On the Impact of Indirect WAN Routing on Geo-Replicated Storage

Raziel Carvajal Gomez, Eduard Luchian, Iustin-Alexandru Ivanciu, Adrian Taut, Virgil Dobrota, Etienne Rivière

On the Impact of Indirect WAN Routing on Geo-Replicated Storage

Raziel Carvajal Gómez*, Eduard Luchian†, Iustin-Alexandru Ivanciu†,
Adrian Taut†, Virgil Dobrota† and Etienne Rivière*
**University of Neuchâtel, Switzerland* and †*TU Cluj-Napoca, Romania*

***Abstract*—Micro-clouds infrastructures allow supporting applications on local and energy-efficient resources. Communication between micro-clouds takes place on shared and non-dedicated Internet links. Network control and optimization can only happen at the edge. For availability and persistence, the storage of application data must be geo-replicated. Maintaining strong data consistency under concurrent accesses requires delay-sensitive coherence protocols, linking the performance of the storage to that of the network linking micro-clouds. We evaluate if the use of network control at the edge of a European-wide multi-site testbed, together with appropriate network monitoring, can allow improving the performance of ZooKeeper, a strongly-consistent replicated store. Our approach leverages the indirect routing of coherence protocol traffic in the presence of network triangle equality violations. We analyze the impact on storage of variations in WAN performance, and show how the use of traffic redirection can help reducing it.**

Keywords: Indirect routing; Geo-replicated storage; Cloud storage; Performance; Evaluation.

1. Introduction

Micro-clouds. We observe a shift from monolithic to multi-site cloud infrastructures. On the one hand, major cloud players and Internet companies operate collections of few, but large data centers, spread geographically and serving a worldwide users population. These data centers are typically linked through high-performance, dedicated wide-area networks (WAN) [1]. On the other hand, alternative cloud providers are deploying infrastructures formed of larger numbers of modestly-sized data centers, termed as *micro-clouds* [2]. This shift to micro-clouds is often motivated by energy efficiency considerations. It is possible for instance to deploy micro-clouds close to sources of renewable energy. A second important rationale for the use of micro-clouds is access locality. Users can interact with close-by micro-clouds, resulting in better responsiveness in interactive applications.

Network performance. In contrast with the dedicated network infrastructure used by large companies for interconnecting their data centers, micro-clouds deployments typically use regular Internet connections that are not owned by the cloud provider itself. The control by the cloud provider over the network is limited to the links at its edge. Network performance metrics between micro-clouds, such as delays and available transfer rates, are influenced by factors such as extraneous Internet traffic or BGP routing decisions. None of these factors is under the control of the cloud provider.

The performance experienced by users of cloud applications is not only influenced by the quality of their connection to a local micro-cloud, but also to a large extent by the performance of the network between application components lying on different micro-clouds. In some cases, due to load variations inside the network core, the standard direct routing between two micro-clouds may not be the best possible choice. There are in particular examples of network triangle inequalities [3], where the performance of a link between two micro-clouds A $\rightarrow$ B is lower than that of the combination of two links A $\rightarrow$ C and C $\rightarrow$ B, C being a third micro-cloud used as a proxy for *indirect routing* [4].

Cloud storage. Efficient storage is a key enabler for modern cloud applications, driving performance, scalability, and availability in the presence of failures. The need for high horizontal scalability led to the advent of non-relational storage, or NoSQL. Representative of such systems are key/value (e.g., Apache Cassandra & Infinispan) and document stores (e.g., MongoDB) used to manage application data, or coordination kernels such as Apache ZooKeeper [5] used for synchronization and metadata management. NoSQL storage systems use replication to ensure data persistence and availability. Data is copied to several servers, or replicas. The failure of one server does not result in data loss. Replication requires however that the different copies of a data item be kept synchronized under concurrent accesses. A coherence protocol implements this synchronization, which requires network exchanges between replicas to agree on the order in which updates should be visible to applications, and to propagate new values.

