A Performance Study of Big Data Workloads in Cloud Datacenters with Network Variability

Alexandru Uta
Vrije Universiteit Amsterdam
The Netherlands
A.Uta@vu.nl

Harry Obaseki
Vrije Universiteit Amsterdam
The Netherlands
i.h.obaseki@student.vu.nl

ABSTRACT

Public cloud computing platforms are a cost-effective solution for individuals and organizations to deploy various types of workloads, ranging from scientific applications, business-critical workloads, e-governance to big data applications. Co-locating all such different types of workloads in a single datacenter leads not only to performance degradation, but also to large degrees of performance variability, which is the result of virtualization, resource sharing and congestion. Many studies have already assessed and characterized the degree of resource variability in public clouds. However, we are missing a clear picture on how resource variability impacts big data workloads. In this work, we take a step towards characterizing the behavior of big data workloads under network bandwidth variability. Emulating real-world clouds' bandwidth distribution, we characterize the performance achieved by running real-world big data applications. We find that most big data workloads are slowed down under network variability scenarios, even those that are not network-bound. Moreover, the maximum average slowdown for the cloud setup with highest variability is 1.48 for CPU-bound workloads, and 1.79 for network-bound workloads.

ACM Reference Format:
Alexandru Uta and Harry Obaseki. 2018. A Performance Study of Big Data Workloads in Cloud Datacenters with Network Variability. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering Companion , April 9–13, 2018, Berlin, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3185768.3186299

1 INTRODUCTION

Cloud computing platforms have become the de facto platform for running various types of workloads, from either individuals and institutions. The economy of scale helps cloud providers reduce costs through co-locating client workloads by means of separating them through virtual machines (VMs), which offer user isolation. Such virtual machines are scheduled on the same physical hardware for an increased total resource utilization. In such setups, a challenging problem appears: *resource variability*. As currently networks are a more scarce resource than, for example, processing power, network variability is more likely to impact application performance, or, more importantly, performance predictability. As many studies [3, 31] already point out, in public clouds the network performance exhibits large degrees of variability - due to virtualization, colocation and congestion overheads [4, 15]. In this paper, we study the performance implications of network variability when running big data workloads.

Due to the data deluge, a new paradigm of data driven science has emerged [11]. As a consequence, big data processing systems have become pervasive. Such systems handle many types of workloads, ranging from scientific applications [24], graph-processing [38], analytics and data-warehousing solutions [1, 33], to machine learning [21], where data collection, curation, and analysis play an increasingly important role in preparing data for artificial intelligence algorithms [32]. Big data processing systems, such as Hadoop [37] or Spark [40] have already been deployed successfully on cloud computing environments. Moreover, for clients that choose not to manage their own VM cluster, commercial clouds even offer Hadoop, or Spark as a service[1]. The key problem when deploying such big data processing systems in cloud computing environments is that they do not run in isolation, and hence are prone to be impacted by resource variability.

This issue is exacerbated by the big data processing systems' intrinsic design: such systems are built to run in isolation, in private cluster computing environments. Therefore, the computation is optimized for, and achieves predictable performance on homogeneous and symmetric setups. A limited number of approaches try to address the issue of network heterogeneity in big data platforms [10, 23, 28, 41], but are either tailored towards the MapReduce ecosystem, or consider multi-cloud environments and do not optimize for the variability within a single cloud datacenter.

Moreover, such approaches suffer from several significant shortcomings. First, these solutions focus on bandwidth heterogeneity, not *variability*. Cloud network bandwidth is not only heterogeneous, but also highly variable with time, with possibly highly unpredictable behavior. Second, they are not evaluated under real-world cloud bandwidth distributions. Furthermore, such solutions do not offer any performance predictability guarantees, models, or insights into how variable network affects the behavior of big data workloads.

