Improving the Network Performance of a Container-based Cloud Environment for Hadoop Systems

Cassiano Rista
Pontifical University Catholic of Rio Grande do Sul (PUCRS), GMAP Research Group, Porto Alegre – RS – Brazil
Email: luis.rista@acad.pucrs.br

Dalvan Griebler, Carlos A. F. Maron
Três de Maio Faculty (SETREM), LARCC Research Group,
Pontifical University Catholic of Rio Grande do Sul (PUCRS), GMAP Research Group, Porto Alegre – RS – Brazil
Email: {dalvan.griebler, carlos.maron}@acad.pucrs.br

Luiz Gustavo Fernandes
Pontifical University Catholic of Rio Grande do Sul (PUCRS), GMAP Research Group, Porto Alegre – RS – Brazil
Email: luiz.fernandes@acad.pucrs.br

***Abstract*—Cloud computing has emerged as an important paradigm to improve resource utilization, efficiency, flexibility, and the pay-per-use billing structure. However, cloud platforms cause performance degradations due to their virtualization layer and may not be appropriate for the requirements of high-performance applications, such as big data. This paper tackles the problem of improving network performance in container-based cloud instances to create a viable alternative to run network intensive Hadoop applications. Our approach consists of deploying link aggregation via the IEEE 802.3ad standard to increase the available bandwidth and using LXC (Linux Container) cloud instances to create a Hadoop cluster. In order to evaluate the efficiency of our approach and the overhead added by the container-based cloud environment, we ran a set of experiments to measure throughput, latency, bandwidth utilization, and completion times. The results prove that our approach adds minimal overhead in cloud environment as well as increases throughput and reduces latency. Moreover, our approach demonstrates a suitable alternative for running Hadoop applications, reducing completion times up to 33.73%.**

***Index Terms*—Cloud Computing; Network Performance; Big Data; Link Aggregation; Container-Based Cloud.**

I. Introduction

Big data is still considered one of the greatest computational challenges and has been intensively studied over recent years. This challenge becomes even greater due to the large amount of data generated daily by different corporations, web-based services and systems, social media, and others. According to White [1], the term big data refers to large amounts of data that usually transcend the ability of software tools to collect, manage, and process data in acceptable polynomial time.

The Hadoop framework enables distributed processing on computer clusters using simple programming models. It was designed to work with a single server or set of a thousand machines, offering local processing and storage. The framework itself is designed to detect and address application layer failures to provide high availability services, using Commercial Off-The-Shelf (COTS) hardware.

Cloud computing infrastructures support rapid resource provisioning, which in turn is a suitable option to deploy a Hadoop environment. Primarily due to the advances of cloud platforms, there are possibilities to develop new business models based on the pay-per-use billing structure. Also, there are cloud computing technologies (specifically those focused on network and storage) that provide lower performance degradation [2], [3], [4], [5]. However, the use of cloud environments for processing big data applications has traditionally been avoided when it requires high bandwidth and throughput as well as low latency [6], [7]. Traditional cloud providers, which use virtualization technologies that are not optimized for the execution of big data applications, add significant overheads [8]. Our main challenge in this paper is to improve network performance in the container-based cloud environment for Hadoop applications.

We chose to increase the network bandwidth by using the IEEE 802.3ad link aggregation standard [9]. It defines a standard method to combine multiple physical links that can be used as a single logical link. The standard is a layer 2 control protocol that can be used to automatically detect, configure, and manage a single logical link with multiple physical links between two adjacent enabled devices. Thus, link aggregation provides higher availability and capacity while network performance improvements are obtained using existing hardware (IEEE 802.3ad requires support in a network switch).

Also, our goal is to target a high-performance cloud environment by deploying container-based instances, where Hadoop applications may run. We chose container-based technologies to enable multiple isolated Linux systems to run on a single host through Namespaces (providing isolated user environments in the form of containers) and cgroups (providing resource management and accounting). LXC (Linux Container) [10] technology is an important part of this cloud infrastructure, because it is a free software that provides a

978-1-5386-3250-5/17 $31.00 © 2017 IEEE
DOI 10.1109/HPCS.2017.97

powerful set of userspace tools and utilities to manage Linux containers. It combines an easy-to-use interface with easy-to-construct image files. Therefore, it is a piece of software in a complete file system that contains everything needed to run code, runtime, system tools, and system libraries.

Figure 1 presents the container technology architecture. Technically, it is a lightweight alternative to a hypervisor that runs at the operating system level, providing abstractions directly for guest applications. For this reason, all containers share a single operating system kernel and they are supposed to have weaker isolation compared to hypervisor-based systems. However, from the customer's point of view, each container executes exactly the same as a stand-alone operating system.

Fig. 1. High-level representation of container architecture.

