Say No to Rack Boundaries: Towards A Reconfigurable Pod-Centric DCN Architecture

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

ABSTRACT

Data center networks are designed to interconnect large clusters of servers. However, their static, rack-based architecture poses many constraints. For instance, due to over-subscription, bandwidth tends to be highly imbalanced—while servers in the same rack enjoy full bisection bandwidth through a top-of-rack (ToR) switch, servers across racks have much more constrained bandwidth. This translates to a series of performance issues for modern cloud applications. In this paper, we propose a rackless data center (RDC) architecture that removes this fixed "rack boundary". We achieve this by inserting circuit switches at the edge layer, and dynamically reconfiguring the circuits to allow servers from different racks to form "locality groups". RDC optimizes the topology between servers and edge switches based on the changing workloads, and achieves lower flow completion times and improved load balance for realistic workloads.

CCS CONCEPTS

• **Networks** → **Network design principles**; **Physical topologies**; **Data center networks**;

KEYWORDS

Network topology, rackless network, circuit switching

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—with typical over-subscription ratios somewhere between 5:1 to 20:1 [24]. This means that the bandwidths between servers can vary drastically, depending on how "close" they are on the physical topology. Servers communicating in the same rack enjoy full bisection bandwidth, but servers communicating across racks have much lower bandwidths. As a result, the oversubscribed layers can be easily congested by 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 in the hierarchy [39]. For instance, a measurement study shows that even though core links tend to be highly utilized, more than 60% of the edge links are utilized less than 1% [16].

Recent work has looked at making better use of the bandwidth by the design of advanced transport protocols [13, 38], flow scheduling algorithms [12, 20], and job placement strategies [17, 27]. Advanced transport protocols aim to mitigate congestion, scheduling algorithms can lead to higher throughput, and careful job placement and execution strategies reduce inter-rack traffic. All these approaches can and do lead to performance benefits, but the rack boundaries pose physical constraints that are inherently challenging to get around.

At the same time, recent measurement studies show that more and more DCN traffic is escaping the rack boundary and becoming "pod-local" [39], where a pod consists of multiple racks of servers in the same cluster. This is not only due to the ever increasing scale of jobs, but also because racks tend to host servers of similar types—e.g., one rack may host database servers, and another may host cache servers. Therefore, servers on one rack would inevitably need to coordinate with servers on other racks, producing inter-rack traffic [39].

Driven by this observation, we propose a novel DCN architecture specifically designed for pod-local traffic. We aim to develop a *"rackless"* data center (RDC) network architecture to *remove* the fixed, topological rack boundaries in a pod. In RDC, servers are still mounted on physical racks, but they are not bound statically to edge switches. Rather, servers can move from one edge switch to another. Under the hood, this is achieved by the use of circuit switches, which can be dynamically reconfigured to form different connectivity patterns. In other words, servers remain immobile on the racks, but circuit changes may shift them to different topological locations.

In RDC, circuit switches are deployed at the edge layer, with the up-ports connected to the edge switches, and the down-ports connected to the servers. At any point in time, servers with circuits connecting them to the same edge switch form a locality group where they enjoy full communication

SOSR '19, April 3–4, 2019, San Jose, CA, USA
© 2019 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-6710-3/19/04. . . $15.00
https://doi.org/10.1145/3314148.3314350

bandwidth. However, locality groups represent logical boundaries that can be established and removed quickly via on-demand topology reconfiguration. In a traditional data center architecture, servers under the same ToR switch form a fixed locality group, the size of which is on the order of tens. In contrast, in RDC, thousands of servers connected to the same circuit switch can potentially form locality groups when a high-port-count circuit switch is used [31, 35]. This flexibility means that a server can have the opportunity to talk to many more servers with full bandwidth via circuit reconfigurations.

RDC could lead to unique benefits, because it can dynamically form locality groups that are optimized for the underlying traffic pattern. Traffic patterns can change in a DCN at different timescales, depending on the underlying dataset being processed, the particular placement of workers, and many other factors. Even for a single job, it may proceed in multiple stages, with each stage exhibiting a different traffic pattern (e.g., for applications such as SAGE [29] and QBox [25]). As a result, there does not exist a statically optimal locality group that would consistently outperform other configurations across workloads, worker placements, and job stages. RDC, however, avoids the problem of having to stick to any static configuration. Rather, 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.