Geo-replication. Replication in a micro-clouds environment requires considering that due to the lack of redundancy in hardware and networking, an entire micro-cloud can fail and not only a single server. As a result, preventing the loss of data or the unavailability of applications requires hosting replicas on servers that are located at different sites, leading to *geo-replication*. Replicating data on several sites has the added advantage of allowing access from users to the closest copy. Geo-replication requires however that the coherence protocol exchanges synchronization messages between sites, subject to WAN latencies. Typically, several rounds of such WAN communications are required before being able to return the call from the application, leading to cumulatively high latencies. It is therefore of critical importance to maintain the performance of the underlying network as good as possible, and limit the impact of geo-replication on the performance of applications.

Motivation. We are interested in evaluating the potential for dynamic network optimization at the edge of a multi-site environment hosting several micro-clouds. Our target is strongly-consistent geo-replicated NoSQL storage. We wish to answer the following three research questions:

▷ What is the impact of WAN routing between micro-clouds on the performance of geo-replicated storage?
▷ Can network performance indicators at the edge of the network allow driving network optimizations decisions and in particular the use of indirect routing?
▷ Can we dynamically optimize multi-site routing and what is the impact on storage performance?

Contributions. Our research methodology and contributions are as follows. We present the deployment and setup of a four-site testbed across four countries in Europe. Each site is equipped with network function virtualization (NFV), in the form of an edge server supporting an Open vSwitch router [6]. The use of tunneling for application flows, along with the orchestration of edge routers by a common software-defined networking (SDN) controller for all four sites, allows supporting the dynamic and transparent use of indirect routing in the testbed. We present the addition of network monitoring capabilities at the edge using active probing. In particular, we report on an implementation of a cyclic-path active delay measurement technique adapted to a multi-site context. Our study considers a representative strongly-consistent NoSQL storage system, Apache ZooKeeper [5], typically used for application metadata and for coordination purposes. Based on measurements collected in real-time on the platform and under varying network load, we drive the network optimization by switching between direct and indirect routes, and analyze the impact on performance.[1]

Our results show that network triangle inequalities can be observed in our testbed, and that they negatively affect storage performance, with the coherence protocol playing a key role in this degradation. Applying indirect routing rules dynamically allow mitigating to a large extent the effect of network triangle inequalities while incurring a minimal impact for the transition. Our results are encouraging and motivate the use of network adaptation in multi-site environments where control resides only at the edge.

Outline. The remainder of this paper is organized as follows. We start by discussing related work in Section 2. We further detail the network requirements of a geo-replicated ZooKeeper in Section 3. We present our four-node testbed, networking and monitoring in Section 4. We present and discuss our evaluation results in Section 5 and conclude in Section 6.

1. We leave the addition of automated and software-defined adaptation based on network performance indicators to future work. We concentrate on establishing whether such an adaptation is feasible, and beneficial for geo-replicated storage running in multi-micro-cloud environments.

2. Related Work

Previous work emphasized the interest of using indirect routing in various scenarios. Opos et al. [4] show that indirect routing can increase network throughput for at least 30% of a sample of nodes from the PlanetLab testbed. Andersen et al. [7] confirm the assumption that indirect paths between Internet hosts can lead to better resilience when errors on these paths take place at uncorrelated points in time. Wang et al. [8] further confirm the existence of network triangle inequality violations and suggest that distributed systems be aware of this fact, as we intend to achieve.

Monitoring network performance from the edge requires specific solutions. Throughput is typically monitored using a combination of active probing and Kalman filtering [9]. One-way delays are also estimated based on active probing, as we discuss in Section 4.

Network function virtualization and SDN [10] are used to optimize the network layer based on application requirements. Mohammadkhan et al. [11] present a solution to the problem of network function placement in a data center network. This approach is linked to the convergence of SDN and NFV as discussed by Wood et al. [12].

Storage performance optimization in a multi-site environment involves several aspects beyond networking. Considering the impact of failures, Frezewd et al. [13] analyze the placement of replicas on micro-clouds with a focus on recovery time, while Narayanan et al. [14] formalize the optimal capacity allocation for sustained performance under such recoveries. Volley [15] considers the more general problem of service placement in geo-distributed cloud services. Yi et al. [16] study alternative coherence mechanisms for edge clouds. Other approaches [17], [18] reconfigure the geo-replication itself, either by moving replicas or changing the consistency level dynamically. These optimizations are complementary to the ones we develop in this paper.

