

# Is Big Data Performance Reproducible in Modern Cloud Networks?

Alexandru Uta<sup>1</sup>, Alexandru Custura<sup>1</sup>, Dmitry Duplyakin<sup>2</sup>, Ivo Jimenez<sup>3</sup>, Jan Rellermeyer<sup>4</sup>,  
Carlos Maltzahn<sup>3</sup>, Robert Ricci<sup>2</sup>, Alexandru Iosup<sup>1</sup>

<sup>1</sup>VU Amsterdam   <sup>2</sup>University of Utah   <sup>3</sup>University of California, Santa Cruz   <sup>4</sup>TU Delft

## Abstract

Performance variability has been acknowledged as a problem for over a decade by cloud practitioners and performance engineers. Yet, our survey of top systems conferences reveals that the research community regularly disregards variability when running experiments in the cloud. Focusing on networks, we assess the impact of variability on cloud-based big-data workloads by gathering traces from mainstream commercial clouds and private research clouds. Our dataset consists of millions of datapoints gathered while transferring over 9 petabytes on cloud providers’ networks. We characterize the network variability present in our data and show that, even though commercial cloud providers implement mechanisms for quality-of-service enforcement, variability still occurs, and is even exacerbated by such mechanisms and service provider policies. We show how big-data workloads suffer from significant slowdowns and lack predictability and replicability, even when state-of-the-art experimentation techniques are used. We provide guidelines to reduce the volatility of big data performance, making experiments more repeatable.

## 1 Introduction

Performance variability [13, 47] in the cloud is well-known, and has been studied since the early days [7, 35, 55] of cloud computing. Cloud performance variability impacts not only operational concerns, such as cost and predictability [14, 42], but also reproducible experiment design [3, 10, 31, 47].

Big data is now highly embedded in the cloud: for example, Hadoop [64] and Spark [65] processing engines have been deployed for many years on on-demand resources. One key issue when running big data workloads in the cloud is that, due to the multi-tenant nature of clouds, applications see performance effects from other tenants, and are thus susceptible to performance variability, including on the network. Even though recent evidence [50] suggests that there are limited potential gains from *speeding up* the network, it is still the case that variable network performance can *slow down* big data

systems and introduce volatility that makes it more difficult to draw reliable scientific conclusions.

Although cloud performance variability has been thoroughly studied, the resulting work has mostly been in the context of optimizing tail latency [22], with the aim of providing more consistent application-level performance [15, 25, 29, 57]. This is subtly—but importantly—different from understanding the ways that fine-grained, resource-level variability affects the *performance evaluation* of these systems. Application-level effects are especially elusive for complex applications, such as big data, which are not bottlenecked on a specific resource for their entire runtime. As a result, it is difficult for experimenters to understand how to design experiments that lead to reliable conclusions about application performance under variable network conditions.

Modern cloud data centers increasingly rely on software-defined networking to offer flows between VMs with reliable and predictable performance [48]. While modern cloud networks generally promise isolation and predictability [7, 30], in this paper we uncover that they rarely achieve stable performance. Even the mechanisms and policies employed by cloud providers for offering quality of service (QoS) and fairness can result in non-trivial interactions with the user applications, which leads to performance variability.

Although scientists are generally aware of the relationship between repeated experiments and increased confidence in results, the specific strength of these effects, their underlying causes, and methods for improving experiment designs have not been carefully studied in the context of performance experiments run in clouds. Variability has a significant impact on sound experiment design and result reporting [31]. In the presence of variability, large numbers of experiment repetitions must be performed to achieve tight confidence intervals [47]. Although practitioners and performance engineers acknowledge this phenomenon [7, 35, 55], in practice these effects are frequently disregarded in performance studies.

Building on our vision [34], and recognizing the trend of the academic community’s increasing use of the cloud for computing resources [53], we challenge the current state-of-

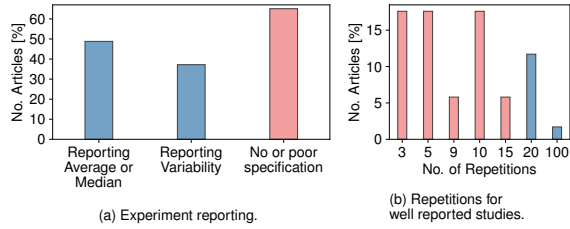


Figure 1: State-of-practice in big data articles with cloud experiments: (a) Aspects reported about experiments. Bars represent aspects that are not mutually exclusive, thus the total can exceed 100%. (b) Number of experiment repetitions performed for the properly specified articles.

practice in cloud-based systems experimentation and advocate for sound experiment design and result reporting. We show that, due to performance variability, flawed cloud-based experimentation could lead to inaccurate or even wrong conclusions. We show, in-depth, the performance implications of network variability when running big data workloads. The interplay between underlying resources and applications is complex, and leads to non-trivial performance behavior.

To characterize such interactions, we run state-of-the-art, real-world applications using Apache Spark [4]. We run big-data workloads either directly on real-world mainstream clouds, or by emulating the network behavior of such clouds. Our results show that variability highly impacts not only performance, but also credible and reproducible experimentation.

Addressing cloud users, performance engineers, and system designers, we examine the implications of *network* variability on big data, and present our main findings and contributions:

1. **Lack of sound experimentation:** Many articles in the literature that present cloud-based experiments are either under-specified (i.e., do not report statistical measures), or run inconclusive numbers of experiment repetitions (Section 2).
2. **Variability in modern cloud networks:** We conduct and analyze measurements of public and private cloud providers, characterize the level of variability, and identify specific sources (Section 3).
3. **Network variability impact on application performance reproducibility:** Low-level network variability can have significant effects on application performance, and can violate assumptions commonly used in performance modeling (such as that experiment runs are independent and identically distributed) (Section 4).
4. **Strategies for running reproducible experiments:** Given our measurement and experience with application-level benchmarks, we make recommendations for improving the reliability and reproducibility of experiments (Section 5).

## 2 Is Cloud Variability Disregarded?

We perform a literature survey to uncover whether and how researchers and practitioners take cloud performance variability into account when running experiments. Our findings are

Table 1: Parameters for the performance variability literature survey. We manually select only the articles with empirical evaluations performed using clouds.

Venues	Keywords	Years
NSDI, OSDI SOSP, SC	big data, streaming, Hadoop, MapReduce, Spark, data storage graph processing, data analytics	2008 - 2018

Table 2: Survey process. Initial filtering done automatically by keywords, then manually for cloud-based experiments. The resulting subset is significant and highly-cited.

Articles Total	Filtered Automatically by Keywords	Filtered Manually for Cloud Experiments	Citations for selected 44 articles
1,867	138	44 (15 NSDI, 7 OSDI, 7 SOSP, 15 SC)	11,203

depicted in Figure 1 and summarized as follows:

**Finding 2.1** Cloud performance variability is largely disregarded when researchers evaluate and prototype distributed systems, or compare established systems.

**F2.2** Most cloud performance studies are under-specified. Most studies: (i) do not specify which performance measures are reported (i.e., mean, median); (ii) do not report minimal statistical variation data (i.e., standard deviation, quartiles); (iii) do not report the number of repetitions of an experiment.

**F2.3** Most cloud performance evaluations are poorly designed: a large majority of such studies only perform small numbers of experiment repetitions (i.e., 3-10 trials), and do not assess variability or confidence.

Over the last decade, big data platforms and applications have been co-evolving with the cloud. This allowed researchers and practitioners to develop, deploy, and evaluate their applications and systems on various virtualized infrastructures. There is much evidence that clouds suffer from performance variability [7, 13, 35, 47]. It is therefore intuitive to ask if practitioners and system designers take variability into account when designing experiments or building systems. To answer these questions, we performed a systematic literature survey covering prominent conferences in the field: NSDI [44], OSDI [5], SOSP [1], and SC [2].

**Survey Methodology:** Table 1 shows the parameters of our survey, and Table 2 presents our survey process in-depth: (1) we started with all articles published in the aforementioned venues; (2) selected automatically a subset, based on string matching our query on keywords, title, and abstract; (3) we manually selected the articles in which the experiments were performed on a public cloud. The 44 selected articles are highly influential, having been cited **11,203** times so far<sup>1</sup>.