Specifically, RDC leads to several key benefits:

- **Increasing traffic locality.** RDC dynamically re-groups servers that communicate heavily with each other via circuit reconfiguration, so that a higher percentage of traffic would enjoy full bandwidth due to the increased locality. Our results show that RDC speeds up flow completion time (FCT) by 4.87× on realistic workloads.
- **Balancing inter-pod traffic.** RDC redistributes servers across a pod to balance the load on the uplinks of edge switches, mitigating congestion.
- **Increasing resilience to edge switch failures.** In today's DCN architecture, an edge switch failure would disconnect all servers in a rack. In RDC, the disconnected servers can be quickly reconnected to a different edge switch for recovery.

We also propose a modular packaging design of RDC pods for a pod to be incrementally deployed in a Clos network. The extra cost of a pod is estimated to be 3.1% more than that of a traditional pod, which seems low enough to be practical.

2 MOTIVATION

Next, we present a set of measurement results over production traffic traces, and discuss how RDC can improve DCN efficiency. The traces are collected from three clusters ("frontend", "database" and "Hadoop") in Facebook data centers in a one-day period. They contain packet-level traces with a sampling rate of 1:30 k, and each sampled packet contains information about the source and destination racks in the data center [1]. To simulate the effect of removing rack boundaries, we simply apply a greedy algorithm that re-groups servers in different racks under a "hypothetical" rack. At the highest level, we found that DCN traffic patterns exhibit pod locality, but not rack locality, and that traffic across racks can be heavily imbalanced. As a result, removing rack boundaries gives us significant opportunity to exploit pod locality and achieve better load balance across racks.

2.1 Observation #1: Pod locality

Figure 1(a) shows the heatmap for the traffic patterns of a representative pod with 74 racks in a 2-minute interval in the frontend trace. If a server in rack i sends more traffic to another server in rack j, then the pixel for (i,j) in the heatmap will become darker. In this presentation, intra-rack traffic always appears on the diagonal (i.e., $i = j$). As we can see, the scattered dots show that the traffic does not exhibit rack locality—in fact, 96.26% of the traffic in this heatmap is inter-rack but intra-pod. Similar trend exists for the database trace: 92.89% of traffic is inter-rack but intra-pod (figure omitted). Hadoop trace has more intra-rack traffic, but still has 52.49% of traffic being inter-rack but intra-pod (figure omitted). This observation is consistent with a larger-scale study of Facebook's traffic patterns, which shows that over 70% of the traffic is pod-local, but only 18.3% is rack-local [39].

Re-grouping servers improves locality. Figure 1(b) shows the heatmap in a hypothetical data center where servers are re-grouped under different racks based on their communication intensity, simulating the case RDC aims to achieve. Here, most of the traffic is on the diagonal and that inter-rack traffic has been reduced significantly to 38.4%. For the database and Hadoop traces, the inter-rack traffic after re-grouping is 28.4% and 41.6%, respectively. Thus, we believe that enormous optimization opportunities exist if we were able to dynamically re-group servers under different physical racks.

2.2 Observation #2: Inter-pod imbalance

Another trend we observe is the heavy imbalance of inter-pod traffic, which can also benefit from re-grouping servers. Figure 1(c) sorts the racks based on the amount of inter-pod traffic they sent (traffic trace: database) in a 20-min interval where the X-axis is the rack ID, and the Y-axis is the (normalized) amount of inter-pod traffic a rack sends. As we can see, the top 11 racks account for nearly 50% of the inter-pod traffic, and almost half of the racks never sent traffic across pods. This means that certain uplinks of edge switches are heavily utilized, whereas other links are almost always free. The load imbalance, defined as $max(L_i)/avg(L_i)$, where L_i is

