IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems

MRemu: An Emulation-based Framework for Datacenter Network Experimentation using Realistic MapReduce Traffic

Marcelo Veiga Neves, Cesar A. F. De Rose
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre, Brazil
marcelo.neves@pucrs.br, cesar.derose@pucrs.br

Kostas Katrinis
IBM Research – Ireland
Dublin, Ireland
katrinisk@ie.ibm.com

***Abstract*—As data volumes and the need for timely analysis grow, Big Data analytics frameworks have to scale out to hundred or even thousands of commodity servers. While such a scale-out is crucial to sustain desired computational throughput/latency and storage capacity, it comes at the cost of increased network traffic volumes and multiplicity of traffic patterns. Despite the sheer reality of the dependency between datacenter network (DCN) and time-to-insight through big data analysis, our experience as active networking researchers conveys that a large fraction of DCN research experimentation is conducted on network traces and/or synthetic flow traces. And while the respective results are often valuable as standalone contributions, in practice it turns out extremely difficult to quantitatively assess how the reported network optimization results translate to performance or fault-tolerance improvement for actual analytics runtimes, e.g., due to the ability of these runtimes to overlap communication with computation. This paper presents MRemu, an emulation-based framework for conducting reproducible datacenter network research using accurate MapReduce workloads and at system scales that are relevant to the size of target deployments, albeit without requiring access to a hardware infrastructure of such scale. We choose the MapReduce (MR) framework as a design point, for it is a common representative of the most widely deployed frameworks for analysis of large volumes of - structured and unstructured - data and is reported to be highly sensitive to network performance. With MRemu, it is possible to quantify the impact of various network design parameters and software-defined control techniques to key performance indicators of a given MR application. We show through targeted experimental validation that MRemu exhibits high fidelity, when compared to the performance of MR applications on a real scale-out cluster of 16 high-end servers.**

I. Introduction

The rise of Internet of Things sensors, social networking and mobile devices has led to an explosion of data available for analysis towards knowledge gaining and final insights. In turn, this has led into the development of dedicated platforms for large-scale data analysis. The MapReduce (MR) framework, as implemented in Hadoop [1], is one of the most popular frameworks for Big Data analysis. To handle the ever-increasing data size, Hadoop is a scalable framework that allows a dedicated and seemingly unbound number of servers to participate in the analytics process. The response time of an analytics request is an important factor for time to value/insights. While the computing and disk I/O requirements can be scaled with the number of servers, scaling the system leads to increased network traffic. Evidently, the communication-heavy phases of MR contributes significantly to the overall response time [2].

Despite Big Data analytics being a very common workload in datacenters, most research on datacenter networks does not really take it into consideration. Instead, researchers normally use synthetic traffic patterns (e.g., random traffic following probabilistic distributions) to evaluate network performance [3], [4], [5], [6]. While useful to rapidly evaluate new algorithms and techniques, this approach often fails to capture the real strain on datacenter networks supporting MR-like runtimes. Thus, it is difficult to determine how these studies would perform in the presence of MR workloads and, most importantly, how they impact MR response time. In fact, recent research has shown that MR applications are sensitive to network performance [7], [2] and that it is possible, by leveraging the emergent technology of software-defined networks (SDN), to develop network control software to adapt the network to the application's needs in order to accelerate the execution of this kind of application [7], [8].

Some studies in the literature use MapReduce-like traffic patterns to evaluate their research [9], [10], [6]. They normally model the MR shuffle pattern as map tasks (typically one per node) sending data to one or more reduce tasks. However, MapReduce frameworks, such as Hadoop, implement some mechanisms to improve network performance that are not taken into account by these works (e.g., transfer scheduling decisions, number of parallel transfers, etc.). Moreover, normally there are a large number of map tasks per node and Hadoop nodes have to serve data to multiple reduce tasks concurrently. As a result, a great deal of research is being conducted using synthetic traffic patterns and may not perform well when applied to real MapReduce applications, particularly those that claim to improve MapReduce performance.

Research in this area often relies on analytical and simulation models to evaluate the network performance, which may not capture all of the characteristics of real networks and applications. The use of real hardware, on the other hand, is often not a valid option, since many researchers do not have access to datacenters robust enough to run data-intensive applications. Moreover, even when datacenter resources are available, it is normally not practical to reconfigure them in order to evaluate different network topologies and characteristics (e.g., bandwidth and latency). In this context, an alternative is to use emulation-based testbeds. In fact, network emulation has been successfully used to reproduce network research experiments with a high level of fidelity [11]. Although existing network emulation systems allow for the use of arbitrary network topologies, they typically run on a single physical node and use some kind of virtualization (e.g., Linux containers) to emulate the datacenter nodes. Therefore, they lack resources to run real Hadoop jobs, which are known to be CPU and IO-intensive.