3. Geo-Replicated Cloud Storage

We start by providing an overview of the coherence protocols used for strongly-consistent geo-replication in Apache ZooKeeper [5]. Our goal is to understand how WAN communication patterns between sites can affect the delay for read and write operations for the clients.

ZooKeeper is a *coordination kernel*, storing key/value pairs named *znodes* and organized in a hierarchy. Reading and writing znodes allows implementing a variety of synchronization patterns, such as locks or barriers. ZooKeeper is also often used for metadata management. It provides strong consistency guarantees for shared data. Write operations are linearizable. The order in which they are seen by clients connected to different sites is unique, and this order respects the real-time ordering of operations. Read operations are sequentially consistent. Reads see updates in the unique order, but may return values "from the past" with respect to the real-time ordering.

We present in Figure 1 the network exchanges for a read and a write to a three-site ZooKeeper. Clients are connected to their *local site*. ZooKeeper elects a *leader* server, which is

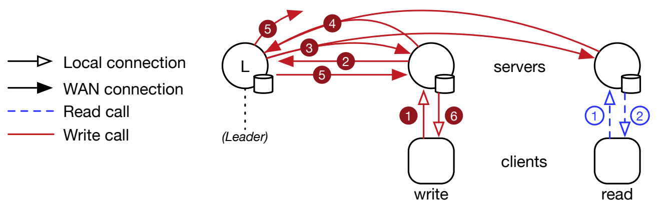

Figure 1: Network flows affecting service delay for read and write operations to a multi-site ZooKeeper installation.

in charge of deciding on the ordering of all write operations. To guarantee linearizability, the ordering of a write operation must be guaranteed between the call from the client ❶ and its return ❻. The local site forwards the write call to the leader ❷. The leader orders the update and sends it through FIFO channels to all sites ❸. Sites do not apply it yet, but acknowledge the reception to the leader ❹. After receiving all acknowledgements, the leader sends a commit message to all sites ❺. All sites apply the update, and the local site can return from the client call ❻. As a result, the service of a write operation involves up to 4 communications between sites (❷ to ❺) (2 when the local site is also the leader). When these rounds are parallel (to or from multiple sites) but must complete before the next round, the experienced delay is that of the slowest link. This applies to steps ❸ and ❹ but not to step ❺. Read operations are simpler as they only require a roundtrip ① and ② from the client to its local site. This complies with sequential consistency as the local site applies updates in the unique order decided by the leader, but may not have received the latest commits (in terms of real-time ordering) before replying to the client.

4. Multi-Site Testbed and Networking

We present in this section our testbed, its network configuration, and how we monitor the performance of WAN links from the edge. We build a 4-site testbed spanning four European countries and represented in Figure 2. Each site hosts virtual machines supporting instances of storage servers and benchmarking clients. An Open vSwitch router at the edge of the network at each site controls inter-site routing. A SDN controller at the CLU site orchestrates these virtualized routers. The testbed is managed using a single OpenStack installation spanning all four sites, with a controller also at the CLU site. All machines feature 2 virtual CPU cores at 2.5 GHz, 2 GB of RAM and run Ubuntu 14.04 Trusty using the libvirt libraries.

We keep a logical separation between the data and management networks. While the latter has a much smaller volume than the former, it should also have a higher priority to enable fast and accurate monitoring and control. All communication between compute nodes, or between the controllers and the compute nodes, is secured using virtual private networks using OpenVPN. We establish *tunnels* between compute nodes, for security and to allow a stable logical topology independent of the physical WAN links, so that the overlay between storage nodes appear as part of the same (virtual) LAN. The use of tunnels is instrumental in allowing dynamic re-routing of application flows, without interrupting these flows. This allows transparency for applications and cloud services using the tunnels. The routing change actually happens at the Open vSwitch router tunnel connection point, allowing to be controlled by the SDN controller, either manually as we do in this paper or automatically as we intend to do in future work.