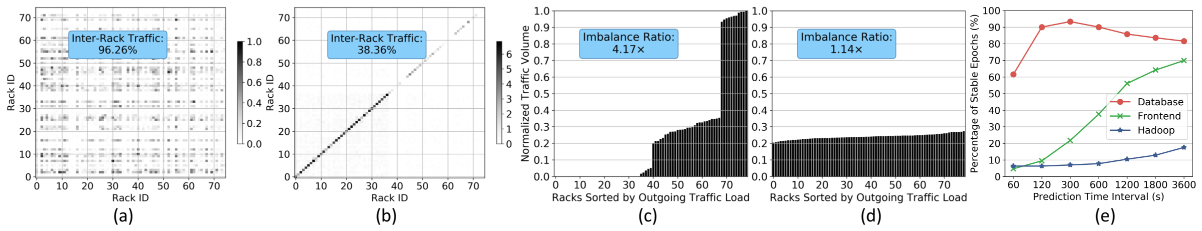

Figure 1: Traffic patterns from the Facebook traces. (a) Rack level traffic heatmap of a representative frontend pod. (b) Server communication heatmap after re-grouping servers in (a). (c) and (d) plot the sorted load of inter-pod traffic across racks in a representative database pod, before and after server re-grouping, respectively. (e) Traffic stability over different timescales.

the amount of inter-pod traffic from rack i, is as much as 4.17. We found similar observations for other traces.

Re-grouping servers mitigates load imbalance. Figure 1(d) shows the results for re-grouped servers. The inter-pod traffic has a load imbalance of 1.14, which leads to a better use of the inter-pod links, and avoids congesting any particular link due to traffic imbalance.

2.3 Observation #3: Predictable patterns

Our third observation is on the degree of predictability of traffic patterns. This does not speak to the benefits of RDC, but its feasibility. We can only dynamically re-group servers based on traffic patterns if the patterns are stable—if traffic patterns change without any predictability, then it would be difficult to find a suitable reconfiguration strategy.

Therefore, we analyzed the stability of inter-rack traffic at different time intervals within a pod, using a metric that is similar to that in MicroTE [15]. We divide time into epochs, and for each epoch, we measure the amount of traffic exchanged between a pair of servers. This would result, for each epoch i, a set of triples (s, r, v), where s is the sender, r is the receiver, and v is the exchanged volume. Then, if we have a triple (s, r, v_1) for epoch i, and a triple (s, r, v_2) for epoch $i + 1$, and $\frac{|v_1 - v_2|}{v_2} < 0.2$, then we consider the communication pattern between s and r to be stable across these two epochs. Figure 1(e) plots the percent of such "stable triples" for the three traces. As we can see, the database trace is highly stable when an epoch lasts from two minutes to one hour. If an epoch is an hour, then 81% of the triples are stable. Although the Hadoop trace is less stable, it still has 18% stable triples for one-hour epochs. We note that these traces are *sampled* from the original traffic, which may exhibit higher stability at finer timescales. Nevertheless, the highest-level takeaway is that the stability of traffic patterns is notable, and this could be learned from past traffic patterns. RDC could then use the stability patterns to determine its reconfiguration period.

3 RELATED WORK

Transport protocols. Recent research projects have developed optimized DCN transports. DCTCP [13] uses an adaptive congestion control mechanism based on ECN. pFabric [14] uses in-network flow prioritization. MPTCP [38] splits a flow onto different DCN paths. Other transport designs explicitly optimize for application-level goals, e.g., deadlines [41], or co-flow completion times [20]. In contrast, RDC optimizes the network topology instead of transport protocols.

Job placement. Job placement can also lead to performance improvement. Sinbad [21] selectively chooses data transfer destinations to avoid network congestion. ShuffleWatcher [10] attempts to localize the shuffle phase of MapReduce jobs to one or a few racks. Corral [27] jointly places input data and computes to reduce inter-rack traffic for recurring jobs. However, optimizations to a stage of an application typically do not generalize to other stages. Since traffic patterns may change dynamically over a job's lifetime, a reconfigurable network architecture can respond to changes in real time.

DCN architecture. Similar to RDC, several other DCN designs provision bandwidth on-demand with reconfigurable links to serve dynamic workloads, e.g., using electrical circuit switches [18], optical circuit switches [19, 22, 42], free space optics [23, 26], and even wireless radio [28, 44]. However, existing systems such as OSA [19], ProjecToR [23] and Firefly [26] require substantial changes to the current DCN architecture and control plane; but RDC allows for local and incremental deployment. c-Through [42], Helios [22], Flyways [28] and Zhou et al. [44] augment existing DCNs with out-of-band bandwidth. Instead, RDC aims to drive up the utilization of existing links. Larry [18] can also reconfigure the topology locally, but it operates at the aggregation layer above ToRs and adds extra links to the unused ports on each ToR. Compared to Larry, RDC removes rack boundaries and efficiently uses the existing bandwidth at the edge layer, and it requires no free ports from the ToRs.