This paper aims at provide a suitable approach to deploy dynamic link configuration and network link aggregation by using IEEE 802.3ad. Also, our goal is to evaluate how real-world Hadoop applications perform in such a container-based cloud environment to demonstrate applicability, feasibility, and efficiency. Therefore, we are making the following contributions:

- A deployment approach that increases the network bandwidth in container-based cloud platforms. Our approach can be deployed without attaching extra hardware, since it can be simply deployed when there is more than one network board available on the data-center server machines. We do so that throughput and latency are improved in container-based cloud instances.
- A solution to reduce the completion time for big data applications in Hadoop Systems. Although a specific Hadoop application has been used, we demonstrated that our deployment approach reduces the execution time significantly for a network intensive workload.
- A set of experiments demonstrating the feasibility and efficiency of our approach. Our set of experiments are representative to demonstrate the feasibility for real-world cloud data-center scenarios. Also, network intensive workloads will efficiently perform the applications, concerning latency, throughput, and execution time.

This paper is organized as follows: Section II discusses related research contributions. Section III describes the proposed approach in detail, focusing on the network enhancement strategy for big data applications. Section IV presents the experiments and performance analysis. Finally, Section V concludes the paper and presents ideas for future work.

II. Related Work

This section describes the primary related works aimed at improving the network in big data applications.

For instance, Yazdanov et al. [11] propose a network-sensitive I/O scheduler called EHadoop for an elastic MapReduce cluster. Their study observed that during load peaks more computing nodes are added to maintain the performance of the cluster. However, the throughput does not improve because of network saturation (bottlenecks). Thus, EHadoop performs task scheduling based on the available bandwidth. The evaluation results show that EHadoop avoids network contention but does not increase the completion time of the MapReduce tasks and even causes network bandwidth degradation.

The approach in [12] shows a distributed control method to allocate computing resources to distributed parallel software components according to a model-driven predictive approach. The approach supposes a static deployment of the components, and it is only throughput-oriented (latency is not addressed).

Renner et al. [13] presented an approach based on containers that takes into account the topology of the network in order to avoid congestion. Containers were arranged to be close to data entries to improve data locality and reduce the need to read remote disks in a distributed file system. The study also presents the development of a prototype for effectively allocating these containers and analysis of scalable data in a shared cluster with hierarchical networks. The prototype was implemented in Hadoop Yarn and evaluated with workloads consisting of different applications and data sets using Apache Flink. The evaluation showed a reduction of up to 67% in the execution times of jobs with intensive network workloads.

Furthermore, an algorithm for bandwidth scheduling with SDN (Software-Defined Network) for Hadoop called BASS (Bandwidth-Aware Scheduling with SDN in Hadoop) was proposed by Qin et al. [14]. The Hadoop scheduler assigns tasks based on the location of data. BASS takes into account the bandwidth of the links (at a given instant) to assign tasks. This feature allows the scheduler to move data from one node to another when necessary for better scheduling.

In addition to minimizing end-to-end communication latency by using local data, Prabhavat et al. [15] proposed a load distribution model, called E-DCLD (Effective Delay Controlled Load Distribution). It can reduce packet latency variation, thereby minimizing the risk of packet reordering without incurring additional network overhead. When fewer packets must be reordered, recovery time is decreased. In other words, the model both reduces the end-to-end communication latency as well as recovery time for packet reordering. Performance was verified by comparing E-DCLD with existing models through analysis and experiments simulating different traffic conditions.

A comparative study between applications with different communication needs and complexity, was carried out by Ekanayake et al. [16]. This study concluded that latency-sensitive applications have higher performance degradation than bandwidth-sensitive applications. Moreover, this study also took into account different implementations of MapReduce, presenting a performance analysis of high performance parallel applications in virtualized environments.

Desai et al. [17] present an architecture called ACDPA

(Advanced Control Distributed Processing Architecture) that is designed to efficiently process and control network traffic. This study considered the use of Hadoop for distributed processing and OpenDaylight as the SDN controller. The basic principle consists of collecting network traffic from the SDN data plane and classifying the entire Hadoop flow. This information is then used as feedback for the SDN controller to define the quality of service (QoS) requirements for each flow category. In general, the information collected is used to determine characteristics and define the priority of flows based on QoS.

A model was presented by Wang et al. [18] to merge GPU computing with the MPTCP (Multipath TCP) protocol to improve the performance of Hadoop distributed computing. In order to do so, GPU computing was used to accelerate the mapping phase, while MPTCP was used to reduce the data transfer time during the reduction phase. The experiments show that the model can accelerate the performance of Hadoop applications with robustness and aggregation of bandwidth, as well as reduce latency for distributed computing.

Radhakrishnan et al. [2] proposed V-Hadoop (Virtualized Hadoop Using Containers), a framework that leverages Linux containers to run Hadoop jobs efficiently without requiring large physical machine clusters. Users can manage a cluster across multiple physical machines and perform simple resource-based scheduling of Linux containers by adding or removing containers dynamically based on resource usage and availability. Also, their experiments demonstrated that V-Hadoop is comparable to a fully-distributed Hadoop cluster.

The previous related works presented different approaches aimed to enhance performance and data location in the network. Some strategies are focused on task scheduling based on the available bandwidth [11] [14] while others tried to reduce communication latency [15] [16]. Some were concerned with providing high-performance for big data applications [2] and others presented solutions to define the data flows based on quality of service (QoS) [17] and select the most appropriate network topology to avoid traffic jams. Finally, [18] addressed mixed GPU hardware with the MPTCP (Multipath TCP) protocol to accelerate big data applications.

