floating-point operation. This enables network performance tools like forward error correction but limits more advanced applications such as machine learning for congestion control for example.

- Limited speed/CPU processing capabilities: only a few operations can be performed on each packet to ensure line speed (tens of nano-seconds on fast hardware). Looping could allow a processed packet to re-enter the ingress queue, but with a cost of increasing latency and reducing forwarding capabilities.

- Performance: for devices located on the links of the distributed network; it is to be evaluated how on-path processing can reduce the FCT (Flow Completion Time) in data center network hence reduce the network traffic/congestion and increase the throughput.

The next sections of this draft review how some of these questions are currently being addressed in the research community.

3. State of the Art in DC Programmability

Recent research has shown that in-network computing can greatly improve the DC network performance of three typical scenarios: aggregate on path- computing, key-value (K-V) cache, and strong consistency. Some of these research results are summarized below.

3.1. In-Network Computing

The goals on on-path computing in DC is 1. to reduce delay and/or increase throughput for improved performance by allowing advanced packet processing and 2. to help reduce network traffic and alleviate

congestion by implementing better traffic and congestion management [REXFORD][SOULE][SAPIO].

However, in terms of research and implementation, there are still open issues that need to be addressed in order to fulfill these promises beyond what was mentioned in the previous section. In particular, the end-to-end principle which has driven most of the networking paradigms of the last 20 years is challenged when in-network computing devices are inserted on the ingress-egress path. This is still an open discussion topic.

The type of computing that can be performed to improve the DC performance is another of the open topics. Computing should improve performance but not at the expense of existing application degradation Computing should also enable new applications to be developed. At the time of this writing those include data intensive applications in workload mode with partition and aggregation functionality.

Data-intensive applications include big data analysis (e.g. data reduction, deduplication and machine learning), graph processing, and stream processing. They support scalability by distributing data and computing to many worker servers. Each worker performs computing on a part of the data, and there is a communication phase to update the shared state or complete the final calculation. This process can be executed iteratively. It is obvious that communication cost and availability of bottleneck resources will be one of the main challenges for such applications to perform well as a large amount of data need to be transmitted frequently in many-to-many mode. But already, there are several distributed frameworks with user-defined aggregation functions, such as mapReduce from Haddop [HADOOP], Pregel from Google [PREGEL], and DryadLinq from Microsoft [DRYAD]. These functions enable application developers to reduce the network load used for messaging by aggregating all single messages together and consequently reduce the task execution time. Currently, these aggregation functions are used only at the worker level. If they are used at the network level, a higher traffic reduction ratio can be reached.

The aggregation functions needed by the data intensive applications, have some features that make it suitable to be at least executed in a programmable. They usually reduce the total amount of data by arithmetic (add) or logical function (minima/maxima detection) that can be parallelized. Performing these functions in the DC at the ingress of the network can be beneficial to reduce the total network traffic and lead to reduced congestion. The challenge is of course not to lose important data in the process especially when applied to

different parts of the input data without considering the order and affect the accuracy of the final result.

In-network computing can also improve the performance of multipath routing by aggregating path capacity to individual flows and providing dynamic path selection, improving scalability and multitenancy.

Other data intensive applications that can be improved in terms of network load by in-network computing include: machine learning, graph analysis, data analytics and map reduce. For all of those, aggregation functions in the computing hardware provides a reduction of potential network congestion; in addition, because of the reduced load, the overall application performance is improved. The traffic reduction was shown to range from 48% up to 93% [SAPIO].

Machine learning is a very active research area for in-network computing because of the large datasets it both requires and generates. For example, in TensorFlow [TENSOR], parameters updates are small deltas that only change a subset of the overall tensor and can be aggregated by a vector addition operation. The overlap of the tensor updates, i.e. the portion of tensor elements that are updated by multi workers at the same time, is representative of the possible data reduction achievable when the updates are aggregated inside the network.

In graph analysis, three algorithms with various characteristics have been considered in [SAPIO]: PageRank, Single Source Shortest Path (SSSP) and Weakly Connected Components (WCC) with a commutative and associative aggregation function. Experiment shows that the potential traffic reduction ratio in the three applications is signification.

Finally, in map-reduce, experiments in the same paper show that after aggregation computing, the number of packets received by the reducer decreases by 88%~90% compared UDP, by 40% compared to using TCP. There is thus great promise for mapReduce-like to take advantage of computing and storage optimization.