4 THE RDC DESIGN

In this section, we present an initial design of the RDC architecture and its control system.

Circuit switching. RDC relies on switching technologies that provide on-demand point-to-point connectivity. Optical or electrical circuit switches do not decode/encode or buffer packets and are thus protocol or data rate transparent. They can provide ultra-high bandwidth at low power. For example, a 320-port optical circuit switch (OCS) based on 3D-MEMS

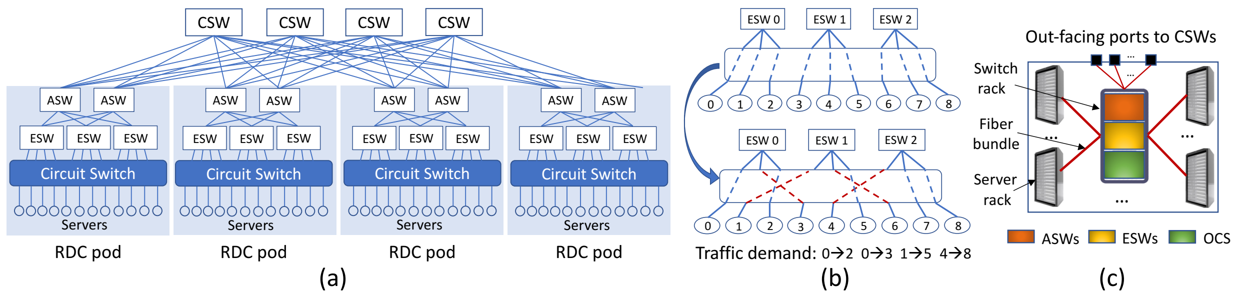

Figure 2: RDC architecture illustration. (a) An example of a 4-pod RDC. Circuit switches are inserted at the edge between servers and edge switches (ESW). Connectivities for aggregation switches (ASW) and core switches (CSW) remain the same as traditional Clos networks. (b) How circuits are reconfigured given the traffic demand among servers. Reconfigured links are shown in red dashed lines. (c) The physical packaging design of an RDC pod.

technology can provide >100Gbps with less than 45 Watts [2]. The switching speed of OCSes is usually several milliseconds, while the speed for electrical circuit switches can be as fast as a few nanoseconds [34]. An OCS with 1100 ports and 4dB insertion loss has already been fabricated [31]. To further scale up, the internal switching technology of an OCS may have to change significantly (e.g., using DMD-based switching [23]), or by sacrificing some degree of reconfigurability [35].

Architecture. RDC changes the traditional Clos topology [11, 24] by inserting circuit switches at the edge layer between servers and edge switches. The aggregation layer and the core layer of the network remain the same. Circuit switches are used per pod. Each pod has d servers and m edge switches, and requires $2d$ ports on the OCS [1]. For example, a 16-rack pod with 32 servers per rack requires 1024 ports on the OCS. Figure 2(a) shows an example RDC architecture with $d = 9$. The connectivities between edge switches (ESW) and aggregation switches (ASW) and between aggregation switches and core switches (CSW) are the same as those in Clos networks.

Figure 2(b) shows how RDC reconfigures its circuits given the traffic demand between servers. In traditional data centers, every server has a fixed connection to a single ESW: servers 0-2 are connected to ESW 1, 3-5 are connected to ESW 2, and 6-8 are connected to ESW 3. In other words, each locality group has a size of three. In RDC, however, we have full flexibility to permute the server-ESW connectivity, allowing runtime topology optimization for dynamic traffic demands. For example, with the traffic demand shown in Figure 2(b), flows $0 \longrightarrow 3$ and $1 \longrightarrow 5$ can be served by simply flipping the links of servers 1 and 3. Without RDC, these two flows would have to go through the oversubscribed aggregation or core layer and experience lower performance.

4.1 RDC reconfiguration

Traffic data collection. Circuit reconfigurations are handled by per-pod circuit managers that talk to the OCS in an out-of-band communication channel. A key property of RDC is that it does not require end host modification—servers do not need to report traffic demands actively. In RDC, each ESW maintains a byte counter for each server pair (s, d) where s is a server currently in its rack while d can be any other server in the same pod. The circuit manager collects these counters from all ESWs and then constructs the overall cross-server traffic data. The data collection period should be carefully chosen to attain a balance between speed and optimality, which could be dependent on the workload.