We advocate for performance predictability and for devising realistic performance models for big data applications even under the influence of variable networks. Even though recent seminal work [26] towards understanding the performance of big data platforms shows that network is not a performance bottleneck, this

ICPE '18, April 9–13, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5629-9/18/04…$15.00
https://doi.org/10.1145/3185768.3186299

[1] https://aws.amazon.com/emr/

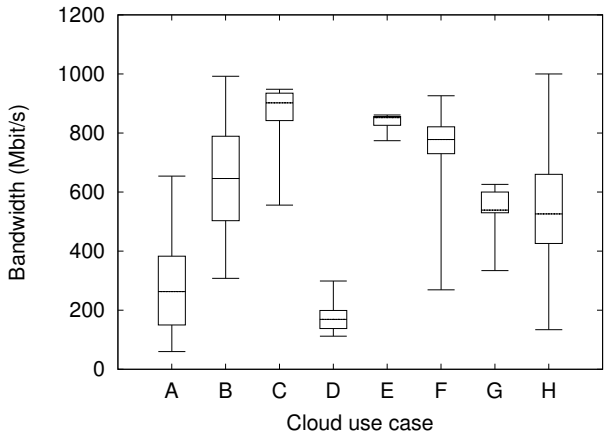


Figure 1: Bandwidth distributions for A-H cloud use cases. 1st-25th-50th-75th-99th percentiles. Data points courtesy of authors of [3].

does not imply that its variability does not impact application performance (predictability). We conjecture that network variability generates performance variability in big data processing systems.

Towards validating this statement, the following research questions arise: *how to assess the impact of network bandwidth variability on big data workloads running on clouds?. What is the impact of network bandwidth variability when running big data workloads on clouds?*

To answers these research questions, we make the following contributions:

(1) We create a framework to emulate the network variability behavior of eight real-world clouds [3] (Section 2).
(2) We characterize the performance of six real-world big data applications from the HiBench suite [12] running on top of Spark (Section 3).

2 EXPERIMENTAL SETUP

In this section we present our experimental framework for assessing the impact of network bandwidth variability over big data processing workloads. We introduce the eight real-world cloud bandwidth distributions that we emulate and the mechanisms used to implement the emulation. Furthermore, we present the six real-world big data workloads we run in our emulated setup. We conclude this section by presenting the hardware and software platforms we used for conducting the experiments.

2.1 Emulated Cloud Scenarios

In [3], Ballani et al. present network bandwidth variability results for eight real-world cloud setups in the form of bandwidth distributions. Figure 1 summarizes their findings. Out of eight clouds, six are highly variable in network bandwidth, with differences between minimum and maximum values as high as one order of magnitude (e.g., distribution H).

These bandwidth distributions are the results of benchmarking network traffic between virtual machines in the respective clouds

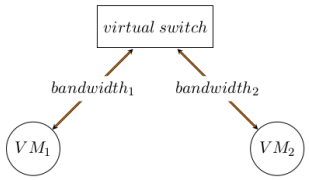


Figure 2: Graph representation of the hose model

Table 1: Resource Utilization for HiBench Workloads.

Application	CPU	Disk	Memory	Network
Wordcount	High	Low	Moderate	Moderate
Sort	Low	High	Moderate	High
Terasort	High	Moderate	High	High
Naive Bayes	Moderate	Moderate	High	Low
K-means	High	Low	Moderate	Low
Pagerank	Moderate	Low	Moderate	Low

over relatively coarse periods of time (hours-days). These results provide clear upper and lower bounds of achievable performance when two virtual machines communicate in a cloud. However, due to the uncertainty of the underlying conditions (e.g. networking activity bursts of other cloud customers' virtual machines co-located on the same physical infrastructure), the bandwidth variation granularity could be much finer (e.g. seconds, minutes). In this work, we only consider scenarios where the achievable bandwidth of one VM does not change during the course of a single experiment. Changing the bandwidth distribution during the course of an experiment remains for future work.

2.2 The Hose Model

As shown in Section 4, there are several models to implement bandwidth guarantees in cloud computing setups. We have chosen to implement the older Hose Model [8], due to its simpler design and flexibility: in our use-case of emulating real-world setups, we want to provide VMs with bandwidth guarantees given by the distributions in Figure 1.