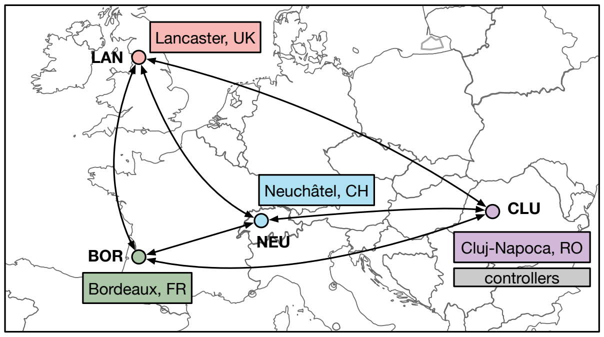

Figure 2: Physical locations of the four sites.

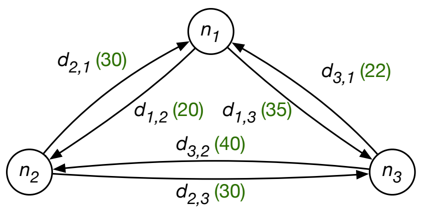

Figure 3: Three-node setup with OWDs (*ms*) in parentheses.

Network measurements. Deciding on the use of indirect routing requires accurate and timely information about the network performance. We are interested in two metrics, the Available Transfer Rate (ATR) and the One-Way Delay (OWD) for each path linking sites in our infrastructure. Based on these measurements, we can determine if a Triangle Inequality Violation (TIV) exists for one of the metrics, justifying the use of indirect routing.

The ATR of a path is the minimum of the difference between the path capacity and the path flow, calculated for all its links. To account for asymmetries in capacities and usage, the ATR is measured separately for each direction of the path. We use our own tool ATRAM [9] to measure ATRs. The tool is based on a combination of active probing and Kalman filtering. We calibrate it using Yaz [19].

Using echo-based measurements (e.g., ICMP and ping) only provides the round-trip time (RTT), and does not allow distinguishing between the two directions of a path. Similarly as for ATRs, we are interested in taking into account asymmetries in path delays. To obtain the one-way delays (OWD), we implement a cyclic-path delay active measurement scheme, based on the model in [20]. We illustrate it using Figure 3, using a simple 3-node setup. This model expresses the cyclic-path delays in terms of one-way delay variables d_{ij}. The goal is to find the maximum number of independent measurements from a source node, n_1 in our example. Let $N = 3$ be the number of nodes and $E = 6$ be the number of directional links. There are $N(N-1) = 6$ independent variables, for which we can measure up to $E - (N-1) = 4$ equations, each giving the sum of OWD over two or more or the E links, and collected from n_1. In our example, we may obtain:

$$d_{1,2} + d_{2,1} = 50 \text{ ms}, \qquad d_{2,3} + d_{3,2} = 70 \text{ ms},$$
$$d_{3,1} + d_{1,3} = 57 \text{ ms}, \quad d_{1,2} + d_{2,3} + d_{3,1} = 72 \text{ ms}.$$

This equation system is underdetermined and cannot be used directly to derive the individual values of OWDs d_{ij}. An estimation of these values is calculated by determining the space of possible values for each d_{ij}, based on the constraint that $\forall i, \forall j, d_{ij} \geq 0$ and constraints derived from the collected equations. These yield ranges of possible values for each d_{ij}. A configurable (in our case, 10) number of sample values are selected for all d_{ij} in their respective ranges that satisfy the constraints. The final estimation for the value of each d_{ij} is taken as the average of all valid solutions. More details about the process can be found in [21]. Resulting OWDs for our example are given directly on Figure 3.

The collection of cyclic-path measurements from the source node works as follows. Our mechanism is designed for overlay networks where all nodes are members of a multicast group. The source node injects *probes* into the network by flooding to this group, and collects responses containing cumulative latencies for paths in the topology. The novelty of our solution is that for the first copy of any probe packet that reaches a new node, we break the flooding rule of not sending back information to the node that issued the packet. Instead, the node returns the packet to the source node, allowing to measure a cyclic-path delay by subtracting the departure time from the arrival time. Note that this only happens for the first reception of the packet: subsequent receptions are transmitted further, allowing to measure other cyclic paths (e.g., $n_1 \rightarrow n_2 \rightarrow n_3$ in Figure 3). The process is scalable as neighboring nodes (members of the multicast group) will act at their turn as the origin of new probes and will replicate the mechanism that we previously described (e.g., allowing to measure $n_2 \rightarrow n_3 \rightarrow n_2$). This process is cyclic and runs every second. To stop a cycle, cancel the current flood and start a new measurement, the source node instructs members of the group to switch to a different listening UDP port. Packets for the previous port are simply discarded. If the network is too large to use a single source node, it is possible to split the domain into sub-domains, each with its own source node and multicast group.