The criteria we looked for when analyzing such articles

<sup>1</sup> according to Google Scholar on May 20, 2019

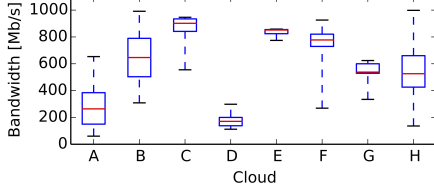


Figure 2: Bandwidth distributions for eight real-world clouds. Box-and-whiskers plots show the 1st, 25th, 50th, 75th, and 99th percentiles. (Distributions derived from the study [7] conducted by Ballani et al.)

are the following: (i) reporting average or median metrics over a number of experiments; (ii) reporting variability (such as standard deviation or percentiles) or confidence (such as confidence intervals); (iii) reporting the number of times an experiment was repeated. These are all critical criteria for determining whether a study’s conclusions may be irreproducible, or worse, not fully supported by the evidence (i.e., flawed). To check the reliability of our manual filtering, it was performed by two separate reviewers, and we applied Cohen’s Kappa coefficient [16] for each category presented in Figure 1a: reporting average or median, statistics, and poor specification. Our Kappa scores for each category, were 0.95, 0.81, and 0.85, respectively. Values larger than 0.8 are interpreted as near-perfect agreement between the reviewers [61].

**Survey Results:** *The systems community centered around cloud computing and big data disregards performance variability when performing empirical evaluations in the cloud.* Figure 1 shows the results of our survey. Out of the two reviewer’s scores, we plot the lower scores, i.e., ones that are more favorable to the articles. We found that over 60% of the surveyed articles are under-specified (i.e., the authors do not mention how many times they repeated the experiments or even whether they are reporting average, median, etc.); a subset of the articles report averages or medians, but out of those, only 37% report variance or confidence (i.e., error-bars, percentiles). We further found that most articles that do report repetitions perform only 3, 5 or 10 repetitions of the experiments. The reason for such practices might be that experimenters are more used to evaluating software in controlled environments—what is true in controlled environments often does not hold in clouds.

Moreover, 76% of the properly specified studies use no more than 15 repetitions. Coupled with the effects of cloud variability, such experiment design practices could lead to wrong or ambiguous conclusions, as we show next.

## 2.1 How credible are experiments with few repetitions?

Experiments with few repetitions run the risk of reporting inaccurate results; the higher the variability, the greater the risk that a low-repetition experiment’s results are unreliable. We

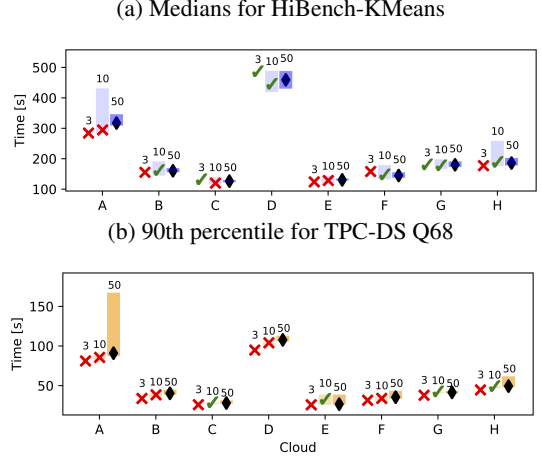


Figure 3: Medians and 90th percentiles for K-Means (a) and TPC-DS Q68 (b). Estimates are shown along with their 95% confidence intervals (CIs) for performance measurements under the A-H distributions.  $\diamond$  depicts estimates 50-runs. Judged by the 50-run CIs we consider *gold standard*, accurate estimates (inside those CIs) are  $\checkmark$ ; inaccurate estimates (outside those CIs) are  $\times$  for 3- and 10-run sets.

use application-level benchmarks to show how the bandwidth distributions found by Ballani et al. [7] for eight real-world clouds—shown in Figure 2—do affect findings in practice.

We emulate the behavior of the eight clouds presented in Figure 2, which were contemporary with most articles found in our survey. In a private Spark [4] cluster of 16 machines, we limit the bandwidth achieved by machines according to distributions A – H. We uniformly sample bandwidth values from these distributions every  $x \in \{5, 50\}$  seconds. We used 50 experiment repetitions as our “gold standard” to demonstrate the intuition that running more experiments yields more accurate results, and compared them to the 3- and 10-repetitions commonly found in our literature survey. (In Section 4 we propose better methods for experiment design.)

**Emulation Results:** We found that experiments with few repetitions often produced medians that are *outside of* the 95% confidence intervals (CIs) for larger experiment sequences. The 95% CIs for medians represent ranges in which we would find true medians with 95% probability, if we were able to run infinite repetitions. Thus, when the low-repetition medians lie outside of the high-repetition CIs, there is a 95% probability that the former are inaccurate. This can be seen in Figure 3, which plots estimates of 95% nonparametric (asymmetric) CIs [11] for experiments using bandwidth distributions A – H from Figure 2. For each bandwidth distribution, we show the medians and CIs for 3-, 5-, and 50-repetition experiments.<sup>2</sup> The median for the “gold standard” experiment is marked with a diamond; medians for lower-repetition experiments are shown with an “X” if outside the gold-standard 95% CI, or a check-mark if within it.

<sup>2</sup>Three repetitions are insufficient to calculate CIs; we plot medians because this is representative of what is often found in the literature.

The top of Figure 3 (part (a)) shows our estimates of *medians* for the K-Means application from HiBench [32]. Of the eight cloud bandwidth distributions, the 3-run median falls outside of the gold-standard CI for six of them (75%), and the 10-run median for three (38%). The bottom half of Figure 3 (part (b)) shows the same type of analysis, but this time, for tail performance [22] instead of the median. To obtain these results, we used TPC-DS [49] Query-68 measurements and the method from Le Boudec [11] to calculate nonparametric estimates for the 90th percentile performance, as well as their confidence bounds. As can be seen in this figure, it is even more difficult to get robust *tail* performance estimates.

**Emulation Methodology:** The quartiles in Ballani’s study (Figure 2) give us only a rough idea about the probability densities and there is uncertainty about fluctuations, as there is no data about sample-to-sample variability. Considering that the referenced studies reveal no autocovariance information, we are left with using the available information to sample bandwidth uniformly. Regarding the sampling rate, we found the following: (1) As shown in Section 3 two out of the three clouds we measured exhibits significant sample-to-sample variability on the order of tens of seconds; (2) The cases F-G from Ballani’s study support fine sampling rates: variability at sub-second scales [63] and at the 20s intervals [24] is significant. Therefore, we sample at relatively fine-grained intervals: 5s for Figure 3(a), and 50s for Figure 3(b). Furthermore, sampling at these two different rates shows that benchmark volatility is not dependent on the sampling rate, but rather on the distribution itself.

### 3 How Variable Are Cloud Networks?

We now gather and analyze network variability data for three different clouds: two large-scale commercial clouds, and a smaller-scale private research cloud. Our main findings can be summarized as follows:

**F3.1** Commercial clouds implement various mechanisms and policies for network performance QoS enforcement, and these policies are opaque to users and vary over time. We found (i) token-bucket approaches, where bandwidth is cut by an order of magnitude after several minutes of transfer; (ii) a per-core bandwidth QoS, prioritizing heavy flows; (iii) instance types that, when created repeatedly, are given different bandwidth policies unpredictably.

**F3.2** Private clouds can exhibit more variability than public commercial clouds. Such systems are orders of magnitude smaller than public clouds (in both resources and clients), meaning that when competing traffic does occur, there is less statistical multiplexing to “smooth out” variation.

**F3.3** Base latency levels can vary by a factor of almost 10x between clouds, and implementation choices in the cloud’s virtual network layer can cause latency variations over two orders of magnitude depending on the details of the application.

Table 3: Experiment summary for determining performance variability in modern cloud networks. Experiments marked with a star (\*) are presented in depth in this article. Due to space limitations, we release the other data in our repository [59]. All Amazon EC2 instance types are typical offerings of a big data processing company [20].

Cloud	Instance Type	QoS (Gbps)	Exp. Duration	Exhibits Variability	Cost (\$)
*Amazon	c5.XL	$\leq 10$	3 weeks	Yes	171
Amazon	m5.XL	$\leq 10$	3 weeks	Yes	193
Amazon	c5.9XL	10	1 day	Yes	73
Amazon	m4.16XL	20	1 day	Yes	153
Google	1 core	2	3 weeks	Yes	34
Google	2 core	4	3 weeks	Yes	67
Google	4 core	8	3 weeks	Yes	135
*Google	8 core	16	3 weeks	Yes	269
HPCCloud	2 core	N/A	1 week	Yes	N/A
HPCCloud	4 core	N/A	1 week	Yes	N/A
*HPCCloud	8 core	N/A	1 week	Yes	N/A

### 3.1 Bandwidth

We run our bandwidth measurements in two prominent commercial clouds, Amazon EC2 (us-east region) and Google Cloud (us-east region), and one private research cloud, HPC-Cloud<sup>3</sup>. Table 3 summarizes our experiments. In the interest of space, in this paper we focus on three experiments; all data we collected is available in our repository [59]. We collected the data between October 2018 and February 2019. In total, we have over 21 weeks of nearly-continuous data transfers, which amount for over 1 million datapoints and over 9 petabytes of transferred data.