Reconfiguration algorithm. Our circuit manager takes cross-server traffic matrices as input, and produces an assignment that maps each server to an ESW, minimizing the amount of traffic traveling through the ASWs and CSWs. We formulate the server assignment problem as a balanced graph partitioning problem [32]. The traffic matrix is a graph $G = (E, V)$, where V is the vertex set (i.e., servers) and E is the edge set. The weight of an edge e, $w(e)$, is the traffic volume between the vertices. We need to partition the graph into subgraphs of equal numbers of vertices such that the weighted sum of cross-subgraph edges is minimized. We require partitions of the same size because each ESW must connect to exactly the same number of servers. The balanced graph partitioning problem is NP-hard, but fast parallel heuristics are available [33]. The computation is very fast (5-10 ms) due to the limited number of servers in a pod and can run in parallel with ongoing traffic transmission.

Routing. When servers are moved topologically, routing information needs to be updated to reflect that. At a high level, every switch has a forwarding table entry for all server IP addresses in the same pod. When a server is moved by a circuit reconfiguration, the server and its ESW would observe an interface down event followed by an interface up. Upon these events, the server would send a message to its newly connected ESW to announce its new address and location (a method that's similarly used in previous work [30, 36, 37]). This ESW then further updates its forwarding table entry accordingly, and propagates this change to its upper-layer ASWs to apply the changes. The previous ESW, to which the

[1]Half of the ports are connected to servers and the other half to edge switches.

server is no longer connected, executes a similar procedure for updates. When multiple simultaneous movements occur, updates can be further aggregated and propagated to ASWs together. Although updates may result in packet loss, RDC still guarantees that forwarding loops would not form.

Limitations. A full reconfiguration cycle of RDC includes a) obtaining historical traffic data from ESWs, b) computing the server-rack connectivities via a graph partition, c) reconfiguring the OCS and, d) updating the routing rules on ESWs and ASWs. Steps a) and b) can be performed without interrupting ongoing traffic transmissions, with the tradeoff that historical traffic data could be a bit stale as it does not capture ongoing traffic. In practice, however, this staleness should not be a serious problem, as the algorithm computation time is on a much shorter timescale (several milliseconds) than the traffic stability period (e.g., 10s of minutes). Steps c) and d) will cause a short downtime for servers. While previous work [43] has shown that OCS reconfiguration delay alone would not trigger TCP timeouts on 10G connections, it remains to be studied experimentally whether this OCS delay plus routing convergence delay would significantly disrupt TCP connections.

4.2 Feasibility analysis

Next, we discuss several practical issues in RDC.

Packaging. Figure 2(c) shows the packaging design of an RDC pod, which is somewhat different from that of a traditional pod. RDC has a central rack dedicated to hosting ESWs, ASWs, and the OCS. Server racks are placed around the switch rack. Each server rack is connected to the OCS via a fiber bundle to reduce wiring complexity. On the central rack, ESWs are connected to the OCS and ASWs using short fibers and cables, respectively. ASWs provide similar connectivity to CSWs outside the pod, just like in traditional data centers. The modular pod design allows the network to scale up easily by adding more RDC pods.

Cost. The cost of RDC can be estimated as follows. In a traditional pod, servers are connected to a ToR switch in the same rack via short, direct attach cables instead of fibers. ToR switches are connected to ASWs in a central switch rack via longer optical fibers. Assuming similar fiber bundling mechanisms, the wiring cost of RDC would be similar, with extra cost mostly coming from the OCS and per-server optical transceivers. Optical transceivers used to be the primary networking cost of data centers, but their price has declined sharply over the years due to massive production: a QSFP+ 40Gbps transceiver with 150m transmission distance costs only $39; a 10Gbps transceiver costs about $16. Based on feedback from a commercial optical switch vendor, we have learned that a) today's high port-count MEMS OCS has a per-port cost on the order of $500, but b) this high cost is mainly due to the low volume at which these devices are produced, and finally, c) the higher the volume, the lower the cost. For instance, if production volume goes up to 50 k units per year, the per-port cost can be dramatically reduced to below $50.