In the hose model (Figure 2), the VM cluster is connected to a *virtual switch* that guarantees, for each endpoint (i.e., VM) a certain *egress* and *ingress* bandwidth. Therefore, using the aforementioned bandwidth distributions, we only have to specify the pair of (egress, ingress) guarantees for each VM in the cluster. To implement the hose model in our setup, we use VM-based rate-limiting achieved through the Linux *tc* [13] tool.

2.3 Big Data Workloads

We selected a number of six big data applications from the HiBench [12] suite: Wordcount, Sort, Terasort, K-Means, Bayes, Pagerank. Table 1 presents the resource usage characteristics of these workloads. We characterize these applications in terms of CPU, Disk, Memory and Network utilization, using three increments *low* (1%-30% utilization), *moderate* (31%-60% utilization), and *high*

(61%-100%). This characteristics are determined from both the HiBench [12] study and our own empirical analysis. We consider these workloads representative for a wide range of big data applications, as they cover a wide spectrum of resource utilization combinations. The HiBench *data size* parameter is set to *large*. Experimenting with larger problem sizes remains for future work.

2.4 Experimental Workflow

The logical workflow of our experiments is as follows. We deploy 16 VMs in our OpenNebula cluster. Then, we limit the VMs network bandwidth according to each of the eight bandwidth distributions depicted in Figure 1. For each bandwidth setup, we run the six big data applications. Each application is executed 10 times, and we record average running times and the standard deviation. The six big data workloads are run on top of Spark [40], a widely-used in-memory cluster computing engine, which extends the classical MapReduce model with a very expressive and extensive API.

2.5 Hardware and Software Platforms

We perform our experiments on the local DAS-4 cluster [2]. Each compute node is equipped with a dual-quad-core Intel E5620 2.4 GHz CPUs and 24GB memory. The nodes are connected by a commodity 1Gb/s Ethernet and a Quad Data Rate (QDR) InfiniBand providing a theoretical peak bandwidth of 32Gb/s. Since the eight bandwidth distributions presented in Figure 1 contain values of at most 1Gb/s, in our experiments we use the 1Gb/s Ethernet.

To provision the VMs, the DAS-4 cluster runs OpenNebula [25]. The VMs run Spark 2.0.2, Hadoop 2.7.3, and HiBench 5.0. For our experiments, we provision 16 VMs, each equipped with 8 virtual cores, 20GB memory, and a 30GB disk. For our experiments, each node runs a Spark worker that uses 7 virtual cores (1 virtual core is reserved for the operating system) and 18GB memory (2GB are reserved for the operating system).

3 EXPERIMENT RESULTS

In this section we present the results of our study. Using the experimental platform introduced in Section 2, we run six real-world big data workloads on eight real-world cloud bandwidth distributions. Our main findings are:

(1) Our experimental framework is effective in emulating bandwidth variability for the provisioned VMs.
(2) Network-bound workloads (e.g., Sort, Terasort) are impacted most by bandwidth variability (up to 1.79 slowdown).
(3) Even CPU-bound applications (e.g., Wordcount, K-means) exhibit significant slowdowns (up to 1.48 slowdown).
(4) Generally, the more variable setups generate more variance when repeatedly performing the same experiment.