The Amazon instances we chose are typical instance types that a cloud-based big data company offers to its customers [20], and these instances have AWS’s “enhanced networking capabilities” [6]. On Google Cloud (GCE), we chose the instance types that were as close as possible (though not identical) to the Amazon EC2 offerings. HPCCloud offered a more limited set of instance types. We limit our study to this set of cloud resources and their network offerings, as big data frameworks are not equipped to make use of more advanced networking features (i.e., InfiniBand), as they are generally designed for commodity hardware. Moreover, vanilla Spark deployments, using typical data formats such as Parquet or Avro, are not able to routinely exploit links faster than 10 Gbps, unless significant optimization is performed [58]. Therefore, the results we present in this article are highly likely to occur in real-world scenarios.

In the studied clouds, for each pair of VMs of similar instance types, we measured bandwidth continuously for one week. Since big data workloads have different network access patterns, we tested multiple scenarios:

- **full-speed** - continuously transferring data, and summarizing performance metrics (bandwidth, retransmis-

<sup>3</sup><https://userinfo.surfsara.nl/systems/hpc-cloud>



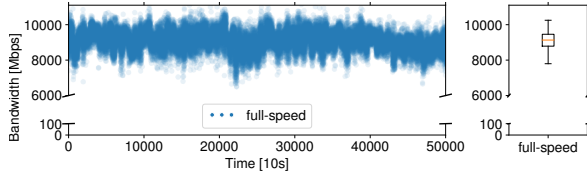


Figure 4: Variable network bandwidth performance in the HPCCloud (left); the statistical performance distribution, plotted as an IQR box; the whiskers represent 1st and 99th percentiles (right). Duration: a week of continuous experimentation; each point is average over 10 seconds.

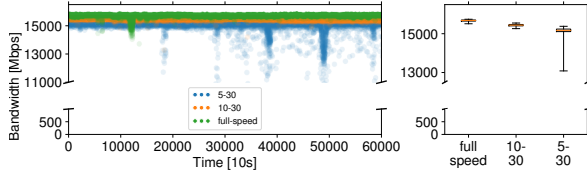


Figure 5: Variable network bandwidth performance in the Google Cloud (left), and the statistical performance distribution, plotted as an IQR box, where the whiskers are 1st and 99th percentiles (right). The duration is a week of continuous experimentation, each point is an average over 10 seconds.

sions, CPU load etc.) every 10 seconds;

- **10-30** - transfer data 10 seconds, wait 30 seconds;
- **5-30** - transfer data 5 seconds, wait 30 seconds.

The first transmission regime models highly network intensive applications, such as long-running batch processing or streaming. The last two modes mimic short-lived analytics queries, such as TPC-H, or TPC-DS.

**HPCCloud.** Small-scale (i.e., up to 100 physical machines and several hundred users) private (research) clouds often do not use mechanisms to enforce network QoS. We measured the network performance variability between pairs of VMs, each having 8 cores. Figure 4 plots the results. We show our measurements only for "full-speed" (i.e., continuous communication) because our other experiments show similar behavior. We observe that the network bandwidth shows high variability, ranging from 7.7 Gbps to 10.4 Gbps.

**Google Cloud.** GCE states that it enforces network bandwidth QoS by guaranteeing a "per-core" amount of bandwidth. Our measurements, plotted in Figure 5, fall close to the QoS reported by the provider, but access pattern affects variability to a greater degree than in other clouds. Longer streams (*full-speed*) exhibit low variability and better overall performance, while *5-30* has a long tail. This could be due to the design of the Google Cloud network, where idle flows use dedicated gateways for routing through the virtual network [18]. We observe that network bandwidth varies significantly, depending on access patterns, between 13 Gbps and 15.8 Gbps.

**Amazon EC2.** We discover the opposite behavior in EC2: heavier streams achieve less performance and more variability compared to lighter (shorter) streams, as shown in Figure 6. Considering the large performance differences between

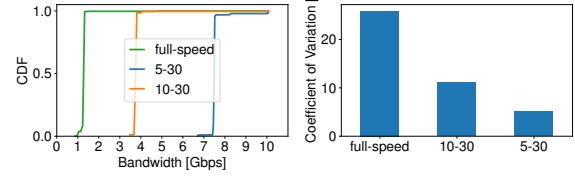


Figure 6: Variable network bandwidth performance in Amazon EC2, plotted as an empirical cumulative distribution (left), barplot of the coefficient of variation (right). The duration is a week of continuous experimentation, each data point representing an average over 10 seconds.

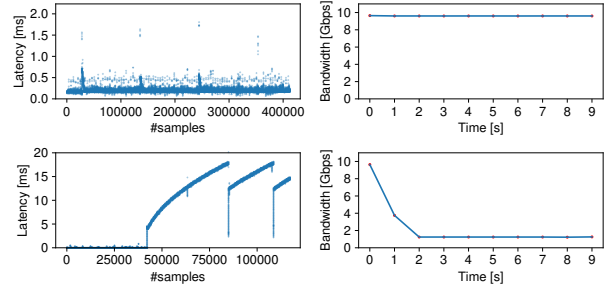


Figure 7: Example of observed Amazon EC2 latency for a 10-second TCP sample on *c5.xlarge*. Left: RTT latency for TCP packets. Right: achieved *iperf* bandwidth. Top: regular Amazon EC2 behavior. Bottom: latency behavior when a drop in bandwidth occurs.

these experiments, we plot our measurements as a CDF and a barplot of coefficient of variation to improve visibility. There are approximately 3x and 7x slowdowns between *10-30* and *5-30* and *full-speed*, respectively. The achieved bandwidth varies between 1 Gbps and 10 Gbps. We investigate the causes of this behavior in Section 3.3.

**How rapidly does bandwidth vary?** Our analysis shows the level of *measurement-to-measurement* variability is significant: bandwidth in HPCCloud (*full-speed*) and Google Cloud (*5-30*) varies between consecutive 10-second measurements up to 33% and 114%, respectively. While a small sample may exhibit only modest fluctuations, the long-tailed distributions we observed here strongly suggest using the analysis techniques we discuss in Section 4.1. Amazon EC2's variability is more particular and policy-dependent (Section 3.3).

## 3.2 Latency

Commercial clouds implement their virtual networks using very different mechanisms and policies. We can see this in more detail by looking at the round-trip latencies seen in Google Cloud and Amazon EC2. We measure the application-observed TCP RTT, as this is what impacts the high-level networking stacks of big data frameworks. For our experiments, we run 10-second streams of *iperf* tests, capturing all packet headers with *tcpdump*. We perform an offline analysis of the packet dumps using *wireshark*, which compares the

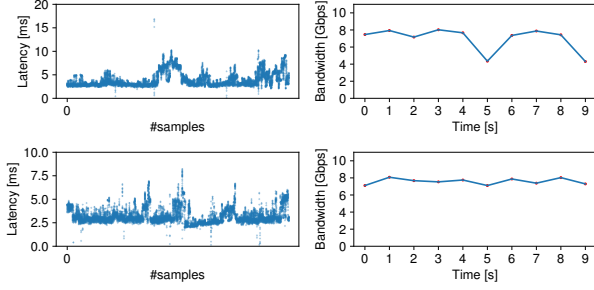


Figure 8: Example of observed Google Cloud latency for a 10-second TCP sample on a 4-core instance. Left: RTT latency for TCP packets. Right: achieved *iperf* bandwidth.

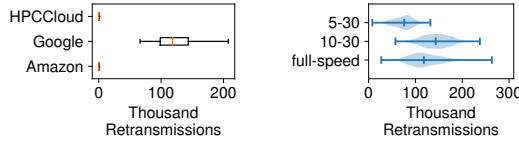


Figure 9: TCP retransmission analysis, summarized for all experiments presented before, in all clouds. Left: retransmissions as IQR boxplots, with the whiskers representing 1st and 99th percentiles; Right: violin plot for retransmissions in Google Cloud; thickness of the plot is proportional to the probability density of the data.

time between when a TCP segment is sent to the (virtual) network device and when it is acknowledged. Our data was collected between August and September 2019. In total, it contains over 50 million RTT datapoints.