5. Evaluation

We start by evaluating the base performance of our testbed network. We consider our testbed with no deployed application. We collect individual measurements of one-way delays (OWD) and available transfer rates (ATR) every second, for all 12 links of our 4-node testbed and for a duration of 4 minutes. Figure 4 presents the distribution of ATR measurements and Figure 5.(left) presents the distribution of OWD measurements.[2] In the both cases, and for the rest of our evaluation, each pair of sites yields two distributions: the two sites are given by the abscissa and the arrows represent the direction of the link that is measured (e.g., the first distribution is from `BOR` to `CLU` and the second is from `CLU` to `BOR`). The median ATR is heterogeneous and ranges from 2.4 to 18 Mbits/s. We also observe that links are typically asymmetric, with ATRs varying significantly for the both directions of a link, e.g. for `BOR`$\rightarrow$`NEU`. Delays are more symmetric, but we observe a factor of ∼3 between the fastest and the slowest links.

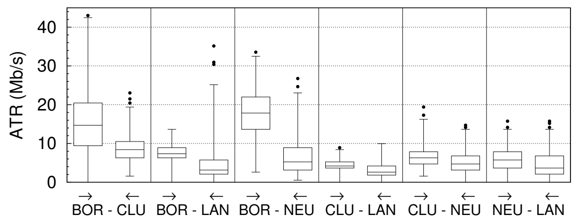

Figure 4: Baseline network performance: Available Transfer Rates (ATR) measured with no application deployed.

Triangle Equality Violation. As discussed in the introduction, a TIV is identified when there exists a triplet of sites A, B and C, for which the routing between A and B yields a lower performance than the routing between A and C, and then from C to B. A TIV may exist for either (or both) of the ATR and OWD metrics. For the ATR, the performance is defined as a minimum along the paths: a TIV exists if the ATR from A to B is lower than both ATRs from A to C and from C to B. For the OWD, it is defined as the cumulative delay over the paths: a TIV exists if the delay from A to B is higher than the sum of the delays from A to C and from C to B. We can see that when considering the medians of measurement distributions in Figure 4 and Figure 5.(left) our testbed does not initially feature a TIV for either of the metrics. However, when traffic happens in the testbed, a TIV can be observed clearly. We concentrate in the following of this evaluation on a TIV for the OWD metric. Indeed, our target application, ZooKeeper, is latency-sensitive: the delay seen by clients for read and writes applications is the key metric for characterizing its performance [5]. We identify an example OWD TIV when the path between `CLU` and `LAN` is subject to traffic. We consider a traffic goal of 98% of the median ATR (Figure 4) for both the `CLU`$\rightarrow$`LAN` and `LAN`$\rightarrow$`CLU` links. We use the iPerf tool to generate this target traffic. Figure 5.(middle) presents the distribution of measured OWD in this context. We can clearly see that the median delay for the `CLU`$\rightarrow$`LAN` link, about 65 ms, is higher than the sum of the `CLU`$\rightarrow$`NEU` median delay (∼23 ms) and `NEU`$\rightarrow$`LAN` median delay (∼16 ms). We could identify other examples of OWD TIV but will concentrate on this `CLU-NEU-LAN` TIV for the remainder of this evaluation.

2. We use box-plots with whiskers for representing distributions: The middle bar is the median, boundaries of the box are the first and third quartiles. Lines down and up from the box indicate the *span* covered by 99% of values from the distribution. Individual points show outliers.