1526-7539/15 $31.00 © 2015 IEEE
DOI 10.1109/MASCOTS.2015.36

To overcome the above gap, we propose to combine network emulation with trace-driven MapReduce emulation. For this, we have implemented a framework called MRemu that reproduces executions of MapReduce jobs by mimicking Hadoop MapReduce internals (e.g., scheduling decisions) and generating traffic in the same way a real Hadoop job would. The proposed framework enables networking research experiments targeting datacenter networks using SDN and MapReduce-like workloads, however by slashing the requirements of continuous access to a large infrastructure. For example, it is possible to compare the performance (e.g., job completion time) of a given MapReduce job on different network topologies and network control software. The MapReduce emulation also works as a stand-alone tool that can be used to run network experiments in clusters with limited resources (i.e., not robust enough to run real Hadoop jobs [10]). MRemu sports following features and functionality:

- Ability to create arbitrary network topologies, including complex multi-path topologies (e.g., fat-tree), and network parameters (e.g., bandwidth, latency, delay, and packet-loss ratio);
- Accurate reproduction of MapReduce workloads, including jobs with skewed transfer patterns, without requiring a real datacenter infrastructure;
- Support for evaluating real software-defined network (SDN) control code running on production SDN controllers, through interfacing MRemu with SDN controllers via the OpenFlow protocol and thus slashing "from-lab-to-market" integration times. We have successfully tested MRemu working in tandem with production SDN controllers, notably OpenDaylight [12] and POX/NOX [13].

This paper is structured as follows. Section II outlines background information related to MapReduce and associated data movement patterns. Section III presents the proposed testbed, including network emulation framework and MapReduce workload generation. Sections IV presents evaluation results manifesting the accuracy and impact of our approach and framework embodiment. Conclusions and future work are presented in Section V.

II. Background

This section provides an overview of the Hadoop MapReduce implementation. Although there are currently several implementations that are functionally equivalent to MapReduce (e.g., Hadoop [1], Dryad [14], Twister [15], Spark [16]), this work will focus on Hadoop because it is one of the most popular open-source implementations for Big Data analysis implementations.

The Hadoop framework can be roughly divided in two main components: the Hadoop MapReduce, an open-source realization of the MapReduce model and the Hadoop Distributed File System (HDFS), a distributed file system that provides resilient, high-throughput access to application data. The execution environment includes a job scheduling system that coordinates the execution of multiple MapReduce programs, which are submitted as batch jobs. Assuming that the input data is already loaded into the distributed file system (i.e. excluding data movement due to loading/exporting data to/from HDFS), intensive data movement in Hadoop is mainly attributed to shuffling intermediate mapper output to reducers. For instance, a recent analysis of MR traces from Facebook revealed that 33% of the execution time of a large number of jobs is spent at the MR phase that shuffles data between the various data-crunching nodes [2]. This same study also reports that for 26% of Facebooks MR jobs with reduce tasks, the shuffle phase accounts for more than 50% of the job completion time, and in 16% of jobs, it accounts for more than 70% of the running time.

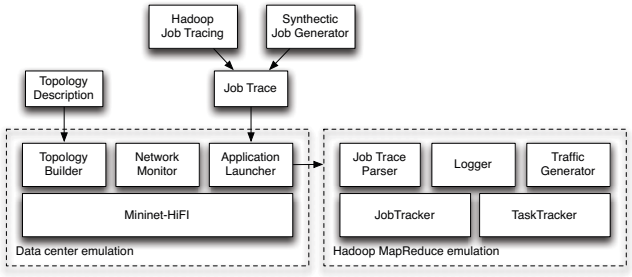

Fig. 1. Block diagram showing MRemu architecture.

Hadoop Data Shuffling. In the shuffle phase of a MR job, each reduce task collects the intermediate results from all completed map tasks. Reduce tasks are normally scheduled after a fraction of map tasks have been completed (by default 5%). Once running, a reduce task does not wait for all map tasks to be completed to start fetching map results. Instead, it starts scheduling copier threads to copy map output data as soon as each map task commits and the data becomes available. This technique (often referred to as early shuffle) causes the overlap between the execution of map tasks and the shuffle phase, which typically reduces the job completion time. However, the reduction itself starts only after all map tasks have finished and all intermediate data becomes available, which works as an implicit synchronization barrier that is affected by network performance.