The behavior we observe is inherently different: Google Cloud exhibits latency in the order of milliseconds, with an upper limit of 10ms. Amazon EC2 generally exhibits faster sub-millisecond latency under typical conditions, but when the traffic shaping mechanism (detailed in Section 3.3) takes effect, the latency increases by two orders of magnitude, suggesting large queues in the virtual device driver. Figure 7 shows representative patterns of latency in the Amazon EC2 cloud, while Figure 8 is representative of Google Cloud. Both figures plot latency as RTT packet data obtained from a 10-

second TCP stream obtained running an *iperf* benchmark.

The behavior observed in the top half of Figure 7 lasts for approximately ten minutes of full-speed transfer on *c5.xlarge* instances. After this time, the VMs’ bandwidth gets throttled down to about 1 Gbps (bottom half of Figure 7), which also significantly increases latency. On Google Cloud, there is no throttling effect, but the bandwidth and latency vary more from sample to sample.

### 3.3 Identifying Mechanisms and Policies

The behavior exhibited by the two commercial providers is notably different. We uncover mechanisms and policies for enforcing client QoS by performing extra analysis, depicted in Figures 9 and 10. The former plots the number of retransmissions per experiment (part (a)) and a zoomed-in view of Google Cloud (part (b)). Amazon EC2 and HPCCloud have a negligible number of retransmissions, yet retransmission are common in Google Cloud: roughly 2% per experiment.

Figure 10 plots the total amount of traffic for Amazon EC2 and Google Cloud over the entire duration of our experiments. It is clear that in Google Cloud’s case the amount of traffic generated by *full-speed* is orders of magnitude larger than for the intermittent access patterns. In Amazon EC2’s case, the total amount of data sent for all three kinds of experiments is roughly equal. By corroborating this finding the more fine-grained experiments we performed presented in Figure 7, and other empirical studies [51, 62], we find that Amazon EC2 uses a *token-bucket* algorithm to allocate bandwidth to users.

**Token-Bucket Analysis.** The token-bucket algorithm operation can be explained as follows. When a VM is provided to the user, its associated *bucket* holds a certain amount of tokens (i.e., a budget). This budget is allowed to be spent at a high rate (i.e., 10 Gbps). When the budget is depleted (e.g., after about 10 minutes of continuous transfer on a *c5.xlarge* instance, the QoS is limited to a low rate (e.g., 1 Gbps). The bucket is also subject to a replenishing rate that we empirically found to be approximately 1 Gbit token per second, i.e., every second users receive the amount of tokens needed to send 1 Gbit of data at the high (10 Gbps) rate. Once the token bucket empties, transmission at the capped rate is sufficient to keep it from filling back up. The user must *rest* the network, and re-filling the bucket completely takes several minutes.

We analyze the behavior of multiple types of VMs from the *c5.\** family, and find that their token-bucket parameters differ. More expensive machines benefit from larger initial budgets, as well as higher bandwidths when their budget depletes. Figure 11 plots the token-bucket parameter analysis for four VMs of the *c5.\** family. For each VM type, we ran an *iperf* test continuously until the achieved bandwidth dropped significantly and stabilized at a lower value. For each instance type, we ran 15 tests. Figure 11 shows the time taken to empty the token bucket, the *high* (non-empty bucket) bandwidth value, and the *low* (empty bucket) bandwidth value. As the *size* (i.e., number

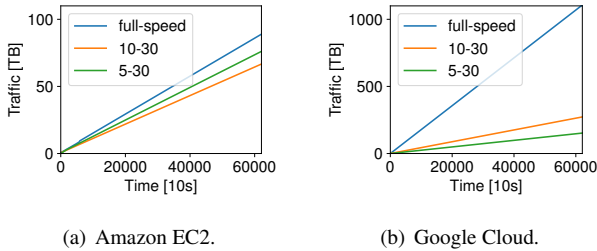


Figure 10: The total amount of data transferred between the pairs of virtual machines involved in the three types of experiments performed. The total time is a week, while each point on the horizontal axis represents 10 seconds.

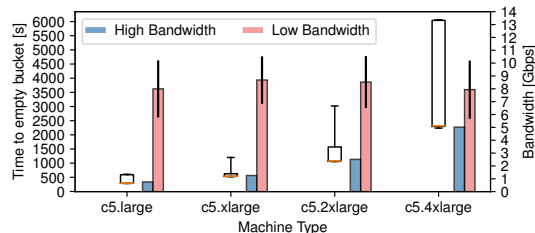


Figure 11: The token-bucket parameters identified for several instances of Amazon EC2 *c5.\** family. The elapsed time to empty the token bucket is depicted with boxplots associated with *left* vertical axis. The *high* and *low* bandwidths of the token bucket are depicted with bar plots with whiskers and are associated with the *right* vertical axis.

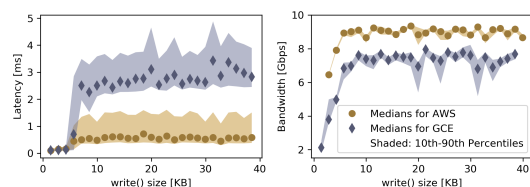


Figure 12: Measured latency and bandwidth for Amazon EC2 (*c5.xlarge*) and GCE (4-core VM with advertised 8 *Gbps*) instances as functions of the `write()` size.

of cores, amount of memory etc.) of the VM increases, we notice that the *bucket size* and the *low* bandwidth increase proportionally. However, as the magnitude of the boxplots suggests, as well as the error bars we plotted for the *high* bandwidth, these parameters are not always consistent for multiple incarnations of the same instance type.

**Virtual NIC Implementations.** We found that differences in EC2 and GCE’s implementations of virtual NICs can lead to significantly different observed behavior. EC2’s virtual NICs advertise an MTU of 9000 bytes, a standard “jumbo frame” size. GCE’s only advertise an MTU of 1500 bytes (standard Ethernet frame size), but instead enable TCP Segmentation Offloading (TSO), in which the NIC accepts larger “packets” from the device driver, but then breaks them down into smaller Ethernet frames before transmission (we do not know whether this occurs at the virtual or physical NIC in GCE’s implementation). Both of these techniques serve the same basic function—reducing overhead by sending fewer, larger packets on the virtual NIC, but result in different observable behavior on the host, and the details of this behavior depend heavily on the application and workload.

The most striking effect is the way that the size of the `write()`s done by the application affects latency and packet retransmission. Figure 12 plots the effects of the `write()` size on latency and bandwidth. On EC2, the size of a single “packet” tops out at the MTU of 9K, whereas on GCE, TSO can result in single “packet” at the virtual NIC being as large as 64K in our experiments. With such large “packets,” per-

ceived latency increases greatly due to the higher perceived “transmission time” for these large packets. The number of retransmissions also goes up greatly, presumably due to limited buffer space in the bottom half of the virtual NIC driver or tighter bursts on the physical NIC. In practice, the size of the “packets” passed to the virtual NIC in Linux tends to equal to the `write` on the socket (up to the cap noted above). This makes the observed behavior—and thus repeatability and the ability to generalize results between clouds—highly application-dependent. It is also worth noting that all streams are affected when one stream sends large “packets”, since they share a queue in the virtual device driver. On GCE, when we limited our benchmarks to `writes` of 9K, we got near-zero packet retransmission and an average RTT of about 2.3ms. When the benchmark used its default `write()` size of 128K, we saw the hundreds of thousands of retransmission shown in Figure 9 and latencies as high as 10ms.

## 4 Performance Reproducibility For Big Data Applications

Having looked at low-level variability in bandwidth and latency, we now move “up” a level to applications and workloads. Our main findings are:

**F4.1** Under variability resembling Google Cloud and HPC-Cloud, which can be modeled as stochastic noise, reproducible experiments can be obtained using sufficient repetitions and sound statistical analyses.

**F4.2** Application transfer patterns exhibit non-trivial interactions with token-bucket network traffic shapers. Depending on the bucket budget and the application, significant application performance variability is incurred.

**F4.3** Token-bucket traffic shapers in conjunction with (imbalanced) big data applications can create stragglers.

**F4.4** In long-running cloud deployments that have incurred large amounts of varied network traffic, it is highly difficult to predict application performance, as it is dependent on the state of the individual nodes’ remaining token-bucket budgets.