Note that the literature does not present any work focused on demonstrating the advantages of link aggregation for network bandwidth, latency, throughput, or execution time in big data applications as our work does. In fact, we have also targeted the container-based cloud computing environment, which presents lower abstraction overhead and performance degradation. In contrast to previous works, our research studies and proposes a low-level network performance-aware deployment model for Hadoop applications in the cloud scenario. Our approach is evaluated through a set of experiments to demonstrate its benefits and overheads, using a consolidated network benchmark and Hadoop application.

III. Improving Hadoop Performance

This section presents the proposed approach enhancing Hadoop's distributed computing performance in a cloud environment. Our approach takes advantage of technologies such as IEEE 802.3ad and LXC. First, we discuss how the LXC strategy is applied to Hadoop. Second, we present an overview of the architecture. Finally, the main characteristics and benefits of MapReduce IEEE 802.3ad stage are described.

A. Hadoop on LXC Cloud Instances

One of the main advantages of integrating Hadoop into a cloud computing environment is that it provides the opportunity to manage the trade-offs between scale-up and scale-out. For instance, in the bare-metal scenario, the size of each cluster node is defined by the available hardware. Consequently, the system administrator has to adjust the application to fit this size. On the other hand, when system administrators are in a cloud scenario, they may reconfigure the amount of resources based on the needs of the application. This feature allows the administrator to optimize resource usage while delivering better completion times.

In addition to the advantages of the cloud computing environment, we are also interested in reducing performance degradation, which is commonly caused by using virtualization technologies. Due to this, we chose to use LXC in our proposed model and also deployed the Hadoop cluster environment. According to Rizki et al. [19], container-based instances provide almost the same performance as the native environment (bare-metal). Moreover, using LXC allowed us to still take advantage of elasticity, multi-tenancy, and resource management.

The main goal of our container-based strategy is to allow Hadoop system administrators to create clusters and take advantage of the cloud computing environment while offering the best performance.

B. Introducing IEEE 802.3ad on Hadoop Architecture

Hadoop was originally created for MapReduce distributed processing on Distributed File System (DFS) with a dedicated cluster of servers [1]. In such a cluster, each server is used as node. The master node coordinates the daemon processes such as Job Tracker and Name Node (see Figure [1]). On the other side of the Hadoop cluster, the slave nodes store data blocks where the Task Tracker and Data Node eventually compute the data.

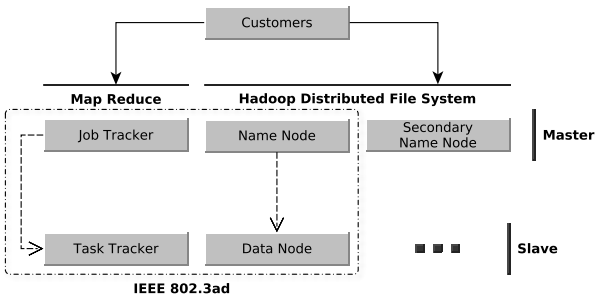

Fig. 2. High-level representation of IEEE 802.3ad on Hadoop.

Figure 2 illustrates where the link aggregation via IEEE 802.3ad [9] is introduced to improve performance in the Hadoop architecture. Note that the Job Tracker supervises and coordinates the data parallel processing using the MapReduce pattern while the Name Node is responsible for coordinating the data storage (called HDFS - Hadoop Distributed File System) through the master node. Also, each slave node runs a Data Node daemon and Task Tracker which communicate and receive master node instructions. We can easily identify where link aggregation will be helpful to improve Hadoop application performance, especially for communication between master and slave nodes.

Therefore, the distributed computing mechanism of Hadoop (including HDFS and MapReduce) will take advantage of two physical network interfaces working in the bridge mode of each cluster node. This allows network packages to use multiple paths when establishing simultaneous connections. Our goal of improving bandwidth with IEEE 802.3ad is to reduce the transmission time and increase network throughput when delivering computing results between Map and Reduce stages, which will be explained in detail in the next section.

C. Hadoop MapReduce Taking Advantage of IEEE 802.3ad

This section discusses how big data applications implemented with the MapReduce framework (which runs on top of Hadoop architecture, previously highlighted in Section III-B) may take advantage of IEEE 802.3ad. Firstly, the basic concept of MapReduce is to process big data in parallel, performing Map and Reduce operations in stages. However, there is an intermediate stage that is called Shuffle. It collects mapped data from the Map operation to merge and deliver computations to the Reduce operation. Because there are many communications that take place between many processes and nodes in the Shuffle stage, a great deal of data is being transferred through the network. Figure 3 represents how communication occurs in high-level manner when using MapReduce on Hadoop.