3.1 Validating the Emulated Setup

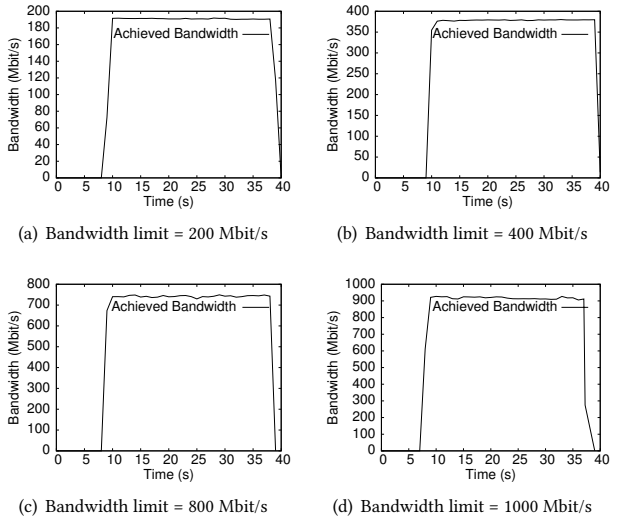
We first assess whether our hose model implementation is able to emulate the eight cloud bandwidth distributions. We conduct experiments using the Linux *iperf* [34] tool. We set pairs of VMs to communicate using four different bandwidth values. Figures 3(a)-3(d) show the achieved bandwidth of a virtual machine when its network traffic is limited to $bw \in \{200, 400, 800, 1000\}$ Mb/s. We notice that the achieved values are, in practice, slightly lower than the theoretical limitation. However, this is an expected behaviour and it is important to notice that the difference is always below 10%.

3.2 The Implications of Network Variability on Big Data Workloads

We run the six big data workloads on the eight bandwidth distributions. We compare the results achieved with a setup where the bandwidth is not limited, and thus can make use of the entire 1Gbps the system can achieve. Figure 4 plots the results. For each application and bandwidth distribution we made 10 repetitions. We plot the average runtime of each of the six application, and we also report the variability range.

We find that the more network-bound workloads (i.e., Terasort, Sort) are not only slower than in the homogeneous setup, but also exhibit more variability between runs, as shown by the range of the error bars. As expected, the distributions that contain lower bandwidth values (A, D, F, H) give the slowest average running times for each of the six workloads.

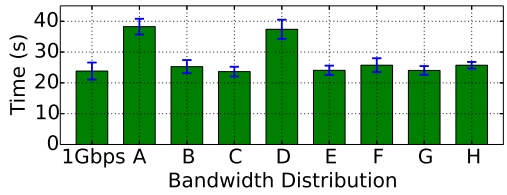
We quantify the average slowdown induced by the network variable bandwidth distributions. The results are depicted in Figure 5. All applications are slowed down by 7 out of 8 scenarios. The only distribution that does not slow down any application is C. This is an expected result, as 75% of its bandwidth values are clustered close together in an interval between 950 Mbit/s and 850 Mbit/s, therefore having low variability. Furthermore, we find that distributions A and D generate the largest slowdowns. These two distributions contain the smallest bandwidth values and also a large amount of variability. It is interesting to notice that the distributions with a median value higher than 500 Mbit/s (B, F, G, H) generate small, but still significant slowdowns.



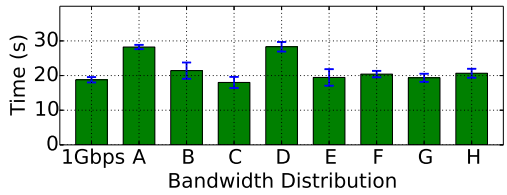
(a) Bandwidth limit = 200 Mbit/s (b) Bandwidth limit = 400 Mbit/s

(c) Bandwidth limit = 800 Mbit/s (d) Bandwidth limit = 1000 Mbit/s

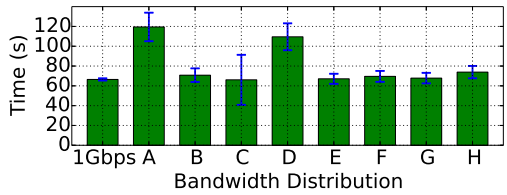
Figure 3: VM achieved bandwidth when network link is limited to $bw \in \{200, 400, 800, 1000\}$.