**Big Data Workloads.** In this section, we run the Hi-Bench [32] and TPC-DS [49] benchmarks on Spark 2.4.0 (see Table 4) to showcase our main findings on network variability and big data workloads reproducibility. In 2015, Ousterhout et al. [50] found that big data workloads are mostly CPU bound. The workloads we chose here are no exception. However, they are *sensitive* to oscillations in the network transfer performance. Moreover, most of the CPU load in [50] is attributed to the framework’s inefficiencies [19], which have been solved in later releases. As a consequence, modern Spark implementations are more sensitive to network variations.

### 4.1 Experiments and Stochastic Noise

As detailed in Section 3, the behavior of network performance variability for Google Cloud and HPCcloud is closer in na-

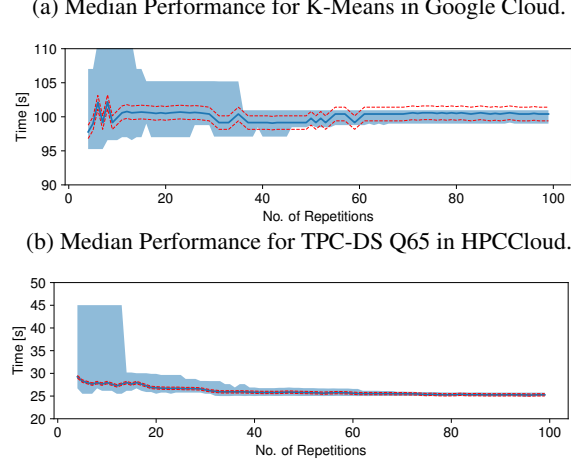


Figure 13: CONFIRM analysis for K-Means and TPC-DS Q65 on Google Cloud and HPCCloud. Median estimates (blue thick curve), 95% nonparametric confidence intervals (light blue filled space), and 1% error bounds (red dotted curves). Vertical axis not starting at 0 for visibility.

Table 4: Big data experiments on modern cloud networks.

Workload	Size	Network	Software	#Nodes
HiBench [32]	BigData	Token-bucket, Figure 14	Spark 2.4.0, Hadoop 2.7.3	12
TPC-DS [49]	SF-2000	Token-bucket, Figure 14	Spark 2.4.0, Hadoop 2.7.3	12

ture to stochastic variability given by transient conditions in the underlying resources, such as noisy neighbors. To achieve reproducible experiments under such conditions, system designers and experimenters need to carefully craft and plan their tests, using multiple repetitions, and must perform sound statistical analyses.

We ran several HiBench [32] and TPC-DS [49] benchmarks directly on the Google Cloud and HPCCloud clouds and report how many repetitions an experimenter needs to perform in order to achieve trustworthy experiments. While it is true that running experiments directly on these clouds we cannot differentiate the effects of network variability from other sources of variability, the main take-away message of this type of experiment is that this kind of stochastic variability can be accounted for with proper experimentation techniques.

On the performance data we obtained, we performed a CONFIRM [47] analysis to predict how many repetitions an experiment will require to achieve a desired confidence interval. Figure 13 presents our findings, showing that for these two common benchmarks, it can take 70 repetitions or more to achieve 95% confidence intervals within 1% of the measured median. As we saw in Section 2, this is far more repetitions than are commonly found in the literature: most papers are on the extreme left side of this figure, where the confidence intervals are quite wide. This points to the need for stronger experiment design and analysis in our community.

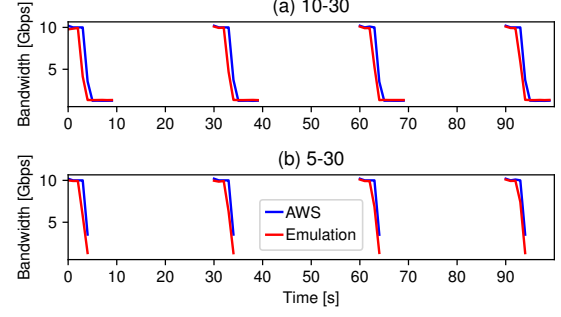


Figure 14: Validation of the emulation of the token-bucket policy of Amazon EC2. The similar aspect of the two curves indicates that emulation is high-quality.

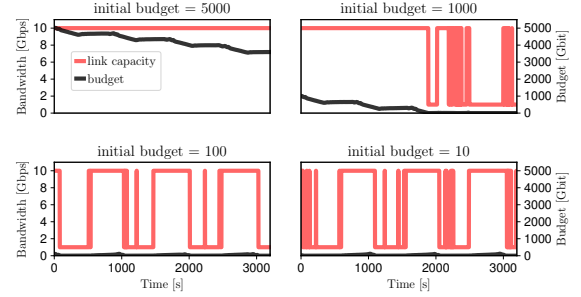


Figure 15: Link capacity allocated when running Terasort on a token bucket. Left vertical axis shows the link capacity; right vertical axis shows the token bucket budget. Budget depletes due to application network transfers.

## 4.2 Experiments and Token-Buckets

In contrast to Google Cloud and HPCCloud, the *token-bucket* shaping policy of Amazon EC2 is *not* stochastic noise, and needs in-depth analysis. Because token-bucket behavior is dependent on past network access patterns, *an application influences not only its own runtime, but also future applications' runtimes*.

**Token-bucket Emulator.** We decided to emulate the behavior of Amazon EC2 token-bucket instead of directly running applications in this cloud. We believe this type of experimentation is superior to the other two alternatives: (i) simulation, or (ii) directly running applications on the cloud. For the former, we believe the behavior of big data applications under network performance variability is far too subtle and complex to properly simulate while modeling and capturing all possible variables. For the latter, we perform the emulation in an isolated setup, i.e., a private cluster, that does not share resources. This allows us to test in isolation the effects of network performance variability, excluding as much as possible all other sources of variability one could encounter in a cloud (e.g., CPU, memory bandwidth, I/O etc.). If we were to directly run applications in a cloud, it would have been difficult to separate the effects of network variability



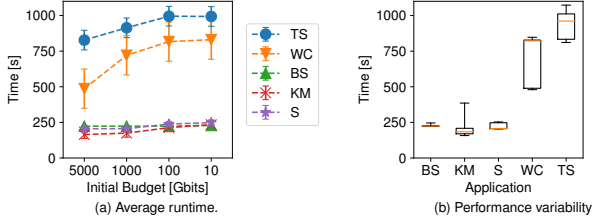


Figure 16: HiBench average runtime (left) and performance variability (right), plotted as IQR box (whiskers represent 1st and 99th percentiles), induced by token bucket budget variability. The more network-dependent applications are affected more by lower budgets.

from, for example, the effects of CPU variability.

We built a network emulator based on the Linux `tc` [33] facility. Figure 14 plots the real-world behavior encountered in Amazon EC2 in comparison with our emulation. This experiment is a zoomed-in view of the experiment in Section 3.1, where our servers were communicating for either five or ten seconds, then slept for 30 seconds. At the beginning of each experiment, we made sure that the token-bucket budget is nearly empty. During the first few seconds of the experiment the token-bucket budget gets completely exhausted. For each sending phase of 5 or 10 seconds, the system starts at a high QoS (10 Gbps bandwidth), after a few seconds the budget is emptied, and the system drops to a low QoS (1 Gbps).

**Experiment Setup.** We perform the experiments described in Table 4 on a 12-node cluster. Each node has 16 cores, 64GB memory, a 256GB SSD, and FDR InfiniBand network. Using the emulator presented in Figure 14, we run on the emulated Amazon EC2 token-bucket policy all applications and queries in the HiBench [32] and TPC-DS [49] benchmark suites. The emulated setup is that of the *c5.xlarge* instance type, which typically sees a high bandwidth of 10 Gbps and a low bandwidth of 1 Gbps. Throughout our experiments we vary the token bucket budget to assess its impact on big data applications. We run each workload a minimum of 10 times for each token-bucket configuration and report full statistical distributions of our experiments.

**Token-bucket-induced Performance Variability.** One important parameter for the token-bucket is its budget: the number of tokens available at a certain moment in time. This is highly dependent on the previous state of the virtual machine (i.e., how much network traffic has it sent recently), and has a large impact on the performance of future deployed applications. Note that it is difficult to estimate the currently-available budget for anything other than a “fresh” set of VMs: each VM has its own token bucket, the remaining budget is a function of previous runs, and, as we saw in Figure 11 the constants controlling the bucket are not always identical.

Application performance is highly dependent on the budget, and deployments with smaller budgets create more network performance variability. Figure 15 shows the network traf-

fic behavior of the Terasort application with different initial budgets. For each budget, the subfigures show the application network profile for 5 consecutive runs. We notice a strong correlation between small budgets and network performance variability: there is much more variability for budgets  $\in \{10, 100\}$  Gbits, than for budgets  $\in \{1000, 5000\}$  Gbits.