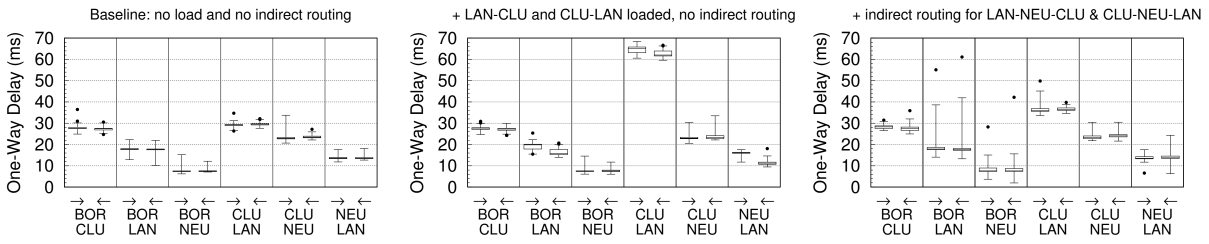

Figure 5: Distribution of OWD in different configurations, without and with the use of indirect routing.

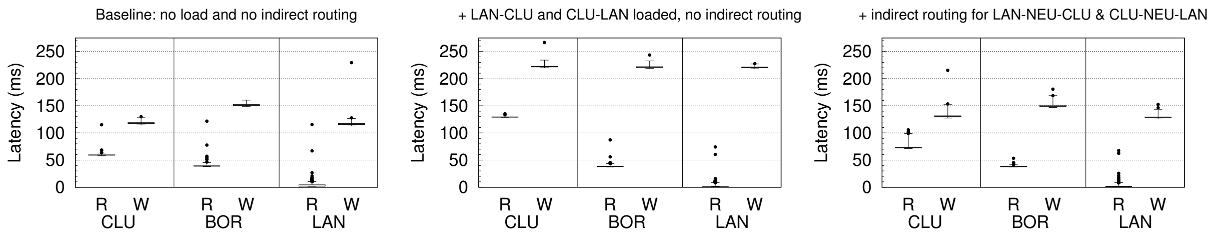

Figure 6: ZooKeeper performance of read (R) and write (W) operations from client in LAN connecting to the three sites.

Impact of indirect routing. Using the SDN controller, we install routing rules at the Open vSwitch routers of the three sites of the CLU–NEU–LAN triangle, setting up the indirect routing of the CLU-LAN tunnels for application traffic through NEU in the both directions. We maintain the load on the CLU→LAN and LAN→CLU links. This load does not use the tunnels and is therefore not redirected via NEU. The resulting OWD distributions are shown in Figure 5.(right). We can clearly observe that the delay for the CLU→LAN and LAN→CLU links reaches a median delay of about 36 ms in the both directions, more than the direct unloaded routes but significantly lower than the loaded ones.

Performance of geo-replicated storage. To guarantee availability in the presence of $f = 1$ site fault, ZooKeeper requires servers on $2f + 1 = 3$ sites. We deploy them at CLU, BOR and LAN. We set the CLU site to host the leader server. We use a client running the ZooKeeper benchmark to measure the delays of synchronous reads and writes to ZooKeeper. This client is deployed on the LAN site and we evaluate its performance when it connects to its local server in LAN but also when it connects to servers in BOR and CLU. The benchmark reads and writes znodes of size 1 KB.

We consider first a static use of indirect routing, where the benchmark is stopped and relaunched between experiments, in order to collect distributions of delays in the same configuration. Figure 6 presents the performance of read and writes to ZooKeeper from the client in LAN. The site given as the abscissa is the connection point for the client (CLU, BOR or locally to LAN). The three plots correspond to the three configurations used for Figure 5: (a) the baseline testbed with no traffic other than that of ZooKeeper itself; (b) the same testbed with traffic injection at 98% of the median measured ATR for both directions of the LAN→CLU link and (c) same as before but with the indirect routing via NEU for the LAN→CLU application traffic in the both directions.

We can observe that, as expected, the performance of read operations in the baseline configuration (a) is much better than that of write operations that involve multiple rounds of inter-site communications. In particular, connections from the client to the local site are very fast. Connections from the client to a distant site correspond roughly to the sum of the OWD to and from this distant site, e.g., a read from the client in LAN to the server in CLU is ∼60 ms, very close to the sum of the OWD for LAN→CLU and CLU→LAN in Figure 5.(left). The performance of writes is also consistent: a write to the local site LAN yields a delay involves the following steps (see Figure 1): ❶ LAN→LAN (negligible), ❷ LAN→CLU (leader), ❸ CLU→LAN and ❹ LAN→CLU (longest delay for send/ack steps), ❺ CLU→LAN, ❻ LAN→LAN (negligible). The sum of median OWDs, about 120 ms, is consistent with the median performance seen by the application.