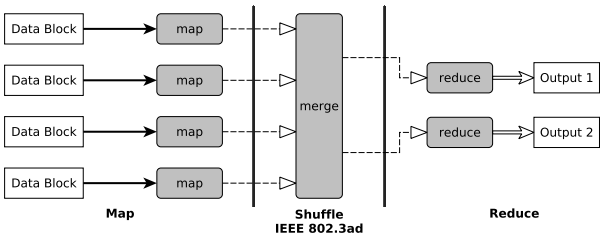

Fig. 3. High-level representation of MapReduce stages.

Once the location of the network intensive communication stage was identified, we proposed IEEE 802.3ad to aggregate physical network interface, since it allows us to increase network bandwidth. We hoped to reduce the transferring data time when performing the Shuffle stage as highlighted in Figure 3. Therefore, our hypothesis that taking advantage of IEEE 802.3ad should provide better network latency and throughput, allowing Hadoop MapReduce applications to achieve better completion times.

The IEEE 802.3ad enables dynamic link aggregation and dis-aggregation by exchanging the packets with the cluster nodes. In this case, the switch equipment dynamically groups similar ports into a single logical link, increasing the bandwidth and balancing the load for the Shuffle stage of MapReduce.

In addition, link aggregation increases availability for Hadoop cluster, providing limited degradation when failure occurs. It provides network redundancy by load-balancing traffic across all available links. If one of the links fails, the system automatically load-balances traffic across all remaining links. In the next section, we present the results of our experiments to show the efficiency of our approach.

IV. Experiments and Evaluations

This section presents experiments that were conducted to evaluate our proposed approach. Our main goal was to improve the network performance in a cloud environment for Hadoop distributed computing by using link aggregation and container- based cloud instances. First, we describe the design and methodology of our experiment. Second, we present the results of the network performance evaluation, which evaluated throughput and latency in different scenarios. Finally, we present the results in Hadoop, comparing network usage rates and execution times.

A. Experimental Setup and Methodology

The hardware setup consisted of two ProLiant DL385 G6 servers. Each server had two six-core AMD Opteron processor 2425 HE and 32GB of RAM. These servers ran the operating system Ubuntu Server 14.04 64-bit, Hadoop 2.7.3, Ethernet channel bonding driver 3.7.1, and LXC container 1.0.8 release. A Gigabit Ethernet Switch with IEEE 802.3ad connected the servers through two Gigabit Ethernet controllers on each server. All tests were run 15 times for each experiment and considering a confidence interval of 95%. The following experiments were supported by two benchmarks:

- Netpipe [20] is a protocol independent performance benchmark that visually represents the network performance under a variety of conditions. It uses a simple series of ping-pong tests over a range of message sizes to provide a complete measurement of network performance. Message sizes are chosen at regular intervals with slight differences to provide a complete evaluation of the communication system. Each data point includes many ping-pong tests to provide accurate timing. Latencies are calculated by dividing the round trip time in half for small messages (less than 64 Bytes).
- TestDFSIO [21] is a benchmark that performs intensive read and write computations on HDFS. It writes or reads many files in the HDFS, where there is one map task per file. The size and number of files are specified as command-line arguments. In the following experiments, 48 files of size 512 MB are specified. Thus, the main use

of TestDFSIO is for stressing HDFS system to discover performance bottlenecks (particularly in the NameNode and DataNodes) of the Hadoop cluster.

Based on the conceptual deployment model presented in Figure 4, the two scenarios used in our experiments are described as follows:

- **Scenario 1**: verifies throughput and latency as well as compares the overhead added by using container-based cloud instances. We used the NetPipe benchmark in two bare-metal configurations (Regular TCP and IEEE 802.3ad) and a cloud instance configuration with LXC and IEEE 802.3ad deployed. These settings are related to the experiments conducted in Section IV-B.
- **Scenario 2**: aims to evaluate the bandwidth utilization rate and execution time in big data applications. We therefore used the TestDFSIO benchmark, running on a container-based cloud with LXC and supporting both Regular TCP and IEEE 802.3ad configurations. Also, TestDFSIO was instantiated to run up to three concurrent instances in order to generate different loads and network stresses. These settings are related to the experiments in Section IV-C.

Unlike what is illustrated in Figure 4, the Regular TCP settings will only have a single communication channel between servers. Meanwhile, Figure 4 clearly demonstrates that IEEE 802.3ad is able to aggregate two communication channels, whereas for the LXC instance there is only a single network interface lxc-br0.

It is important to highlight that for the LXC instances there are no limitations based on CPU, memory, disk or network bandwidth. Moreover, when running the TestDFSIO benchmark in concurrently, our goal was to simulate a situation closer to the real world where multiple clients are simultaneously accessing the environment, thereby generating greater concurrency in the network.

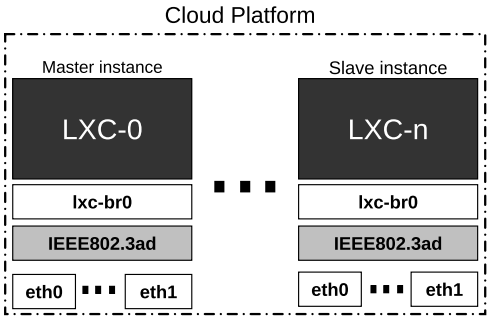

Fig. 4. The conceptual deployment model shows the configuration of processing nodes of the cloud, the communication links between them, and the component instances and objects that reside in them.