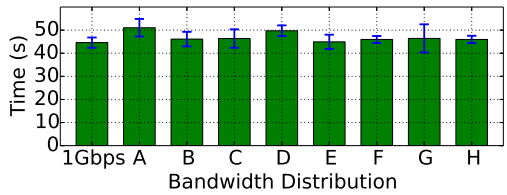
(a) Wordcount



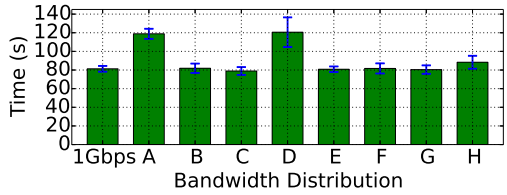
(b) Sort



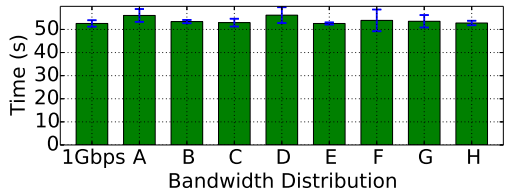
(c) Terasort



(d) Naive Bayes

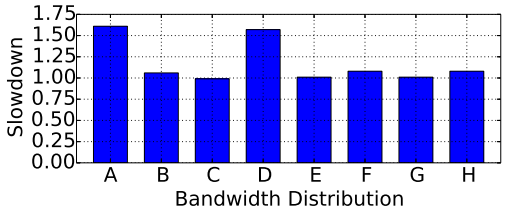


(e) K-Means

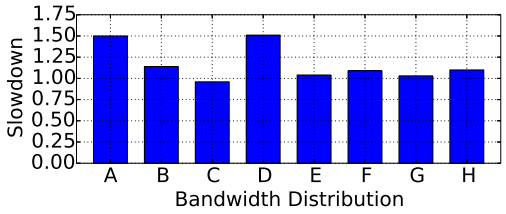


(f) Pagerank

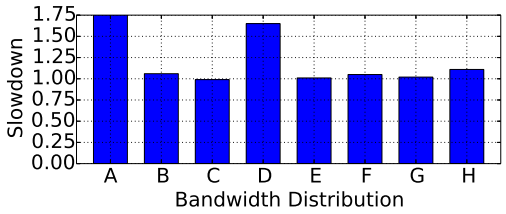
Figure 4: Average Runtime for Wordcount (a), Sort (b), Terasort (c), Naive Bayes (d), K-Means (e), and Pagerank (f) when deployed on the eight bandwidth distributions and a 1Gbps homogeneous setup.



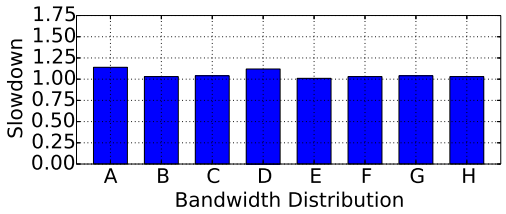
(a) Wordcount



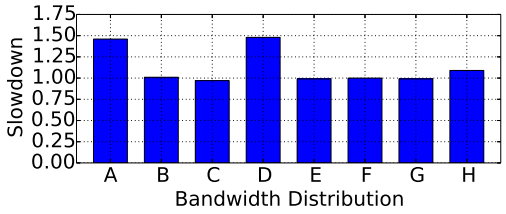
(b) Sort



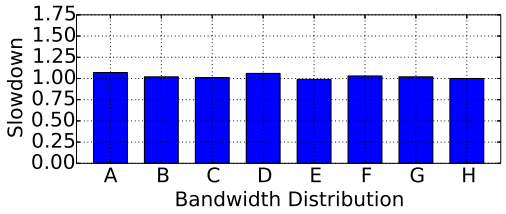
(c) Terasort



(d) Naive Bayes



(e) K-Means



(f) Pagerank

Figure 5: Application Runtime Slowdown fir Wordcount (a), Sort (b), Terasort (c), Naive Bayes (d), K-Means (e), and Pagerank (f) when deployed on the eight bandwidth distributions.

Table 2: Maximum slowdown per application.

Application	Maximum Slowdown	Bandwidth Distribution
Wordcount	1.61	A
Sort	1.51	D
Terasort	1.79	A
K-Means	1.48	D
Bayes	1.14	A
Pagerank	1.07	A

Finally, we quantify the maximum slowdown per application. Table 2 depicts the results. The maximum slowdown ranges between 1.07 (Pagerank) and 1.79 (Terasort). It is clear that the most slowed down application is the most network-bound. However, even the more CPU-bound applications, such as K-Means, or Wordcount suffer from significant performance degradation. Moreover, the two slowest bandwidth distributions (e.g., A and D) generate the largest amount of slowdown.