Figure 16 shows how this effect manifests in the runtimes of HiBench: it plots the average application runtime (left) over 10 runs for budgets  $\in \{10, 100, 1000, 5000\}$  Gbits, and the performance variability over the same budgets (right). For the more network-intensive applications (i.e., TS, WC), the initial state of the budget can have a 25%–50% impact on performance.

A similar behavior is observed for the TPC-DS benchmark suite. Figure 17 shows the query sensitivity to the token budget and the variability induced by different budget levels. Figure 17(a) plots average runtime slowdown for 10-run sets of TPC-DS queries for budgets  $\in \{10, 100, 1000\}$  Gbits, compared to the 5000 Gbit budget. For all queries, larger budgets lead to better performance. Figure 17(b) plots the performance variability over all tested budgets. Queries with higher network demands exhibit more sensitivity to the budget and hence higher performance variability.

These results clearly show that if the system is left in an unknown state (e.g., a partially-full token bucket, left over from previous experiments), the result is likely to be an inaccurate performance estimate. Evidence from Figures 16(b) and 17(b) strongly supports this, as performance varies widely for the network-intensive queries and applications depending on the token-bucket budget.

**Token-bucket-induced Stragglers.** Non-trivial combinations of token-bucket budgets, application scheduling imbalances, and network access patterns lead to straggler nodes. Figure 18 shows that for budget = 2500 Gbits and application TPC-DS, the application gets slowed down by a straggler: all nodes but one in the deployment do not deplete their budgets completely, thus remaining at a high bandwidth QoS of 10 Gbps. However, there is one node on which the token-bucket budget is depleted, causing its bandwidth to get limited to 1 Gbps. Exacerbating the variability, the behavior is not consistent: this node oscillates between high and low bandwidths in short periods of time. Such unpredictable behavior leads to both performance variability of the entire setup and also poor experiment reproducibility. This behavior will be prevalent in many *unbalanced* networked applications, where certain servers might perform more transfers than others. Especially in long-running clusters, the state of the individual servers’ token-buckets will be highly different. As a direct consequence, the overall system will suffer from stragglers.

**Repeatable experiments and token-buckets.** Token-bucket policies for enforcing network QoS can have unexpected and detrimental impacts on sound cloud-based experimentation. To explore this, we compute medians and their nonparametric confidence intervals (CIs), similar to the work

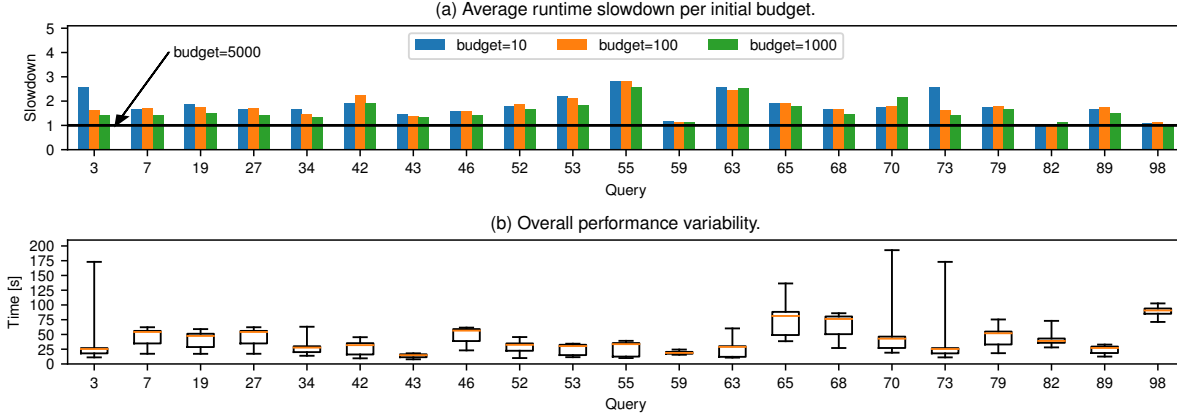


Figure 17: TPC-DS average runtime slowdown per query depending on initial budget (top); overall performance variability, summarized over initial budgets (bottom), plotted as IQR box; whiskers represent 1st and 99th percentiles.

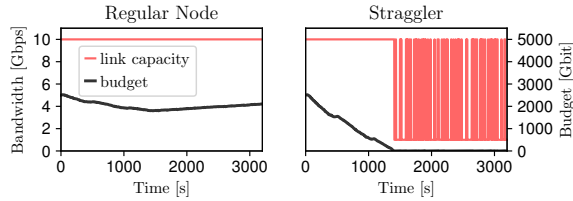


Figure 18: Link capacity allocation for TPC-DS on a token-bucket network, with initial budget = 2500 Gbit. Regular node network utilization (left); straggler node (right).

by Maricq et al. [47], across a number of initial token budgets. Figure 19 plots median estimates for two TPC-DS queries, along with 95% CIs and 10% error bounds around medians. Repetitions of the experiments are independent: each one runs on fresh machines with flushed caches, and at the beginning of each repetition, we reset the token budget. We reduce this initial budget over time to emulate the effects that previous experiments can have on subsequent ones: what this models is an environment in which many different experiments (or repetitions of the same experiment) are run in quick succession. This is likely to happen when running many experiments back-to-back in the same VM instances.

Query 82 (in the top of Figure 19) is agnostic to the token budget. Running more repetitions of this experiment tightens the confidence intervals, as is expected in CI analysis. In contrast, query 65 (in the bottom of the figure) depends heavily on the bucket budget; as a result, as we run more experiments, depleting the bucket budget, the query slows down significantly, and the initial CI estimates turn out to be inaccurate. In fact, the CIs *widen* with more repetitions, which is unexpected for this type of analysis. This is because the token bucket breaks the assumption that experiments are independent: in this model, more repetitions deplete the bucket that the next experiment begins with. These two queries represent extremes, but, as shown in the bar graph at the bottom of the figure, 80% of all queries we ran from TPC-DS suffer effects

like Query 65: most produce median estimates that are more than 10% incorrect by the time we fully deplete the budget.

This demonstrates that, when designing experiments, we cannot simply rely on the intuition that more repetitions lead to more accurate results: we must ensure that factors hidden in the cloud infrastructure are reset to known conditions so that each run is truly independent. Others have shown that cloud providers use token buckets for other resources such as CPU scheduling [62]. This affects cloud-based experimentation, as the state of these token buckets is not directly visible to users, nor are their budgets or refill policies.

## 5 Summary: Is Big Data Performance Reproducible in Modern Cloud Networks?

We return to our two basic questions: (1) *How reproducible are big data experiments in the cloud?*; and (2) *What can experimenters do to make sure their experiments are meaningful and robust?* Our findings are:

**F5.1: Network-heavy experiments run on different clouds cannot be directly compared.** Building a cloud involves trade-offs and implementation decisions, especially at the virtualization layer. Some of these decisions are well-documented by the platforms [6, 28], but others, including the ones we have examined in this paper, are not. Unfortunately, these differences can cause behaviors that result in different application performance, such as the bandwidth differences seen in Figure 10 or the latency effects seen in Figure 12.

Both of these effects are rather large, and are dependent on factors such as the size of the application's write buffer and specific patterns of communication. While these decisions presumably serve the clouds' commercial customers well, they complicate things for those who are trying to draw scientific conclusions; when comparing to previously-published performance numbers, it is important to use the same cloud to ensure that differences measured are that of the systems

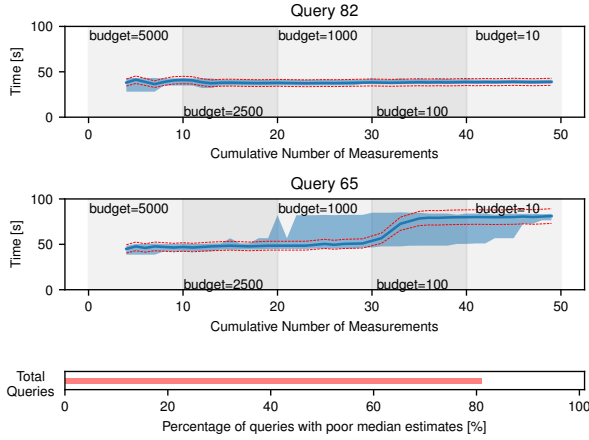


Figure 19: Median estimates (blue thick curve), 95% non-parametric confidence intervals (light blue filled space), and 10% error bounds (red dotted curves) for running two TPC-DS queries, over 5 token-bucket budgets. Bottom: number of queries for which we cannot achieve tight confidence intervals and accurate median estimates.

under test, and not artifacts of the cloud platform. Running on multiple clouds, can, however, be a good way to perform sensitivity analysis [36]: by running the same system with the same input data and same parameters on multiple clouds, experimenters can reveal how sensitive the results are to the choices made by each provider.

