The Case for a Rackless Data Center Network Architecture

Dingming Wu, Ang Chen, T. S. Eugene Ng
Rice University

CCS CONCEPTS

• **Networks** → **Data center networks**;

KEYWORDS

Data Center Network; Network Architecture

ACM Reference Format:
Dingming Wu, Ang Chen, T. S. Eugene Ng. 2018. The Case for a Rackless Data Center Network Architecture. In *SIGCOMM Posters and Demos '18: ACM SIGCOMM 2018 Conference Posters and Demos, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3234200.3234233

1 INTRODUCTION

Today's data centers consist of tens of thousands of servers connected by multi-layered Clos networks. To reduce equipment and operational cost, the network is typically oversubscribed at the core—sometimes with an oversubscription ratio of 240 [8]. This means that the bandwidths between servers can vary drastically, depending on how "close" they are on the physical topology: servers within the same rack enjoy full bisection bandwidth, but servers on different racks have much lower bandwidths. The oversubscribed core can be easily congested by concurrent, inter-rack flows, causing inflated latency and poor application performance; and the edge links tend not to be fully utilized due to congestion higher up the hierarchy [14]. For instance, a measurement study on DCN traffic characteristics shows that more than 98% of the links are utilized less than 1% [4].

Therefore, recent work has considered ways of making better use of the DCN bandwidth, including using advanced transport protocols [3, 13], optimized flow scheduling [2, 7], and intelligent job placement [5, 10]. They attack the problem from different angles—advanced transport protocols can mitigate congestion more effectively, better scheduling algorithms can lead to higher throughput, and careful job placement and execution strategies could reduce inter-rack traffic. All these approaches can and do lead to performance benefits, but the rack boundaries pose physical and topological constraints that are inherently challenging to get around.

We, instead, ask a very different question: Can we *remove* the rack boundaries and adopt a *rackless data center* (RDC) network architecture? In an RDC network, servers are no longer bound to a rack; rather, their connectivity to other servers can be reconfigured dynamically, so that a server A can reach any other server B via a similar network hop distance regardless of their physical locations. At any point in time, the servers may still form "locality groups" where intra-group communication is cheaper than inter-group communication, but locality groups only represent *logical boundaries* that can be quickly established and removed by real-time reconfiguration of the network topology.

RDC could lead to unique benefits. For instance, if a job proceeds in multiple stages where each stage involves a different traffic pattern (e.g., MPI applications such as QBox [9]), RDC could form different locality groups optimized for each stage. Or, if a VM migrates from one server to another server, the locality group could exclude the former and include the latter. More generally, regardless of the underlying reason for traffic pattern change, servers that heavily communicate with each other can be grouped together on-demand, and they can be regrouped as soon as the pattern changes again. Instead of optimizing the workloads for the topology, RDC optimizes the topology for the changing workloads.

RDC is achievable by circuit switches, which have been used for physical-layer topology adaptation in many novel network architectures [6, 15, 16]. Circuit switches are deployed only at the DCN edge, with the up-ports connecting to the ToRs and down-ports connecting to the servers. Theoretically, all the servers and ToRs can be connected to a single giant circuit switch. Through circuit reconfiguration, we can allow any server to talk to any other server via any ToR. However, a circuit switch has limited port-count, e.g., 384 [1], so this rackless flexibility can only be enabled per server group of size 192 (the other 192 ports would be used to connect to the ToR switches), or 4-5 racks if each rack has 40 servers. To scale out, we can stripe the server and ToR links across distributed circuit switches. The loss of flexibility due to such

SIGCOMM Posters and Demos '18, August 20–25, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5915-3/18/08…$15.00
https://doi.org/10.1145/3234200.3234233

link striping is small; each server can still connect to any ToR, and locality groups can still be dynamically formed by servers under different circuit switches.

This rackless architecture also raises several interesting open problems. Standard server racks enable unified power supply and cooling, and lead to considerable space and cable savings; can RDC achieve a competitive *operational cost*? A uniformly non-blocking DCN has well-understood performance guarantees; can RDC deliver a similar *performance*? Can RDC co-exist with today's data center networks in a *partial deployment*? Do applications running on an RDC require customized *scheduling algorithms*? We seek to answer all these questions in our ongoing work, but we only sketch an initial design of RDC in the rest of this paper.

2 THE RDC NETWORK ARCHITECTURE

Building Blocks: Circuit Switching. RDC relies on switching technologies that provide on-demand point-to-point connectivity at high speed and low cost. Circuit switches do not decode/encode or buffer packets. An electrical crosspoint circuit switch has a reconfiguration delay of O(10) nanoseconds [12]. This is fast enough to be transparent to the upper layer applications; packet loss during circuit reconfiguration can be rapidly recovered by fast retransmission without triggering TCP timeouts. Circuit switches have relatively simple hardware design and bare-minimum software stack, leading to significantly lower cost compared to packet switches. For example, a commercial 160-port 10Gbps electrical crosspoint switch has per-port cost of just $3 [12].

"One Big Circuit Switch" (OBCS). RDC does not make any changes to the topology or control plane of the network core, as the circuit switches are inserted at the edge, forming a "One Big Circuit Switch" that connects all ToR switches and all servers. Fig 1(a) shows an example of RDC with 3 racks and 9 servers. In today's architecture, each server has a fixed connection to a single ToR: servers 0-2 are connected to ToR0, 3-5 are connected to ToR1, and 6-8 are connected to ToR2. In RDC, we have full flexibility to permute the server-ToR connectivity, allowing runtime topology optimization for dynamic traffic demands. For example, with the traffic demand shown in Fig 1(b), flows $0 \rightarrow 3$ and $1 \rightarrow 5$ can be served by simply flipping the physical links of servers 1 and 3. Without RDC, these two flows would have to go through the oversubscribed core and experience lower performance.