B. Evaluating the Network Performance

Our first scenario consists of evaluating network performance. Therefore, we performed these experiments using the NetPipe benchmark to measure latency and throughput, considering different deployment environments. Also, we compared network performance in bare-metal and LXC instances.

Figure 5 presents the results of the throughput measurement. The first environment is the Regular TCP which is not included in our approach and was run on bare-metal instances. Then, the results of IEEE 802.3ad were achieved on the bare-metal instances while IEEE 802.3ad-LXC were achieved on container-based cloud instances.

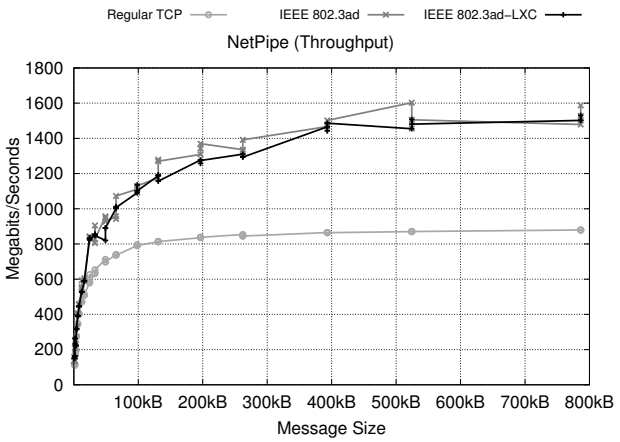

Fig. 5. Comparative evaluation of the throughput.

Note that IEEE 802.3ad (our approach) significantly increases the throughput compared to the two bare-metal versions in Figure 5. This improves even more when there are larger message sizes. The throughput of our approach running in bare-metal is not significantly different than running it in container-based instances. These results confirm that our approach does not add significant overhead in the cloud environment, but it is still able to increase the application's throughput.

The same environments in Figure 5 were used to evaluate the latency performance in Figure 6. We can observe that when not using our approach, the latency may be greater as the message size increases. The latency of our approach running in bare-metal is not significantly different than when running it in container-based instances.

With these results, we were able to demonstrate that link aggregation via IEEE 802.3ad increases the throughput and reduces the latency in the network application. Also, container-based instances add only very small overheads in the network performance when the link aggregation is introduced via IEEE 802.3ad. Consequently, our experiments indicate that our approach may provide good results for other network intensive applications. The next section will confirm whether this is actually true for big data applications.

C. Evaluating Hadoop's Performance

Our second scenario consists of evaluating a Hadoop MapReduce application, which is called TestDFSIO. As we confirmed no significant overhead by introducing our approach

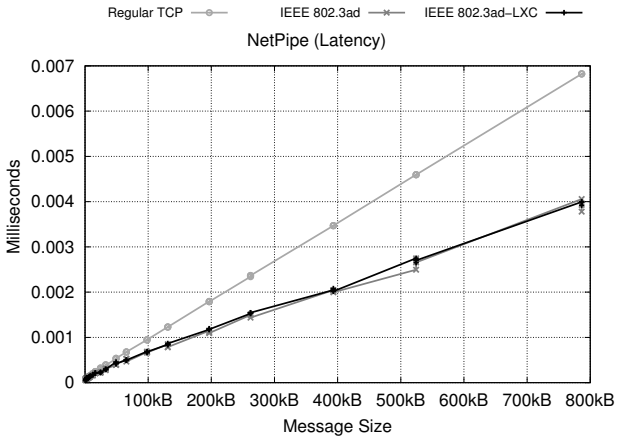

Fig. 6. Comparative evaluation of the latency.

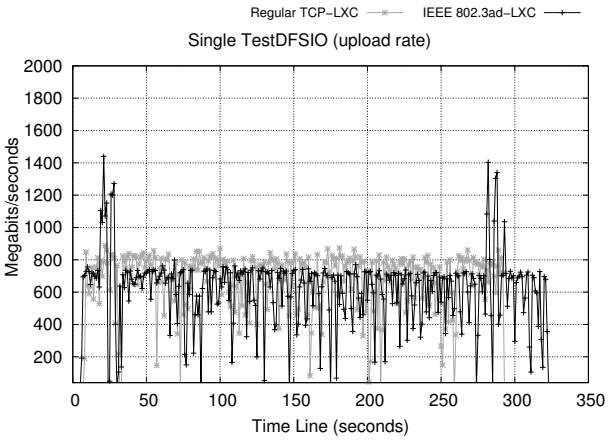

Fig. 7. Running single instance of TestDFSIO.

in container-based cloud instance, we deployed a Hadoop cluster on LXC cloud instances and only ran the tests there to compare our approach to Regular TCP (which does not perform link aggregation). However, we varied the number of concurrent TestDFSIO applications running during the test. We collected the upload and download rates monitoring the network interface (`lxc-br0` as can be seen in 4) of the master cloud instance with the "dstat" tool.