Table 1: Cost estimation of network components and their quantities needed by a) an RDC pod with 4:1 oversubscription b) a traditional pod with 4:1 oversubscription (Baseline) and c) a non-blocking pod (NBLK). "p.c." stands for personal communication.

Component	Price	Note	Source	Quantity Needed		
				RDC	Baseline	NBLK
Ethernet port	$200 per-port	40Gbps	[8]	768	768	1536
40G Transceiver	$39	QSFP+ 150m	[5]	1024	256	1024
Optical fiber, 8m	$4.4	inter-rack	[6]	512	128	512
Optical fiber, 3m	$3.4	intra-rack	[6]	512	0	0
Electrical cable, 3m	$40	intra-rack	[4]	128	512	512
OCS	$50 per-port	price at scale	p.c.	1024	0	0

As an example, consider an RDC pod with 4 ASWs and 16 ESWs (oversubscription ratio 4:1), and 512 servers, each of which has a 40 Gbps transceiver and connects to a 1024-port OCS. The required quantities and estimated prices for these components are listed in Table 1. The overall networking cost of a 4:1 oversubscribed traditional pod (baseline), an RDC pod, and a non-blocking pod is $184.6 k, $253.3 k, and $369.8 k, respectively. The extra cost of RDC compared to the 4:1 oversubscribed baseline is dominated by the OCS. However, RDC is significantly cheaper (by 31.5%) than the non-blocking network. Assuming that a server costs $4000, then the relative extra cost of RDC increase compared to the baseline is only 3.1% per pod. It is difficult to extrapolate the cost curve of an RDC pod beyond 1000s of servers, as the optical switching technology would need to be significantly different from the MEMS-based OCSes; we leave the cost estimate of larger-scale RDC as future work.

Handling OCS failures. OCSes are physical layer devices and highly reliable [40], with reported Mean-Time-Between-Failure (MTBF) being over 30 years [7]. For reliability, we can use 1:1 backup to handle switch failures. To achieve this, each server is connected to a primary OCS and a second backup OCS, via an inexpensive 1:2 optical splitter [3]. Traffic leaving the two OCSes is multiplexed via a cheap 2:1 optical MUX [9] and connected to the ESW. A similar setup is also used for the ESW to server traffic direction to interpose the primary and backup OCS in between. During normal operation, only the primary OCS transmits optical signals, and the backup's circuits are disabled. Upon primary failure, the backup OCS changes its circuit configuration based on the configuration of the primary. This would double the OCS cost, but does not require additional transceivers on both ends of the fibers. This brings the extra cost to about 6% for our RDC pod example, including the extra splitters and MUXes.

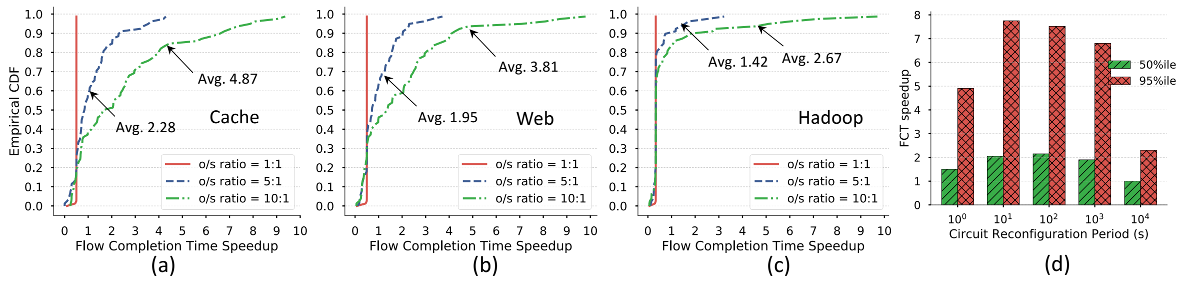

Figure 3: Distribution of FCT speedups for (a) cache (b) web and (c) Hadoop traffic under different oversubscription ratios. (d) The 50th and 95th percentiles of FCT speedups for cache traffic with varying circuit reconfiguration periods. The oversubscription ratio in (d) is 10:1.