III. The MRemu Framework

This section describes the design and architecture of the proposed emulation-based experimentation framework to enable networking experiments with MapReduce applications. As introduced earlier, the main goals of this work are (1) to enable the evaluation of network design parameters to MapReduce-like application performance (e.g., job completion time), (2) to enable the evaluation of novel software-defined datacenter network control algorithms and policies to MapReduce-like applications and (3) to provide for a means of identifying network bottlenecks and optimization opportunities caused by new MapReduce-like applications (e.g. workflows comprising inter-dependent MR jobs). For this, we decided to combine network emulation with trace-driven MapReduce emulation, as conveyed in the block diagram of Figure 1 showing the functional blocks of the MRemu framework. The rest of this section will explain each of the proposed system's components in detail.

A. MapReduce Job Tracing

The emulation-based experiments are driven by job traces that can be either extracted from past job executions or synthetically generated by statistical distributions. Hadoop has a built-in tool, called Rumen [17], that generates job trace files for past executions. However, the trace data produced is insufficient for network-related analysis, since it lacks information about individual network transfers. Thus, we developed a

new tool for extracting meaningful information from Hadoop logs (including network-related information) and generating comprehensive MR execution traces. This tool is also able to create job traces with different numbers of nodes and tasks by using linear extrapolation, which is particularly useful for scaling experiments for setups larger than those where the job traces were collected from.

B. Data Center Emulation

This work relies on Mininet-HiFi [11] for network emulation. Mininet-HiFI, also called Mininet 2.0, is a container-based network emulation tool designed to allow reproducible network experiments. It extends the original Mininet [18] with features for performance isolation, resource provisioning, and performance fidelity monitoring. Performance isolation and resource provisioning are provided using Linux containers features (e.g, cgroups and namespaces) and network traffic control (tc). Fidelity is achieved by using probes to verify that the amount of allocated resources were sufficient to perform the experiment.

We have developed a tool for automatically setting up a datacenter network experiment scenario, launching the experiment code and monitoring performance information. It consists of three main components: TopologyBuilder, ApplicationLauncher and NetworkMonitor. The TopologyBuilder component ingests a datacenter network description file, which can be either the topology file extracted from Hadoop traces (cf. Section III-A) or a manually created file supporting the use of any arbitrary network topology, and uses Mininet's Python APIs to create a complete network topology. ApplicationLauncher is an extensive component that launches the application code in each emulated node and waits for its completion. Currently, it supports two applications: the MapReduce emulator (cf. Section III-C) and a synthetic traffic generator using iperf. Finally, NetworkMonitor allows for probes to be installed in every node or network element to collect monitoring information. Examples of information that can be collected are network bandwidth usage, queue length, number of active flows, CPU usage, etc.

C. Hadoop MapReduce Emulation

We implemented a tool that reproduces executions of MapReduce jobs by mimicking Hadoop MapReduce internals (e.g., scheduling decisions) and generating traffic in the same way a real Hadoop job does. Although it internally simulates part of MapReduce functionality, it constitutes a MapReduce emulation tool from the system networking point of view: to the network system, our evaluation tool has exactly the same effect as a real MapReduce application producing real network traffic and logging events to local files. This also allows, for example, plugging systems that extract information from Hadoop logs to predict network transfers [7], [8]. Since we are interested mainly in the MapReduce shuffle phase, simulating the details of Hadoop daemons and tasks (e.g., task processing internals, disk I/O operations, control messages, etc.) is not a goal, unlike MapReduce simulators (e.g., MRPerf [19]). Instead, we use information extracted from job traces (e.g., task durations, wait times, etc.) to represent the latencies during different, non-shuffling phases of the MapReduce processing.

Our emulator implements two operation modes: REPLAY, which reproduces the exact scheduling decisions from MR application execution log traces, and HADOOP, which computes new scheduling decisions based on the same algorithms that Hadoop uses to schedule jobs and transfers used by Hadoop. As the name conveys, the first mode allows us to reproduce the exact order and timing of individual transfers, as these occurred during execution of the MR application on a real scale-out machine. The second mode may not reproduce the same order and execution timing, but allows us to employ trace-driven evaluation by varying the network parameters and network control logic, without having to replicate the respective network configuration of the datacenter that the traces were collected from.