In [26], Ousterhout et al. show that networks are generally not the bottleneck when running big data applications. Our study complements such results by providing empirical evidence that network variability has performance variability implications. Therefore, even though network is not necessarily a bottleneck for big data workloads, the absence of predictable network performance also implies the absence of predictable application performance.

4 RELATED WORK

In this section we discuss results and studies related to our work. We identify two categories: (i) literature that quantifies and assesses network variability in clouds; (ii) models for providing infrastructure-level solutions to achieving predictable network performance; (iii) literature that addresses the problem of scheduling various types of workloads in datacenters with heterogeneous network links.

4.1 Network Variability in Cloud Datacenters

Many studies [3, 14, 17–19, 27, 31, 36, 41] have already assessed the performance variability of cloud datacenter networks. Cloud vendors offer little to no guarantees when it comes to the network. The main finding of the work in this category is that network performance is highly variable, in terms of both latency and bandwidth. The latter's performance varies in some instances by even a factor or 5 or more. Moreover, both TCP and UDP performance are similarly impacted.

4.2 Bandwidth Guarantees in Datacenters

Motivated by these findings, there are many approaches [3, 5, 9, 22, 29, 30] to solve the variability problem at the datacenter infrastructure layer.

Distributed Rate Limiting [29] (DSL) imposes an overall limit on the sum of the traffic generated by all of the sites for a given tenant (if the tenant has VMs in multiple sites). The core focus of DSL is to support flat-rate, rather than usage-based pricing, and to allow a given service some flexibility in how it allocates the total bandwidth among its VMs.

GateKeeper [30] focuses on providing predictable performance. It attempts to provide each tenant the illusion of a single, non-blocking switch connecting all VMs. Each VM is given guaranteed bandwidth, specified per-VM, into and out of this switch. Furthermore, Gatekeeper uses hypervisor-based rate limits, and a feedback-based mechanism to prevent remote VMs from sending more traffic to a VM than it is allowed to receive.

Oktopus [3] is an implementation of the Virtual Oversubscribed Cluster (VOC) model, using hypervisor-based rate limiters together with an algorithm for placing VMs to meet the requested bandwidth demands (or to reject requests that cannot be met.) Because Oktopus relies on VM placement, it could introduce some delays associated with moving VMs as workloads change.

ConEx [5] is a model which focuses on the amount of congestion that each tenant imposes on the network, based on the intuition that a tenant's network load that does not create congestion does not interfere with other tenants. Tenants purchase congestion-bit-rates, which represent congestion allowances for that tenant. ConEx uses ECN support from switches to measure congestion, and hypervisor rate-limiters to ensure tenants do not cause more congestion than they have purchased. The ConEx proposal requires users to express their requests in terms of congestion allowances instead of bandwidth guarantees, and hence, it could be hard to provide predictable behavior.

SecondNet [9] is a Virtual Data Center (VDC) abstraction with three service models: type-0 service guarantees bandwidth between pairs of VMs; type-1 service provides ingress/egress guarantees for a specific endpoints; other traffic is treated as best-effort. An endpoint can be an entire VM, or a TCP/UDP port on a VM.

4.3 Adapting to Network Heterogeneity

We identify several approaches to adapt Hadoop-based ecosystems to heterogeneous networks [16, 28, 39, 41]. Such work involves either a software-defined networking solution to improve MapReduce scheduling, online profiling of tasks, or speculative task re-execution.

The problem of adapting to network heterogeneity has also been treated for scientific applications [20] and workflows [6, 35]. Such work involves heuristic scheduling techniques and online task profiling, and skewed data distribution for in-memory storage. Furthermore, for cloud-based graph-processing [7], a bandwidth-aware graph partitioning scheme is deployed to improve graph algorithm runtimes.