As in the master cloud instance, the greatest network traffic is during the upload, therefore we will only discuss upload results in Figures 7, 8, and 9. Further details regarding download rate results are discussed in Appendix. Figure 7 illustrates that running a single TestDFSIO execution will already demand the nominal capacity of 1 Gbps of a single physical network interface. In this case, the TestDFSIO demonstrated to have upload peaks when looking at the results of our approach, which nominally should have 2 Gbps capacity due to the link aggregation deployed.

These upload peaks are confirmed with the results presented in Figure 8, where they are more pronounced. Although there is enough bandwidth in our approach because of the link aggregation, in Figure 7 we can see that Regular TCP uses the network less when running a single TestDFSIO. This was directly reflected in the execution times, which are shown in Table I. Therefore, we observed an unexpected result because our approach should nominally have more network bandwidth available, but a higher completion time.

The overhead presented in our approach in Figure 7 occurs due to the administrative overhead on the network packets and the connection that our approach uses (IEEE 802.3ad). Consequently, it becomes more visible when the application does not requires higher network bandwidth than the physical network supports, as reported by other authors [22] [23].

In Figure 8, we can explicitly identify that two TestDFSIO running concurrently will saturate the bandwidth in the Regular TCP approach unlike our approach. As expected, the completion times are even better using our approach (see in Table I), because it provides more bandwidth.

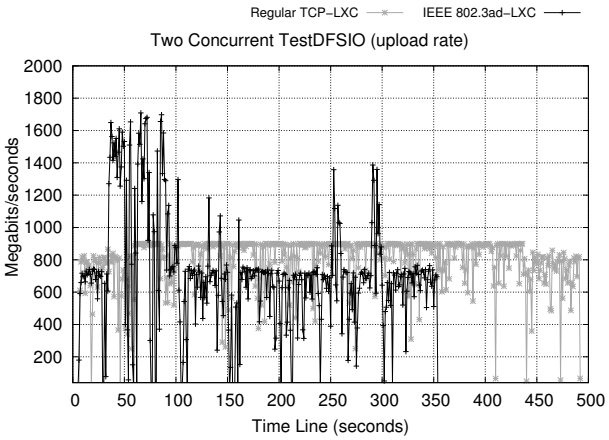

Fig. 8. Running two concurrent instances of TestDFSIO.

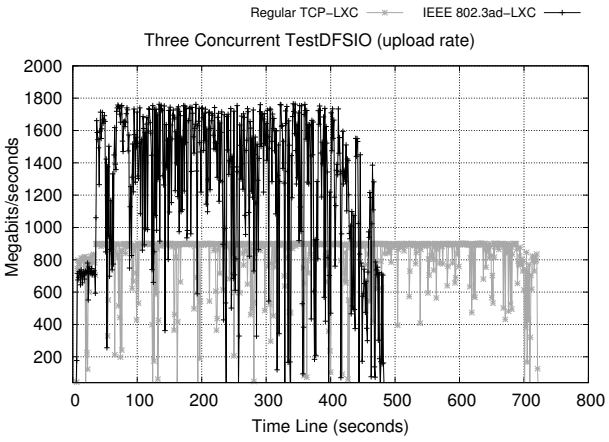

Fig. 9. Running three concurrent instances of TestDFSIO.

The results of running three concurrent TestDFSIO are presented in Figure 9. In this case, the upload rate significantly increases, reaching about 1.7 Gbps. As we have seen in the

results, these three TestDFSIO running concurrently demands high bandwidth that Regular TCP cannot sustain. Also, the completion time in our approach were reduced about 33% with respect to the ones in Regular TCP deployment. The absolute numbers of the completion times on all TestDFSIO environments are reported in Table I.

TABLE I
COMPLETION TIMES OF TESTDFSIO ON TCP REGULAR AND IEEE 802.3AD CLOUD INSTANCES.

	Number of Concurrent TestDFSIO		
Environment	**1** (seconds)	**2** (seconds)	**3** (seconds)
IEEE 802.3ad (LXC)	338	375	501
Regular TCP (LXC)	315	507	756

Finally, the results presented in this section provided us with fruitful insights, demonstrating when our approach will improve Hadoop's network performance. We can highlight that NetPipe benchmark results on our approach have shown good results concerning latency and throughput (in Section IV-B). However, the application will only take advantage of our approach when there is a need for higher network bandwidth than what is provided in the Regular TCP environment.

V. CONCLUSIONS

In this paper, we proposed a deployment approach to improve network performance in container-based cloud instances for Hadoop-based big data applications. This approach uses LXC and IEEE 802.3ad link aggregation solutions. The main goal was to increase the network bandwidth. Consequently, we achieved better throughput and latency as Section IV-B presented, and we reduced the completion time in Hadoop applications as discussed in Section IV-C. Therefore, we have concluded that other network intensive applications can take advantage of our approach.

Experiments were performed to evaluate traditional Regular TCP and then compare it with the our approach through a set of experiments to measure throughput, latency, bandwidth utilization, and completion times focusing on the use of big data applications. Our experiments indicate that in the best case (running three concurrent instances of TestDFSIO), completion times were reduced up to 33.73%. The results also showed a higher completion time in the worst case (running single instance of TestDFSIO). This occurred due to the administrative overhead on the network packets and the connection that our approach uses (IEEE 802.3ad).