When subject to traffic on the LAN→CLU and CLU→LAN links, we clearly see a degradation of performance for reads made at the distant site CLU from the client at LAN, but more importantly, a general degradation of performance for writes made at any of the sites. Indeed, the higher delays of the LAN→CLU and CLU→LAN links is reflected on the time taken by the coherency protocol to inform and get an acknowledgment of the ordering of the writes from the leader server in CLU (steps ❸ and ❹).

Figure 6.(right) presents the performance of ZooKeeper when using indirect routing. We can clearly observe that this indirect routing has a very positive impact: by mitigating the high-delay problem of the LAN→CLU and CLU→LAN links, it allows the performance of the read operations to CLU and of the write operations to all sites to reach a level of performance that is very close to that of the un-loaded baseline configuration.

These previous results allow us to positively answer the questions asked in our introduction: (1) WAN routing strongly affects the performance of geo-replicated storage; (2) network performance indicators and in particular our OWD measurement tool allow precisely estimating the performance of the application when combined with the knowledge of

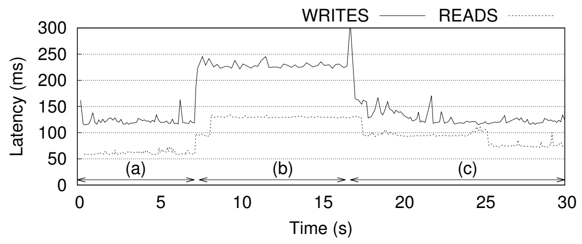

Figure 7: Dynamic adaptation impact on operations delays.

its interaction patterns, and using indirect routing allows mitigating the impact of performance variations. We still need to evaluate the ability of our setup to enable indirect routing with no application interruption, and evaluating the impact of such a dynamic change on its performance. Figure 7 presents this evaluation. It represents individual measurements of alternated read and write calls from the client in `LAN` connected to the master in `CLU`. Steps (a)-(c) correspond to the ones described previously. We can clearly see the decrease in performance when reaching step (b), and that the implementation of indirect routing for the phase (c) is fast. We observe a short spike of delay when enabling indirect routing, but this spike (300 ms) remains within reasonable boundaries. This positively answers our final question and shows the practicality of enabling dynamic adaptation with no application downtime or performance degradation.

6. Conclusion

We have shown that indirect routing can support dynamic network-level adaptation in a multi-site testbed supporting a demanding and latency-sensitive application, geo-replicated ZooKeeper. Our work opens several interesting perspectives. First, we are interested in studying the long-term performance variations of the WAN testbed we have built and evaluate if there exists patterns and trends for the performance that can be observed at the edge of the network. Second, we wish to automatize the decision process for using indirect routing. This could happen at the level of the SDN controller itself, or through the cooperation of the SDN controller and virtualized network functions (VNF) as described in SDNFV [22]. We envision the construction of performance and traffic models for storage applications and coherence protocols, allowing to make informed decisions during the network optimization process. These models, together with performance measurements, could allow deciding on indirect routing in micro-clouds environments based on application needs for reactive data storage and management.

Acknowledgments. We are grateful to Yehia Elkhatib, Inti Gonzales-Herrera and Laurent Réveillère for their help in setting up our testbed. The research leading to these results has received funding from CHIST-ERA under project DIONASYS, from the Swiss National Science Foundation (SNSF) under grant 155249, and from the URBIVEL project under identifier P_40_333.

References

[1] S. Jain *et al.*, “B4: Experience with a globally-deployed software defined WAN,” in *ACM SIGCOMM Conference*, 2013.

[2] I. Cuadrado-Cordero, F. Cuadrado, C. Phillips, A.-C. Orgerie, and C. Morin, “Microcities: A platform based on microclouds for neighborhood services,” in *16th International Conference on Algorithms and Architectures for Parallel Processing*, ser. ICA3PP, 2016.