Distributed OBCS. A single circuit switch has limited port-count and may represent a single point of failure. To achieve higher scalability and fault tolerance, we could use a link striping mechanism with a distributed set of circuit switches. As shown in Fig. 2, the downlinks of each ToR are striped across three circuit switches. Servers can be rearranged for better visualization of the original server-ToR relationships. Using distributed circuit switches introduces

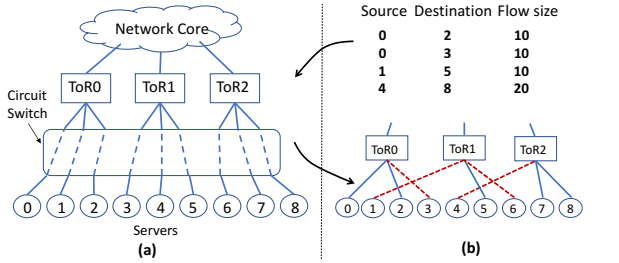

Figure 1: An example of RDC architecture. (a) shows where the circuit switch is deployed in the existing DCN; (b) shows how server-ToR connectivities are reconfigured according to traffic demand among servers. Reconfigured links are shown in red dashed lines.

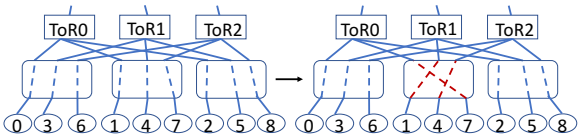

Figure 2: An example of RDC's link striping mechanism using distributed circuit switches for the workload in Figure 2(b).

a certain degree of flexibility loss as server permutation can only be performed per circuit switch. A locality group of servers must be connected to different circuit switches. For the traffic demand in Fig. 1(b), flow $0 \rightarrow 3$ cannot be localized as server 0 and 3 are connected to the same circuit switch whose uplinks are distributed across different ToRs. However, servers still have "coordinated flexibility" to connect to any one of the ToRs. Flows $1 \rightarrow 5$ and $4 \rightarrow 8$ can both be localized by simply reconfiguring the second circuit switch.

3 ONGOING WORK

We plan to further study the control plane design of RDC. It can be installed independently and does not interfere with existing DCN control system. Particularly, we need to figure out how to reconfigure the circuits under a given traffic matrix to minimize the amount of traffic across locality groups. This can be formulated as a balanced graph partitioning problem (BGP) [11]. The traffic matrix is a graph $G = (E, V)$. V is the vertex set in which each vertex represents a server and E is the edge set. The weight of an edge e, $w(e)$, is the traffic volume between the end vertices. BGP partitions the graph into subgraphs of equal number of vertices such that the weight sum of cross-subgraph edges is minimized. BGP is NP-hard, but it has polynomial approximation guarantees.

In the case of distributed OBCS, there are additional constraints: servers under the same circuit switch must be partitioned into different subgraphs. This adds another layer of complexity. We plan to develop heuristics that can optimize for different traffic patterns.

ACKNOWLEDGEMENT

This research was sponsored by the National Science Foundation under the grant CNS-1422925 and CNS-1718980.

REFERENCES

[1] Polatis 7000 series. https://www.phoenixdatacom.com/product-group/optical-matrix-switching.

[2] M. Al-Fares et al. Hedera: Dynamic flow scheduling for data center networks. In *NSDI*. USENIX, 2010.

[3] M. Alizadeh et al. Data center tcp (dctcp). In *SIGCOMM*. ACM, 2010.

[4] T. Benson et al. Network traffic characteristics of data centers in the wild. In *IMC*. ACM, 2010.

[5] P. Bodík et al. Surviving failures in bandwidth-constrained datacenters. In *SIGCOMM*. ACM, 2012.

[6] A. Chatzieleftheriou et al. Larry: Practical network reconfigurability in the data center. In *NSDI*. USENIX, 2018.

[7] M. Chowdhury et al. Efficient coflow scheduling with varys. In *SIGCOMM*. ACM, 2014.

[8] A. Greenberg et al. Vl2: a scalable and flexible data center network. In *SIGCOMM*. ACM, 2009.

[9] F. Gygi. Architecture of qbox: A scalable first-principles molecular dynamics code. *Journal of Research and Development*, 2008.

[10] V. Jalaparti et al. Network-aware scheduling for data-parallel jobs: Plan when you can. *SIGCOMM*, 2015.

[11] R. Krauthgamer et al. Partitioning graphs into balanced components. In *SODA*. Society for Industrial and Applied Mathematics, 2009.

[12] S. Legtchenko et al. Xfabric: A reconfigurable in-rack network for rack-scale computers. In *NSDI*. USENIX, 2016.

[13] C. Raiciu et al. Improving datacenter performance and robustness with multipath tcp. In *SIGCOMM*. ACM, 2011.

[14] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the social network's (datacenter) network. *SIGCOMM Comput. Commun. Rev.*, 45(4):123–137, Aug. 2015.

[15] D. Wu et al. Masking failures from application performance in data center networks with shareable backup. In *SIGCOMM*. ACM, 2018.

[16] Y. Xia et al. A tale of two topologies: Exploring convertible data center network architectures with flat-tree. In *SIGCOMM*. ACM, 2017.