Another important improvement is the use of a container-based environment. In this sense, we found that LXC has near-native network performance. Traditional hypervisors in turn present limitations in dynamic resources allocation, where a range of resizing is limited to that slice. Yet this limitation does not exist in container-based environments, where it is possible to expand particular containers to cover all the resources of the physical node.

In future work, we aim to investigate the possibility of increasing network bandwidth by using MultiPath TCP [24] instead of IEEE 802.3ad link aggregation and compare the performance with our current approach. Secondly, we intend to evaluate our approach in other cloud instance types (*e.g.*, KVM, VMWare, XenServer). Lastly, we hope to provide a framework that can be easily integrated with cloud platforms (*e.g.*, OpenStack and OpenNebula) to enhance network bandwidth in an elastic way.

ACKNOWLEDGMENT

The authors would like to thank CAPES, FAPERGS, PUCRS, FACIN, SETREM, LARCC, and HiPerfCloud project for their partial financial support.

APPENDIX

In this section, we present additional results regarding our experiments for the Hadoop environment (Scenario-2). We highlight the download rates for Regular TCP and our approach implemented with IEEE 802.3ad. The samples were taken from the master node of the deployed Hadoop cluster. In Figure 10, we can observe that the Regular TCP environment was faster than our approach when the TestDFSIO workload did not stress the network bandwidth limits. In this case, there was a single TestDFSIO application running. We did not expect these results. However, we suspect that this occurred due to the combination of the Hadoop scheduler and overhead caused by the package management in the IEEE 802.3ad, which is not ideal when there is weak network traffic. Note that this was also highlighted in Figure 7, where upload rates were similar.

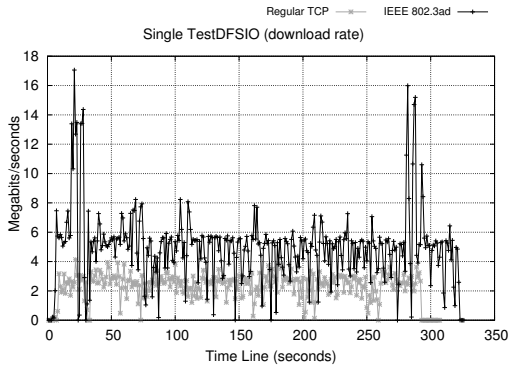

Fig. 10. Running single instance of TestDFSIO.

Figure 11 presents the results achieved when running with two instances of the TestDFSIO application. In this case, our approach demonstrates its efficiency compared to the Regular TCP environment. It becomes evident when comparing the completion times and traffic behavior. The utilization peaks that vary throughout the execution are even more interesting. With our approach, the application may take advantage of the greater bandwidth available, which in turn compensates for the overheads added by the IEEE 802.3ad implementation to manage network packages.

Moreover, Figure 12 illustrates the results achieved with three concurrent instances running the TestDFSIO. There is

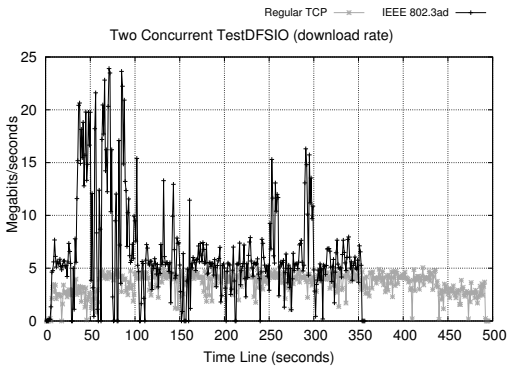

Fig. 11. Running two concurrent instances of TestDFSIO.

now a bigger difference among the tested approaches, with ours demonstrating superior in completion times due to the greater bandwidth available. This behavior coincides with the results achieved in the upload rate results (Figure 9). Finally, note that the download rates are lower than in the upload in the master node. This is primarily because the master receives a large amount of data during the Shuffle stage of Hadoop. However, we have concluded that TestDFSIO was able to perform better in the download and upload rates when there was higher network traffic.

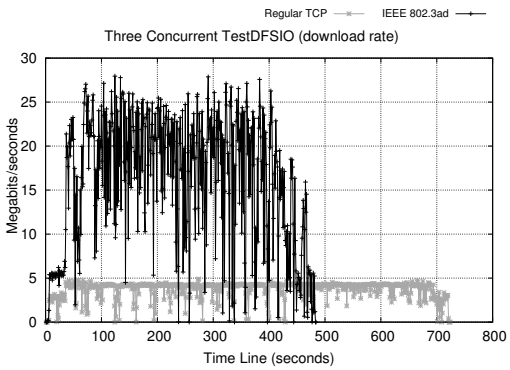

Fig. 12. Running three concurrent instances of TestDFSIO.

REFERENCES

[1] T. White, *Hadoop: The Definitive Guide*, 4th ed. O'Reilly Media, Inc., 2015.