5 INITIAL RESULTS

In order to perform an initial set of experiments, we have developed a flow-level, event-based network simulator to evaluate the potential benefits of RDC. Our simulator computes flow bandwidths using max-min fairness and without simulating the transport protocols. When a flow starts or ends, or if topology changes, the simulator recomputes the throughput for all impacted flows. Our experiments have been conducted on a simulated network with 128 racks and 32 servers per rack; each RDC pod has 16 racks, or 512 servers. The topology is a three-layer single-rooted tree, the uplinks from ESWs are oversubscribed with tunable ratios, and the circuit reconfiguration time is 5 ms. The CSW is not oversubscribed.

Metric. Our primary metric is the speedup of flow completion times (FCTs) in RDC due to its ability to localize traffic, using the Clos topology as the baseline. Evaluations of other aspects (e.g., load balancing, failure recovery) are left to future work.

Workloads. The original traces described in Section 2 have no flow-level information. We therefore generate our own workloads based on the sampled data and the reported results from [39]. Specifically, the source and destination servers of the flows are inferred from the sampled trace. The flow size and the flow arrival rate are sampled from Figures 6 and 14 in the same paper. We omit inter-DC traffic in the traces. The generated workloads cover three types of traffic (denoted as "cache", "web" and "Hadoop") that exhibit different characteristics. Cache traffic has strong pod locality, with around 88% of the traffic being intra-pod. The average flow size is 680 kB. Web traffic has similar pod locality (77% intra-pod traffic) but has a relatively smaller average flow size of 50 kB. Hadoop traffic is the least pod-local traffic among the three. It is instead highly rack-local—about 58% of the traffic is intra-rack. Its average flow size is about 200 kB.

Results. Figure 3 (a)-(c) plot the CDFs of FCTs normalized against the baseline, i.e., speedups[2], using a reconfiguration period of 10s. On average, RDC achieves speedups (>1.0) for all traffic. In cache and web traffic, more than 75% of the flows experience speedups. The average speedups are 2.28 and 4.87 for o/s (oversubscription) ratios 5:1 and 10:1, respectively. Web traffic observes similar benefits, with average speedups of 1.95 and 3.81 for different o/s ratios. Hadoop traffic, instead, sees FCT speedup for only 20% and 32% of the flows for o/s ratios 5:1 and 10:1. This is because most of the Hadoop flows are intra-rack and already have non-blocking bandwidth. The case of o/s ratio 1:1, in particular, reflects the net overhead of RDC compared to a non-blocking network. There are actually very few (<3%) flows being slowed down in this case, and the average speedups are all above 0.99. Such overhead shows that RDC may not be a good fit for a fully provisioned network, at least from the traffic localization perspective. Figure 3(d) shows the 50th and 95th percentiles of FCT speedups for the cache traffic under different reconfiguration periods. As we can see, cache traffic achieves the highest speedup when the period is on the order of 10-1000s. A small period leads to frequent reconfigurations and thus longer circuit downtime. On the contrary, a large period would not fully leverage the opportunity of traffic localization. Instead, we envision that an RDC network can dynamically adjust the period to best suit the observed workload patterns, and we are developing such an algorithm in ongoing work.

6 SUMMARY

We have presented RDC, a "rackless" pod-centric DCN architecture designed to break the traditional rack boundaries in a pod, creating the illusion that servers can move freely among edge switches in response to traffic pattern changes. Rather than optimizing the workloads based on the topology, RDC optimizes the topology to suit the changing workloads. RDC inserts circuit switches between the edge switches and the servers, and reconfigures the circuits on demand to form different connectivity patterns. Our preliminary results on intra-pod localization show that RDC can decrease flow completion times considerably on realistic workloads. As ongoing work, we are working on the RDC's dynamic reconfiguration algorithm, as well as a full RDC prototype.

Acknowledgement. We thank our shepherd, Dina Papagiannaki, and the anonymous reviewers for their valuable feedback. This work was partially supported by NSF grants CNS-1801884, CNS-1422925, CNS-1718980 and CNS-1815525.

[2]Speedup < 1.0 means that the transfer is actually slowed down.

REFERENCES

[1] Facebook network analytics data sharing. https://www.facebook.com/groups/1144031739005495/.

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

[3] 1x2 plc fiber splitter. https://www.fs.com/products/12493.html, 2018.

[4] 40g qsfp+ passive direct attach copper cable. https://www.fs.com/products/30898.html, 2018.

[5] 40G short range transceiver. http://www.fs.com/products/17931.html, 2018.

[6] 9/125 single mode fiber patch cable. https://www.fs.com/products/40200.html, 2018.