**F5.2: Even within a single cloud, it is important to establish baselines for expected network behavior. These baselines should be published along with results, and need to be verified before beginning new experiments.** Because cloud providers’ policies can be opaque, and implementation details can change over time, it is possible for changes to invalidate over time experiments within the same cloud. For example, after several months of running experiments in Amazon EC2, we began encountering new behavior: prior to August 2019, all `c5.xlarge` instances we allocated were given virtual NICs that could transmit at 10 Gbps. Starting in August, we started getting virtual NICs that were capped to 5 Gbps, though not consistently (this behavior is part of the underlying cause of the distributions in Figure 11). The reasons for this are not clear, and we have no way to know whether the “new” behavior is a transient effect in response to increased congestion that month or a new, permanent policy.

If one can establish baseline expectations for how the platform will perform, and incorporate checks for them into the experimental process [37], one can at least detect when changes have occurred. Experimenters should check, through micro-benchmarks, whether specific cloud resources (e.g., CPU, network) are subject to provider QoS policies.

As opposed to contention-related variability, this type of variability is deterministic under carefully selected micro-benchmarks. In the network, these microbenchmarks should at a minimum include base latency, base bandwidth, how la-

tency changes with foreground traffic, and the parameters to bandwidth token-buckets, if they are present. Furthermore, when reporting experiments, always include these *performance fingerprints* together with the actual data, as possible changes in results in the future could be explained by analyzing the micro-benchmark logs.

**F5.3: Some cloud network variability (in particular, interference from neighbors) can be modeled as stochastic noise, and classic techniques from statistics and experiment design are sufficient for producing robust results; however, this often takes more repetitions than are typically found in the literature.** Standard statistical tools such as ANOVA and confidence intervals [11, 36, 47] are effective ways of achieving robust results in the face of random variations, such as those caused by transient “noisy neighbors”; however, in order to be effective, they require many repetitions of an experiment, and, as we saw in Section 2, this bar is often not met in the literature. The more variance, the more repetitions are required, and as we saw in Figures 4, 5, and 6, network variance in the cloud can be rather high, even under “ideal” conditions. An effective way to determine whether enough repetitions have been run is to calculate confidence intervals for the median and tail, and to test whether they fall within some acceptable error bound (e.g., 5% of value they are measuring).

**F5.4: Other sources of variability cause behavior that breaks standard assumptions for statistical analysis, requiring more careful experiment design.** Some of the variability we have seen (e.g., Figures 12, 18, and 19) causes behavior that breaks standard assumptions for statistical analysis (such as iid properties and stationarity). As an integral part of the experimentation procedure, samples collected should be tested for normality [56], independence [46], and stationarity [23]. When results are not normally-distributed, non-parametric statistics can be used [26]. When performance is not stationary, results can be limited to time periods when stationarity holds, or repetitions can be run over longer time frames, different diurnal or calendar cycles, etc. Techniques like CONFIRM [47] can be used to test whether confidence intervals converge as expected.

Discretizing performance evaluation into units of time, e.g., *one hour* is helpful. Gathering median performance for each interval, and applying techniques such as CONFIRM over large-numbers of gathered medians results in significant and realistic performance data. Large intervals can smooth out noise, helping to reduce unrepresentative measurements.

We also find it helpful to “rest” the infrastructure and randomize [3] experiment order. Because it is hard to tell what performance-relevant state may build up in the hidden parts of the underlying cloud infrastructure, experimenters must ensure that the infrastructure is in as “neutral” a state as possible at the beginning of every experiment. The most reliable way to do this is to create a fresh set of VMs for every experiment. When running many small experiments, this can

be cost- or time-prohibitive: in these cases, adding delays between experiments run in the same VMs can help. Data used while gathering baseline runs can be used to determine the appropriate length (e.g., seconds or minutes) of these rests. Randomized experiment order is a useful technique for avoiding self-interference.

**F5.5: Network performance on clouds is largely a function of provider implementation and policies, which can change at any time.** Experimenters cannot treat “the cloud” as an opaque entity; results are significantly impacted by platform details that may or may not be public, and that are subject to change. (Indeed, much of the behavior that we document in Sections 3 and 4 is unlikely to be static over time.) Experimenters can safeguard against this by publishing as much detail as possible about experiment setup (e.g., instance type, region, date of experiment), establishing baseline performance numbers for the cloud itself, and only comparing results to future experiments when these baselines match.

**Applicability to other domains.** In this paper, we focused on big data applications and therefore our findings are most applicable in this domain. The cloud-network related findings we present in Section 3 are general, so practitioners from other domains (e.g., HPC) should take them in to account when designing systems and experiments. However, focusing in depth on other domains might reveal interactions between network variability and experiments that are not applicable to big data due to the intrinsic application characteristics. Therefore, while our findings in Section 4 apply to most other networked applications, they need not be complete. We also believe that a community-wide effort for gathering cloud variability data will help us automate reproducible experiment design that achieves robust and meaningful performance results.

## 6 Related Work

We have showed the extent of network performance variability in modern clouds, as well as how practitioners disregard cloud performance variability when designing and running experiments. Moreover, we have showed what the impact of network performance variability is on experiment design and on the performance of big data applications. We discuss our contributions in contrast to several categories of related work.

**Sound Experimentation (in the Cloud).** Several articles already discuss pitfalls of systems experiment design and presentation. Such work fits two categories: guidelines for better experiment design [3, 17, 38, 47] and avoiding logical fallacies in reasoning and presentation of empirical results [10, 21, 31]. Adding to this type of work, we survey how practitioners apply such knowledge, and assess the impact of poor experiment design on the reliability of the achieved results. We investigate the impact of *variability* on performance reproducibility, and uncover variability behavior on modern clouds.

**Network Variability and Guarantees.** Network variability has been studied throughout the years in multiple contexts,

such as HPC [8, 9], experimental testbeds [47] and virtualized environments [35, 40, 55]. In the latter scenario, many studies have already assessed the performance variability of cloud datacenter networks [43, 51, 63]. To counteract this behavior, cloud providers tackle the variability problem at the infrastructure level [12, 52]. In general, these approaches introduce network virtualization [30, 54], or traffic shaping mechanisms [18], such as the token buckets we identified, at the networking layer (per VM or network device), as well as a scheduling (and placement) policy framework [41]. In this work, we considered both types of variability: the one given by resource sharing and the one introduced by the interaction between applications and cloud QoS policies.

**Variability-aware Network Modeling, Simulation, and Emulation.** Modeling variable networks [27, 45] is a topic of interest. Kanev et al. [39] profiled and measured more than 20,000 Google machines to understand the impact of performance variability on commonly used workloads in clouds. Uta et al. emulate gigabit real-world cloud networks to study their impact on the performance of batch-processing applications [60]. Casale and Tribastone [14] model the exogenous variability of cloud workloads as continuous-time Markov chains. Such work cannot isolate the behavior of network-level variability compared to other types of resources.

## 7 Conclusion

We studied the impact of cloud network performance variability, characterizing its impact on big data experiment reproducibility. We found that many articles disregard network variability in the cloud and perform a limited number of repetitions, which poses a serious threat to the validity of conclusions drawn from such experiment designs. We uncovered and characterized the network variability of modern cloud networks and showed that network performance variability leads to variable slowdowns and poor performance predictability, resulting in non-reproducible performance evaluations. To counter such behavior, we proposed protocols to achieve reliable cloud-based experimentation. As future work, we hope to extend this analysis to application domains other than big data and develop software tools to automate the design of reproducible experiments in the cloud.

## Acknowledgements

We thank our shepherd Amar Phanishayee and all the anonymous reviewers for all their valuable suggestions. Work on this article was funded via NWO VIDI MagnaData (#14826), SURFsara e-infra180061, as well as NSF Grant numbers CNS-1419199, CNS-1743363, OAC-1836650, CNS-1764102, CNS-1705021, OAC-1450488, and the Center for Research in Open Source Software.