3.2. In-Network Caching

Key-value stores are ubiquitous and one of their major challenges are to process their associated data-skewed workload in a dynamic fashion. As in any caches, popular items receive more queries, and the set of popular items can change rapidly, with the occurrence of well-liked posts, limited-time offers, and trending events. The skew generated by the dynamic nature of the K-V can lead to severe load imbalance and significant performance deterioration. The server is

either overused in an area or underused in another, the throughput can decrease rapidly, and the response time latency degrades significantly. When the storage server uses per core sharding/ partitioning to process high concurrency, this degradation will be further amplified. The problem of unbalanced load is especially acute for high performance in memory K-V store.

The selective replication copying of popular items is often used to keep performance high. However, in addition to more hardware resource consumption, selective replication requires a complex mechanism to implement data mobility, data consistency and query routing. As a result, system design becomes complex and overhead is increased.

This is where in-network caching can help. Recent research experiments show that K-V cache throughput can be improved by 3~10 times by introducing in net cache. Analytical results in [FAN] show that a small frontend cache can provide load balancing for N back-end nodes by caching only O(N logN) entries, even under worst-case request patterns. Hence, caching O(NlogN) items is sufficient to balance the load for N storage servers (or CPU cores).

In the NetCache system [JIN], a new rack-scale key-value store design guarantees billions of queries per second (QPS) with bounded latencies even under highly-skewed and rapidly-changing workloads. A programmable switch is used to detect, sort, cache, and obtain a hotspot K-V pair to process load balancing between the switch storage nodes.

3.3. In Network Consensus

Strong consistency and consensus in distributed networks are important. Significant efforts in the in-network computing community have been directed towards it. Coordination is needed to maintain system consistency and it requires a large amount of communication between network nodes and instances, taking away processing capabilities from other more essential tasks. Performance overhead and extra resources often result in a decrease in consistency. And as a result, potential inconsistencies need to be addressed.

Maintaining consistency requires multiple communications rounds in order to reach agreement, hence the danger of creating messaging bottlenecks in large systems. Even without congestion, failure or lost messages, a decision can only be reached as fast as the network round trip time (RTT) permits. Thus, it is essential to find efficient mechanisms for the agreement protocols. One idea is to use the network devices themselves.

Hence, consensus mechanisms for ensuring consistency are some of the most expensive operations in managing large amounts of data [ZSOLT]. Often, there is a tradeoff that involves reducing the coordination overhead at the price of accepting possible data loss or inconsistencies. As the demand for more efficient data centers increases, it is important to provide better ways of ensuring consistency without affecting performance. In [ZSOLT] consensus (atomic broadcast) is removed from the critical path by moving it to hardware. The Zookeeper atomic broadcast (also in Hadoop) proof of concept is implemented at the network level on an FPGA, using both TCP and an application specific network protocol. This design can be used to push more value into the network, e.g., by extending the functionality of middle boxes or adding inexpensive consensus to in-network processing nodes.

A widely used protocol for consensus is Paxos. Paxos is a fundamental protocol used by fault-tolerant systems, and is widely used by data center applications. In summary, Paxos serializes transaction requests from different clients in the leader, ensuring that each learner (message replicator) in the distributed system is implemented in the same order. Each proposal can be an atomic operation (an inseparable operation set). Paxos does not care about specific content of the proposal. Recently, some research evaluation suggests that moving Paxos logic into the network would yield significant performance benefits for distributed applications [DANG15]. In this scheme network switches can play the role of coordinators (request managers) and acceptors (managed storage nodes). Messages travel fewer hops in the network, therefore reducing the latency for the replicated system to reach consensus since coordinators and acceptors typically act as bottlenecks in Paxos implementations, because they must aggregate or multiplex multiple messages. Experiments suggest that moving consensus logic into network devices could dramatically improve the performance of replicated systems. In [DANG15], NetPaxos achieves a maximum throughput of 57,457 messages/s, while basic Paxos the coordinator being CPU bound, is only able to send 6,369 messages/s. In [DANG16], a P4 implementation of Paxos is presented as a result of Paxos implementation with programmable data planes.

Other papers, have shown the use of in-network processing and SDN for Paxos performance improvements using multi-ordered multicast and multi-sequencing [LIJ] [PORTS].

3.4. Research Topics in Next Generation Data Centers

While the previous section introduced the state of the art in data center in-network computing, there are still some open issues that need to be addressed. In this section, some of these questions are

listed as well as the impacts that adding in-network computing will have on existing systems.