[7] Glimmerglass intelligent optical systems. https://www.laser2000.fr/out/groupdownloads/glimmerglass_product_overview(2).pdf, 2018.

[8] N8000-32q 40g sdn switch. https://www.fs.com/products/69342.html, 2018.

[9] Single fiber cwdm mux demux. https://www.fs.com/products/70407.html, 2018.

[10] F. Ahmad et al. Shufflewatcher: Shuffle-aware scheduling in multi-tenant mapreduce clusters. In *ATC*. USENIX, 2014.

[11] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 63–74. ACM, 2008.

[12] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: dynamic flow scheduling for data center networks. In *NSDI*, volume 10, pages 89–92, 2010.

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

[14] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pfabric: Minimal near-optimal datacenter transport. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 435–446. ACM, 2013.

[15] T. Benson, A. Anand, A. Akella, and M. Zhang. Microte: Fine grained traffic engineering for data centers. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, page 8. ACM, 2011.

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

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

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

[19] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen. Osa: An optical switching architecture for data center networks with unprecedented flexibility. *IEEE/ACM Transactions on Networking*, 22(2):498–511, 2014.

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

[21] M. Chowdhury, S. Kandula, and I. Stoica. Leveraging endpoint flexibility in data-intensive clusters. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 231–242. ACM, 2013.

[22] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, 40(4):339–350, 2010.

[23] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper. Projector: Agile reconfigurable data center interconnect. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 216–229. ACM, 2016.

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

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

[26] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer. Firefly: A reconfigurable wireless data center fabric using free-space optics. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 319–330. ACM, 2014.

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

[28] S. Kandula, J. Padhye, and P. Bahl. Flyways to de-congest data center networks. 2009.

[29] D. J. Kerbyson, H. J. Alme, A. Hoisie, F. Petrini, H. J. Wasserman, and M. Gittings. Predictive performance and scalability modeling of a large-scale application. In *Supercomputing, ACM/IEEE 2001 Conference*, pages 39–39. IEEE, 2001.

[30] C. Kim, M. Caesar, and J. Rexford. Floodless in seattle: a scalable ethernet architecture for large enterprises. *ACM SIGCOMM Computer Communication Review*, 38(4):3–14, 2008.

[31] J. Kim, C. J. Nuzman, B. Kumar, D. F. Lieuwen, J. S. Kraus, A. Weiss, C. P. Lichtenwalner, A. R. Papazian, R. E. Frahm, N. R. Basavanhally, D. A. Ramsey, V. A. Aksyuk, F. Pardo, M. E. Simon, V. Lifton, H. B. Chan, M. Haueis, A. Gasparyan, H. R. Shea, S. Arney, C. A. Bolle, P. R. Kolodner, R. Ryf, D. T. Neilson, and J. V. Gates. 1100 x 1100 port mems-based optical crossconnect with 4-db maximum loss. *IEEE Photonics Technology Letters*, 15(11):1537–1539, Nov 2003.

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

[33] D. LaSalle and G. Karypis. Multi-threaded graph partitioning. In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 225–236. IEEE, 2013.

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

[35] W. M. Mellette, G. M. Schuster, G. Porter, G. Papen, and J. E. Ford. A scalable, partially configurable optical switch for data center networks. *Journal of Lightwave Technology*, 35(2):136–144, 2017.

[36] A. Myers, E. Ng, and H. Zhang. Rethinking the service model: Scaling ethernet to a million nodes. In *Proc. HotNets*. Citeseer, 2004.

[37] R. Perlman. Rbridges: transparent routing. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1211–1218. IEEE, 2004.

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

[39] 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.

[40] T. J. Seok, N. Quack, S. Han, W. Zhang, R. S. Muller, and M. C. Wu. Reliability study of digital silicon photonic mems switches. In *Group IV Photonics (GFP), 2015 IEEE 12th International Conference on*, pages 205–206. IEEE, 2015.

[41] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter tcp (d2tcp). *ACM SIGCOMM Computer Communication Review*, 42(4):115–126, 2012.

[42] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan. c-through: Part-time optics in data centers. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 327–338. ACM, 2010.

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

[44] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng. Mirror mirror on the ceiling: Flexible wireless links for data centers. *ACM SIGCOMM CCR*, 42(4):443–454, 2012.