[3] C. Lumezanu, R. Baden, N. Spring, and B. Bhattacharjee, “Triangle inequality variations in the internet,” in *9th ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC, 2009.

[4] J. M. Opos, S. Ramabhadran, A. Terry, J. Pasquale, A. C. Snoeren, and A. Vahdat, “A performance analysis of indirect routing,” in *IEEE Intl. Parallel and Distributed Processing Symp.*, ser. IPDPS, 2007.

[5] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, “Zookeeper: Wait-free coordination for internet-scale systems,” in *USENIX Annual Technical Conference*, ser. ATC, 2010.

[6] M. V. Ulinic, A. B. Rus, and V. Dobrota, “Openflow-based implementation of a gearbox-like routing algorithm selection in runtime,” *Acta Technica Napocensis*, vol. 55, no. 2, pp. 23–32, 2014.

[7] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, “Best-path vs. multi-path overlay routing,” in *3rd ACM SIGCOMM Conference on Internet Measurement*, ser. IMC, 2003.

[8] G. Wang, B. Zhang, and T. S. E. Ng, “Towards network triangle inequality violation aware distributed systems,” in *7th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC, 2007.

[9] I. A. Ivanciu, A. B. Rus, V. Dobrota, and J. Domingo-Pascual, “Active measurement of the available transfer rate used in an algorithm for generalized assignment problem,” in *11th International Symposium on Electronics and Telecommunications*, ser. ISETC, 2014.

[10] R. Jain and S. Paul, “Network virtualization and software defined networking for cloud computing: a survey,” *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, Nov 2013.

[11] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan, and T. Wood, “Virtual function placement and traffic steering in flexible and dynamic software defined networks,” in *21st IEEE International Workshop on Local and Metropolitan Area Networks*, ser. LANMAN, April 2015.

[12] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, “Toward a software-based network: integrating software defined networking and network function virtualization,” *IEEE Network*, vol. 29, no. 3, pp. 36–41, May 2015.

[13] F. Lemma, J. Schad, and C. Fetzer, “Dynamic replication technique for micro-clouds based distributed storage system,” in *2013 International Conference on Cloud and Green Computing*, ser. CGC, 2013.

[14] I. Narayanan, A. Kansal, A. Sivasubramaniam, B. Urgaonkar, and S. Govindan, “Towards a leaner geo-distributed cloud infrastructure,” in *6th USENIX Workshop on Hot Topics in Cloud Computing*, ser. HotCloud, 2014.

[15] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, “Volley: Automated data placement for geo-distributed cloud services,” in *7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI, 2010.

[16] Y. Lin, B. Kemme, M. Patino-Martinez, and R. Jimenez-Peris, “Enhancing edge computing with database replication,” in *26th IEEE Intl. Symp. on Reliable Distributed Systems*, ser. SRDS, 2007.

[17] M. S. Ardekani and D. B. Terry, “A self-configurable geo-replicated cloud storage system,” in *11th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI, 2014.

[18] P. N. Shankaranarayanan, A. Sivakumar, S. Rao, and M. Tawarmalani, “Performance sensitive replication in geo-distributed cloud datastores,” in *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, ser. DSN, 2014.

[19] J. Sommers, P. Barford, and W. Willinger, “A proposed framework for calibration of available bandwidth estimation tools,” in *11th IEEE Symposium on Computers and Communications*, ser. ISCC, 2006.

[20] O. Gurewitz and M. Sidi, “Estimating one-way delays from cyclic-path delay measurements,” in *20th Annual Joint Conference of the IEEE Computer and Communications Societies*, ser. INFOCOM, 2001.

[21] A. Taut, I.-A. Ivanciu, E. Luchian, and V. Dobrota, “Active measurement of the latency in cloud-based networks,” *ACTA TECHNICA NAPOCENSIS, Electronics and Telecommunications*, vol. 58, no. 1, 2017.

[22] W. Zhang, G. Liu, A. Mohammadkhan, J. Hwang, K. K. Ramakrishnan, and T. Wood, “SDNFV: Flexible and dynamic software defined control of an application- and flow-aware data plane,” in *17th International Middleware Conference*, ser. Middleware, 2016.