Adding computing and caching to the network violates the End-to-End principle central to the Internet. And the interaction with encrypted systems can limit the scope of what in-network can do to individual packet. In addition, even when programmable, every switch is still designed for (line speed) forwarding with the resulting limitations, such as lack of floating-point support for advanced algorithms and buffer size limitation. Especially in the high-performance datacenters for in-network computing to be successful, a balance between functionality, performance and cost must be found.

Hence the research areas include but are not limited to:

Protocol design and network architecture: a lot of the current in-network work has targeted network layer optimization; however, transport protocols will influence the performance of any in-network solution. How can in-network optimization interact (or not) with transport layer optimizations?

Can the end-to-end assumptions of existing transport like TCP still be applicable in the in-network compute era? There is heritage in middlebox interactions with existing flows.

Fixed-point calculation for current application vs. of floating-point calculation for more complex operations and services: network switches typically do not support floating-point calculation. Is it necessary to introduce this capability and scale to the demand? For example, AI and ML algorithms currently mainly use floating-point calculation. If the AI algorithm is changed to fixed-point calculation, will the training stop in advance and the training result deteriorate (this needs to be done as joint work with the AI community)?

What are the gains brought by aggregation in distributed and decentralized networks? The built-in buffer of the network device is often limited, and, for example the AI and XR application parameters and caches can reach hundreds of megabytes. There is a trade-off between aggregation of the data on a single network device and its distribution across multiple nodes in terms of performance.

What is the relationship between the depth of packet inspection and not only performance but security and privacy? There is a need to find what application layer cryptography is ready to expose to other layers and even collaborating nodes; this is also related to trust in distributed networks.

Relationship between the speed of creating tables on the data plane and the performance.

3.5. Conclusion

In-network computing as it applies to data centers is a very current and promising research area. Thus, the proposed Research Group creates an opportunity to bring together the community in establishing common goals, identify hurdles and difficulties, provide paths to new research especially in applications and linkage to other new networking research areas at the edge. More information is available in [COIN].

4. References

4.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

4.2. Informative References

[CHAN] Chan et al., M., "Network Support for DNN Training", 2018.

[COIN] He, J., Chen, R., and M. Montpetit, "Computing in the Network, COIN, proposed IRTF group", 2018.

[DANG15] Dang et al., T., "NetPaxos: Consensus at Network Speed", 2015.

[DANG16] Dang et al., T., "Paxos Made Switch-y", 2016.

[DRYAD] Microsoft, "DryadLinq", 2018.

[FAN] Fan et al., B., "Small Cache, Big Effect: Provable Load Balancing for Randomly Partitioned Cluster Services", 2011.

[FORSTER] Forster, N., "To be included", 2018.

[GRAHAM] Graham, R., "Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction.", 2016.

[HADOOP] Hadoop, "Hadoop Distributed Filesystem", 2016.

[JIN] Jin et al., X., "NetCache: Balancing Key-Value Stores with Fast In-Network Caching", 2017.

[LIJ] Li et al., J., "NetCache: Balancing Key-Value Stores with Fast In-Network Caching", 2017.

[LIX] Li et al., X., "Be fast, cheap and in control with SwitchKV", 2016.

[MEM] memcached.org, "Memcached", 2018.

[P4] p4.org, "P4 Language", 2018.

[PORTS] Ports, D., "Designing Distributed Systems Using Approximate Synchrony in Data Center Networks", 2015.

[PREGEL] github.com/igrigorik/pregel, "Pregel", 2018.

[REXFORD] Rexford, J., "Sigcomm 2018 Keynote Address", 2018.

[SAPIO] Sapio et al., A., "In net computing is a dumb idea whose time has come", 2017.

[SOULE] Soule, R., "Sigcomm 2018 Netcompute Workshop Keynote Address", 2018.

[SPARK] Apache, "Spark Graph X", 2018.

[SUBEDI] Subedi et al., T., "OpenFlow-based in-network Layer-2 adaptive multipath aggregation in data centers", 2015.

[TENSOR] tensorflow.org, "Tensorflow", 2018.

[TOFINO] BarefootNetworks, "Tofino", 2018.

[ZSOLT] Zsolt et al., I., "Consensus in a Box", 2018.

Authors' Addresses

Jeffrey He
Huawei

Email: jeffrey.he@huawei.com

Rachel Chen
Huawei

Email: chenlijuan5@huawei.com

Marie-Jose Montpetit (editor)
Triangle Video
Boston, MA
US

Email: marie@mjmontpetit.com