It is worth mentioning that the Hadoop emulation system can also be used as a standalone tool. Although we have developed MRemu to run in container-based emulation environments, it is also possible to use it to perform experiments on real hardware testbeds that are not robust enough (due to ,e.g., lack of memory, compute or I/O rate) to execute real large-scale data-intensive jobs.

IV. EVALUATION

This section describes the experiments conducted to evaluate the fidelity of the MRemu implementation. Since Mininet-HiFi has already been validated [11] and is widely used to reproduce networking research experiments [20], we focus on the evaluation of our MapReduce emulation tool and its ability to accurately reproduce MapReduce workloads.

Our experiments are based on job traces extracted from executions in a real cluster. These traces are used both as input for our emulation system and baseline to evaluate the emulation results accuracy. The cluster setup we have used consists of 16 identical servers, each equipped with 12 x86_64 cores and 128 GB of RAM. The servers are interconnected by an Ethernet switch with 1Gbps links. All servers run Hadoop 1.1.2. The emulation-based experiments were executed on a single server with 16 x86_64 cores and 16 GB of RAM. Our emulation tool runs within Mininet 2.1.0+. We selected popular benchmarks as well as real applications that are representative of significant uses of MapReduce (e.g., data transformation, web search indexing and machine learning). The selected applications are part of the HiBench Benchmark Suite [21], which includes Sort, Nutch, PageRank and Bayes.

A. Evaluation of the Job Completion Time Accuracy

In order to evaluate the accuracy of our Hadoop emulation system, we performed a comparison between job completion times in real hardware and in the emulation environment. We tested both REPLAY and HADOOP operation modes. The graph in Figure 2 shows normalized job completion times compared to those extracted from the original traces. As can be observed, job completion times for emulation in the REPLAY mode are very close to the ones observed in the original execution traces. When using the HADOOP mode (i.e., the one that computes new scheduling decisions, instead of directly using times extracted from traces), the completion times are are slightly deviating for the Bayes application.

B. Evaluation of Individual Network Flow Completion Time Accuracy

After validating the system accuracy in terms of job completion time, we conducted experiments to evaluate individual flow durations. The graph in Figure 3 shows the Cumulative Distribution Function (CDF) for completion times of flows during the shuffle phase of the Sort application. We compared the results of the emulated execution with times extracted from the original execution trace. Although the mean time is very

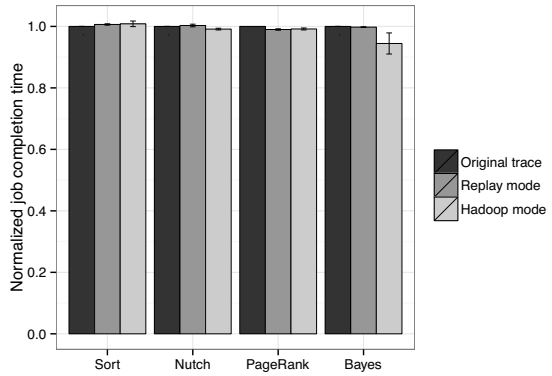

Fig. 2. Comparison between job completion times in real hardware and in the emulation environment for different MapReduce applications.

close, we can see in Figure 3 that the transfers' durations were slightly different. This behavior is expected since we are inferring flow durations from Hadoop logs, which may not represent the exact system behavior due to high-level latencies. Especially for small transfers, we detected that our traffic generation imposes an overhead, since it needs to start two new processes (client and server) at each new transfer (we plan to improve this in future). Nevertheless, the system still achieved completion times very close to the ones extracted from Hadoop traces and, as shown in Section IV-A, it leads to accurate job completion times.

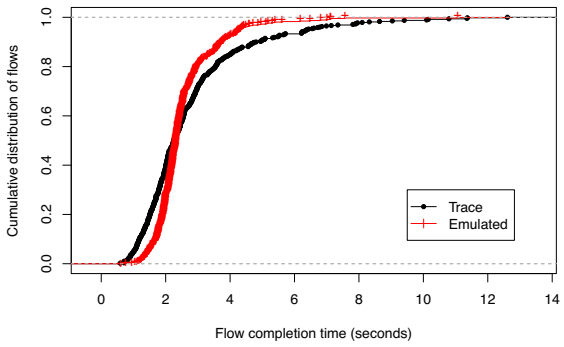

Fig. 3. Comparison of the cumulative distribution of flows durations in emulated execution and original traces.