5 CONCLUSION & FUTURE WORK

Current public clouds suffer from significant network bandwidth variability, due to co-location, congestion and even virtualization overheads. Big data applications are increasingly deployed at massive scale on cloud infrastructure. However, big data processing systems are designed for homogeneous systems (e.g., private clusters), which are not plagued by resource variability.

Therefore, deploying big data applications on clouds raises the problem of unpredictable performance. In this work, we take a step towards understanding the performance of big data workloads on cloud setups with variable network bandwidth. To this end, we emulate eight real-world cloud bandwidth distributions on which

we run six real-world big data workloads. Our main finding is that even though the most impacted workloads are the network-bound ones, the workloads bound on other types of workloads are also slowed down significantly.

For future work we plan to extend the set of experiments to more types of modern big-data workloads, such as Spark SQL and streaming applications. Furthermore, our ambition is to build a performance prediction model for big data applications on bandwidth-variable cloud datacenters and to build a scheduler that takes variability into account to reduce application slowdowns.

REFERENCES

[1] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. 2015. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1383–1394.

[2] Henri Bal, Dick Epema, Cees de Laat, Rob van Nieuwpoort, John Romein, Frank Seinstra, Cees Snoek, and Harry Wijshoff. 2016. A medium-scale distributed system for computer science research: Infrastructure for the long term. *Computer* 49, 5 (2016), 54–63.

[3] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. 2011. Towards predictable datacenter networks. In *ACM SIGCOMM Computer Communication Review*, Vol. 41. ACM, 242–253.

[4] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 267–280.

[5] B. Briscoe and M. Sridharan. 2012. Network Performance Isolation in Data Centres using Congestion Exposure (ConEx). (2012).

[6] Cesar G Chaves, Daniel M Batista, and Nelson LS da Fonseca. 2013. Scheduling cloud applications under uncertain available bandwidth. In *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 3781–3786.

[7] Rishan Chen, Mao Yang, Xuetian Weng, Byron Choi, Bingsheng He, and Xiaoming Li. 2012. Improving large graph processing on partitioned graphs in the cloud. In *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 3.

[8] Nick G Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, Kadangode K Ramakrishnan, and Jacobus E van der Merive. 1999. A flexible model for resource management in virtual private networks. In *ACM SIGCOMM Computer Communication Review*, Vol. 29. ACM, 95–108.

[9] Chuanxiong Guo, Guohan Lu, Helen J Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. 2010. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International COnference*. ACM, 15.

[10] Zhenhua Guo and Geoffrey Fox. 2012. Improving mapreduce performance in heterogeneous network environments and resource utilization. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*. IEEE, 714–716.

[11] Anthony JG Hey, Stewart Tansley, Kristin Michele Tolle, et al. 2009. *The fourth paradigm: data-intensive scientific discovery*. Vol. 1. Microsoft research Redmond, WA.

[12] Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang. 2010. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*. IEEE, 41–51.

[13] Bert Hubert, Thomas Graf, Greg Maxwell, Remco van Mook, Martijn van Oosterhout, P Schroeder, Jasper Spaans, and Pedro Larroy. 2002. Linux advanced routing & traffic control. In *Ottawa Linux Symposium*. 213.

[14] Alexandru Iosup, Nezih Yigitbasi, and Dick Epema. 2011. On the performance variability of production cloud services. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*. IEEE, 104–113.

[15] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. 2009. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 202–208.

[16] Praveenkumar Kondikoppa, Chui-Hui Chiu, Cheng Cui, Lin Xue, and Seung-Jong Park. 2012. Network-aware scheduling of mapreduce framework ondistributed clusters over high speed networks. In *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*. ACM, 39–44.

[17] Donald Kossmann, Tim Kraska, and Simon Loesing. 2010. An evaluation of alternative architectures for transaction processing in the cloud. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 579–590.

[18] Katrina LaCurts, Shuo Deng, Ameesh Goyal, and Hari Balakrishnan. 2013. Choreo: Network-aware task placement for cloud applications. In *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 191–204.