[2] S. Radhakrishnan, B. J. Muscedere, and K. Daudjee, "V-Hadoop: Virtualized Hadoop Using Containers," in *15th International Symposium on Network Computing and Applications (NCA)*, Oct 2016, pp. 237–241.

[3] M. G. Xavier, M. V. Neves, and C. A. F. D. Rose, "A Performance Comparison of Container-Based Virtualization Systems for MapReduce Clusters," in *22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Feb 2014, pp. 299–306.

[4] A. Vogel, D. Griebler, C. A. F. Maron, C. Schepke, and L. G. L. Fernandes, "Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack," in *24rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. Heraklion Crete, Greece: IEEE, Febuary 2016, pp. 672–679.

[5] A. Vogel, D. Griebler, C. Schepke, and L. G. Fernandes, "An Intra-Cloud Networking Performance Evaluation on CloudStack Environment," in *25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. St. Petersburg, Russia: IEEE, March 2017, pp. 468–472.

[6] M. V. Neves, C. A. F. D. Rose, K. Katrinis, and H. Franke, "Pythia: Faster Big Data in Motion Through Predictive Software-Defined Network Optimization at Runtime," in *28th International Parallel and Distributed Processing Symposium (IPDPS)*, ser. IPDPS '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 82–90.

[7] R. F. E. Silva and P. M. Carpenter, "Controlling Network Latency in Mixed Hadoop Clusters: Do We Need Active Queue Management?" in *41st Conf. on Local Computer Networks*, Nov 2016, pp. 415–423.

[8] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing Public Cloud Providers," in *Proceedings of the 10th ACM SIGCOMM Conf. on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 1–14.

[9] "Amendment to carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications-aggregation of multiple link segments," *IEEE Std 802.3ad-2000*, pp. i–173, 2000.

[10] LXC, "Linux containers project (lxc)," 2016. [Online]. Available: https://linuxcontainers.org

[11] L. Yazdanov, M. Gorbunov, and C. Fetzer, "EHadoop: Network I/O Aware Scheduler for Elastic MapReduce Cluster," in *8th Inter. Conference on Cloud Computing (CLOUD)*, June 2015, pp. 821–828.

[12] G. Mencagli, M. Vanneschi, and E. Vespa, "A Cooperative Predictive Control Approach to Improve the Reconfiguration Stability of Adaptive Distributed Parallel Applications," *ACM Trans. Auton. Adapt. Syst.*, vol. 9, no. 1, pp. 2:1–2:27, Mar. 2014. [Online]. Available: http://doi.acm.org/10.1145/2567929

[13] Renner, Thomas and Thamsen, Lauritz and Kao, Odej, "Network-aware Resource Management for Scalable Data Analytics Frameworks," in *Inter. Conference on Big Data (Big Data)*, Oct 2015, pp. 2793–2800.

[14] P. Qin, B. Dai, B. Huang, and G. Xu, "Bandwidth-Aware Scheduling with SDN in Hadoop: A New Trend for Big Data," *arXiv preprint arXiv:1403.2800*, 2014.

[15] S. Prabhavat, H. Nishiyama, N. Ansari, and N. Kato, "Effective Delay-Controlled Load Distribution over Multipath Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 10, pp. 1730–1741, Oct 2011.

[16] J. Ekanayake and G. Fox, *High Performance Parallel Computing with Clouds and Cloud Technologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 20–38.

[17] A. Desai and K. Nagegowda, "Advanced Control Distributed Processing Architecture (ACDPA) Using SDN and Hadoop for Identifying the Flow Characteristics and Setting the Quality of Service(QoS) in the Network," in *Inter. Advance Computing Conference (IACC)*, June 2015, pp. 784–788.

[18] C. H. Wang, C. K. Yang, W. C. Liao, R. I. Chang, and T. T. Wei, "Coupling GPU and MPTCP to Improve Hadoop/MapReduce Performance," in *2nd International Conference on Intelligent Green Building and Smart Grid (IGBSG)*, June 2016, pp. 1–6.

[19] R. Rizki, A. Rakhmatsyah, and M. A. Nugroho, "Performance Analysis of Container-Based Hadoop Cluster: OpenVZ and LXC," in *4th International Conference on Information and Communication Technology (ICoICT)*, May 2016, pp. 1–4.

[20] NetPipe, "A network protocol independent performance evaluator (netpipe)," 2016. [Online]. Available: http://bitspjoule.org/netpipe

[21] D. Eadline, *Hadoop 2 Quick-Start Guide: Learn the Essentials of Big Data Computing in the Apache Hadoop 2 Ecosystem*, 1st ed. Addison-Wesley Professional, 2015.

[22] J. Long, *Storage Networking Protocol Fundamentals*. Pearson Education, 2013.

[23] G. Tomar, R. Chang, O. Gervasi, T. Kim, and S. Bandyopadhyay, *Advanced Computer Science and Information Technology: Second International Conference, AST 2010, Miyazaki, Japan, June 23-25, 2010. Proceedings*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2010.

[24] A. Ford, C. Raiciu, M. J. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 6824, Jan. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc6824.txt