V. Conclusions and Future Work

This paper presented MRemu, an emulation-based framework that enables conducting datacenter network research and resource planning, without requiring expensive and continuous access to large-scale datacenter hardware resources. Among the highlights of MRemu is a) the ability to process logs and traces from executions of real MapReduce applications in production datacenters and replay or extrapolate patterns to facilitate realistic network emulation in the context of datacenter network evaluation, b) the ability to emulate network control code for software-defined networks (SDN) implemented and running directly on the production SDN controller and c) the potential of its use as a standalone tool in legacy computer clusters with limited resources for real data-crunching. We verified through rigorous and methodical experimentation that MRemu exhibits high accuracy of emulation with respect to application performance, when compared to respective application runs in a real datacenter. As such, we are confident that MRemu can be used by the broader research community as a valuable tool in its experimentation toolset. In fact, we are working in this direction and towards making MRemu available to the community as an open-source tool[1]. As part of our next steps, we are working on overcoming its limitations (e.g, distributed emulation, multi-job execution), as well as incorporating further analytics runtimes beyond Hadoop MapReduce into the emulator.

References

[1] Apache Software Foundation, "Apache Hadoop," 2015, accessed on March 2015. [Online]. Available: http://hadoop.apache.org

[2] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing Data Transfers in Computer Clusters with Orchestra," in *Proceedings of the ACM SIGCOMM 2011 Conference*, 2011.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.

[4] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks," in *Proceedings of the USENIX NSDI 2010 Conference*, 2010.

[5] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," *Commun. ACM*, vol. 54, no. 3, Mar. 2011.

[6] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling Flow Management for High-Performance Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, Aug. 2011.

[7] A. Das, C. Lumezanu, Y. Zhang, V. Singh, and G. Jiang, "Transparent and Flexible Network Management for Big Data Processing in the Cloud," in *5th USENIX Workshop on Hot Topics in Cloud Computing*, San Jose, CA, US, Jun. 2013.

[8] M. Neves, K. Katrinis, H. Franke, and C. De Rose, "Pythia: Faster Big Data in Motion through Predictive Software-Defined Network Optimization at Runtime," in *Proceedings of the IEEE IPDPS 2014 Conference.*, 2014.

[9] W. Cui and C. Qian, "DiFS: Distributed Flow Scheduling for Adaptive Routing in Hierarchical Data Center Networks," in *Proceedings of the ACM/IEEE ANCS 2014 Conference*, 2014.

[10] W. C. Moody, J. Anderson, K.-C. Wang, and A. Apon, "Reconfigurable Network Testbed for Evaluation of Datacenter Topologies," in *Proceedings of the ACM DIDC 2014 Conference*, 2014.

[11] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible Network Experiments Using Container-Based Emulation," in *Proceedings of the ACM CoNEXT 2012 Conference*, 2012, pp. 253–264.

[12] Linux Foundation. (2015) OpenDaylight Project. Accessed on March 2015. [Online]. Available: http://www.opendaylight.org

[13] NOX/POX, "NOX/POX OpenFlow Controller," 2015, accessed on March 2015. [Online]. Available: http://www.noxrepo.org

[14] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," in *Proceedings of the ACM SIGOPS/EuroSys 2007 Conference*, 2007.

[15] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: A Runtime for Iterative MapReduce," in *Proceedings of the ACM HPDC 2010 Conference*, 2010, pp. 810–818.

[16] Apache Software Foundation, "Apache Spark," 2015, accessed on March 2015. [Online]. Available: http://spark-project.org.

[17] ——, "Rumen," 2015, accessed on March 2015. [Online]. Available: http://goo.gl/8W4Riz

[18] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-defined Networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.

[19] G. Wang, A. R. Butt, P. Pandey, and K. Gupta, "A Simulation Approach to Evaluating Design Decisions in MapReduce Setups," in *Proceedings of the IEEE MASCOTS 2009 Conference*, 2009.

[20] Mininet, "Reproducing Network Research," 2015, accessed on March 2015. [Online]. Available: http://goo.gl/idfDBR

[21] Intel Corporation. (2015) HiBench. Accessed on March 2015. [Online]. Available: https://github.com/intel-hadoop/HiBench

[1] https://github.com/mvneves/mremu