[19] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. 2010. CloudCmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 1–14.

[20] Weiwei Lin, Chen Liang, James Z Wang, and Rajkumar Buyya. 2014. Bandwidth-aware divisible task scheduling for cloud computing. *Software: Practice and Experience* 44, 2 (2014), 163–174.

[21] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. 2016. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research* 17, 1 (2016), 1235–1241.

[22] Jeffrey C Mogul and Lucian Popa. 2012. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Computer Communication Review* 42, 5 (2012), 44–48.

[23] Bogdan Nicolae, Carlos HA Costa, Claudia Misale, Kostas Katrinis, and Yoonho Park. 2017. Leveraging adaptive I/O to optimize collective data shuffling patterns for big data analytics. *IEEE Transactions on Parallel and Distributed Systems* 28, 6 (2017), 1663–1674.

[24] Frank Austin Nothaft, Matt Massie, Timothy Danford, Zhao Zhang, Uri Laserson, Carl Yeksigian, Jey Kottalam, Arun Ahuja, Jeff Hammerbacher, Michael Linderman, et al. 2015. Rethinking data-intensive science using scalable analytics systems. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 631–646.

[25] opennebula 2018. Open Nebula. http://www.opennebula.org. (2018).

[26] Kay Ousterhout, Ryan Rasti, Sylvia Ratnasamy, Scott Shenker, Byung-Gon Chun, and V ICSI. 2015. Making Sense of Performance in Data Analytics Frameworks.. In *NSDI*, Vol. 15. 293–307.

[27] Valerio Persico, Pietro Marchetta, Alessio Botta, and Antonio Pescapé. 2015. Measuring network throughput in the cloud: the case of amazon ec2. *Computer Networks* 93 (2015), 408–422.

[28] Peng Qin, Bin Dai, Benxiong Huang, and Guan Xu. 2015. Bandwidth-aware scheduling with sdn in hadoop: A new trend for big data. *IEEE Systems Journal* (2015).

[29] Barath Raghavan, Kashi Vishwanath, Sriram Ramabhadran, Kenneth Yocum, and Alex C Snoeren. 2007. Cloud control with distributed rate limiting. *ACM SIGCOMM Computer Communication Review* 37, 4 (2007), 337–348.

[30] Henrique Rodrigues, Jose Renato Santos, Yoshio Turner, Paolo Soares, and Dorgival O Guedes. 2011. Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks.. In *WIOV*.

[31] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. 2010. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 460–471.

[32] D Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*. 2503–2511.

[33] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. 2009. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment* 2, 2 (2009), 1626–1629.

[34] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, and Kevin Gibbs. 2006. Iperf. (2006).

[35] Alexandru Uta, Ove Danner, Cas van der Weegen, Ana-Maria Oprescu, Andreea Sandu, Stefania Costache, and Thilo Kielmann. 2017. MemEFS: A network-aware elastic in-memory runtime distributed file system. *Future Generation Computer Systems* (2017).

[36] Guohui Wang and TS Eugene Ng. 2010. The impact of virtualization on network performance of amazon ec2 data center. In *Infocom, 2010 proceedings ieee*. IEEE, 1–9.

[37] Tom White. 2012. *Hadoop: The definitive guide.* " O'Reilly Media, Inc.".

[38] Reynold S Xin, Joseph E Gonzalez, Michael J Franklin, and Ion Stoica. 2013. Graphx: A resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*. ACM, 2.

[39] Lenar Yazdanov, Maxim Gorbunov, and Christof Fetzer. 2015. EHadoop: network I/O aware scheduler for elastic MapReduce cluster. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*. IEEE, 821–828.

[40] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster computing with working sets. *HotCloud* 10, 10-10 (2010), 95.

[41] Matei Zaharia, Andy Konwinski, Anthony D Joseph, Randy H Katz, and Ion Stoica. 2008. Improving MapReduce performance in heterogeneous environments.. In *OSDI*, Vol. 8. 7.