## Appendix – Code and Data Artifacts

### Raw Cloud Data:

[DOI:10.5281/zenodo.3576604](https://doi.org/10.5281/zenodo.3576604)

### Bandwidth Emulator:

[github.com/alexandru-uta/bandwidth\\_emulator](https://github.com/alexandru-uta/bandwidth_emulator)

### Cloud Benchmarking:

[github.com/alexandru-uta/measure-tcp-latency](https://github.com/alexandru-uta/measure-tcp-latency)

## References

- [1] *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*. ACM, 2017.
- [2] *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, Dallas, TX, USA, November 11-16, 2018*. IEEE / ACM, 2018.
- [3] A. Abedi and T. Brecht. Conducting repeatable experiments in highly variable cloud computing environments. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, pages 287–292. ACM, 2017.
- [4] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, et al. Spark SQL: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1383–1394. ACM, 2015.
- [5] A. C. Arpaci-Dusseau and G. Voelker, editors. *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018*. USENIX Association, 2018.
- [6] AWS Enhanced Networking. <https://aws.amazon.com/ec2/features/#enhanced-networking>, 2019.
- [7] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 242–253. ACM, 2011.
- [8] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs. There goes the neighborhood: performance degradation due to nearby jobs. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 41. ACM, 2013.
- [9] A. Bhatele, A. R. Titus, J. J. Thiagarajan, N. Jain, T. Gamblin, P.-T. Bremer, M. Schulz, and L. V. Kale. Identifying the culprits behind network congestion. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 113–122. IEEE, 2015.
- [10] S. M. Blackburn, A. Diwan, M. Hauswirth, P. F. Sweeney, J. N. Amaral, T. Brecht, L. Bulej, C. Click, L. Eeckhout, S. Fischmeister, et al. The truth, the whole truth, and nothing but the truth: A pragmatic guide to assessing empirical evaluations. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 38(4):15, 2016.
- [11] J.-Y. L. Boudec. *Performance Evaluation of Computer and Communication Systems*. EFPL Press, 2011.
- [12] B. Briscoe and M. Sridharan. Network performance isolation in data centres using congestion exposure (ConEx). IETF draft, 2012.
- [13] Z. Cao, V. Tarasov, H. P. Raman, D. Hildebrand, and E. Zadok. On the performance variation in modern storage stacks. In *15th USENIX Conference on File and Storage Technologies (FAST 17)*, pages 329–344, 2017.
- [14] G. Casale and M. Tribastone. Modelling exogenous variability in cloud deployments. *ACM SIGMETRICS Performance Evaluation Review*, 2013.
- [15] N. Chaimov, A. Malony, S. Canon, C. Iancu, K. Z. Ibrahim, and J. Srinivasan. Scaling Spark on HPC systems. In *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, pages 97–110. ACM, 2016.
- [16] J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.
- [17] C. Curtsinger and E. D. Berger. Stabilizer: Statistically sound performance evaluation. *SIGARCH Comput. Archit. News*, 41(1):219–228, Mar. 2013.
- [18] M. Dalton, D. Schultz, J. Adriaens, A. Arefin, A. Gupta, B. Fahs, D. Rubinstein, E. C. Zerneno, E. Rubow, J. A. Docauer, et al. Andromeda: Performance, isolation, and velocity at scale in cloud network virtualization. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI’18)*, pages 373–387, 2018.
- [19] Databricks Project Tungsten. <https://databricks.com/blog/2015/04/28/project-tungsten-bringing-spark-closer-to-bare-metal.html>.
- [20] Databricks Instance Types. <https://databricks.com/product/aws-pricing/instance-types>, 2019.

- [21] A. B. De Oliveira, S. Fischmeister, A. Diwan, M. Hauswirth, and P. F. Sweeney. Why you should care about quantile regression. In *ACM SIGPLAN Notices*, volume 48, pages 207–218. ACM, 2013.
- [22] J. Dean and L. A. Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, 2013.
- [23] D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366a):427–431, 1979.
- [24] B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift. More for your money: exploiting performance heterogeneity in public clouds. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 20. ACM, 2012.
- [25] B. Ghit and D. Epema. Reducing job slowdown variability for data-intensive workloads. In *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE, 2015.
- [26] J. D. Gibbons and S. Chakraborti. *Nonparametric statistical inference*. Springer, 2011.
- [27] Y. Gong, B. He, and D. Li. Finding constant from change: Revisiting network performance aware optimizations on iaas clouds. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 982–993. IEEE Press, 2014.
- [28] Google Andromeda Networking. <https://cloud.google.com/blog/products/networking/google-cloud-networking-in-depth-how-andromeda-2-2-enables-high-throughput-vms>, 2019.
- [29] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. Watson, A. W. Moore, S. Hand, and J. Crowcroft. Queues don’t matter when you can JUMP them! In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI’15)*, pages 1–14, 2015.
- [30] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International Co-NEXT Conference*, page 15. ACM, 2010.
- [31] T. Hoefler and R. Belli. Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015.
- [32] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang. The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, pages 41–51. IEEE, 2010.
- [33] B. Hubert, T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. Schroeder, J. Spaans, and P. Larroy. Linux advanced routing & traffic control. In *Ottawa Linux Symposium*, page 213, 2002.
- [34] A. Iosup, A. Uta, L. Versluis, G. Andreadis, E. Van Eyk, T. Hegeman, S. Talluri, V. Van Beek, and L. Toader. Massivizing computer systems: a vision to understand, design, and engineer computer ecosystems through and beyond modern distributed systems. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1224–1237. IEEE, 2018.
- [35] A. Iosup, N. Yigitbasi, and D. Epema. On the performance variability of production cloud services. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 104–113. IEEE, 2011.
- [36] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, 1990.
- [37] I. Jimenez, M. Sevilla, N. Watkins, C. Maltzahn, J. Lofstead, K. Mohror, A. Arpaci-Dusseau, and R. Arpaci-Dusseau. The popper convention: Making reproducible systems evaluation practical. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1561–1570. IEEE, 2017.
- [38] T. Kalibera and R. Jones. Rigorous benchmarking in reasonable time. *SIGPLAN Not.*, 48, 2013.
- [39] S. Kanev, J. P. Darago, K. Hazelwood, P. Ranganathan, T. Moseley, G.-Y. Wei, and D. Brooks. Profiling a warehouse-scale computer. In *ACM SIGARCH Computer Architecture News*, volume 43, pages 158–169. ACM, 2015.
- [40] D. Kossmann, T. Kraska, and S. Loesing. 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, 2010.
- [41] K. LaCurts, S. Deng, A. Goyal, and H. Balakrishnan. Choreo: Network-aware task placement for cloud applications. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 191–204. ACM, 2013.

- [42] P. Leitner and J. Cito. Patterns in the chaos—a study of performance variation and predictability in public iaas clouds. *ACM Transactions on Internet Technology (TOIT)*, 16(3):15, 2016.
- [43] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010.
- [44] J. R. Lorch and M. Yu, editors. *16th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2019, Boston, MA, February 26-28, 2019*. USENIX Association, 2019.
- [45] S. Madireddy, P. Balaprakash, P. Carns, R. Latham, R. Ross, S. Snyder, and S. Wild. Modeling I/O performance variability using conditional variational autoencoders. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 109–113. IEEE, 2018.
- [46] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, pages 50–60, 1947.
- [47] A. Maricq, D. Duplyakin, I. Jimenez, C. Maltzahn, R. Stutsman, and R. Ricci. Taming performance variability. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 409–425, 2018.
- [48] J. C. Mogul and L. Popa. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Computer Communication Review*, 42(5):44–48, 2012.
- [49] R. O. Nambiar and M. Poess. The making of TPC-DS. In *VLDB*, 2006.
- [50] K. Ousterhout, R. Rasti, S. Ratnasamy, S. Shenker, and B.-G. Chun. Making sense of performance in data analytics frameworks. In *NSDI '15*, volume 15, pages 293–307, 2015.
- [51] V. Persico, P. Marchetta, A. Botta, and A. Pescapé. Measuring network throughput in the cloud: the case of amazon ec2. *Computer Networks*, 93:408–422, 2015.
- [52] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren. Cloud control with distributed rate limiting. *ACM SIGCOMM Computer Communication Review*, 37(4):337–348, 2007.
- [53] J. Rexford, M. Balazinska, D. Culler, and J. Wing. Enabling computer and information science and engineering research and education in the cloud. 2018.
- [54] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. O. Guedes. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *WIOV*, 2011.
- [55] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of the VLDB Endowment*, 3(1-2):460–471, 2010.
- [56] S. Shapiro and M. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52:591–611, Dec. 1965.
- [57] L. Suresh, M. Canini, S. Schmid, and A. Feldmann. C3: Cutting tail latency in cloud data stores via adaptive replica selection. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15)*, pages 513–527, 2015.
- [58] A. Trivedi, P. Stuedi, J. Pfefferle, A. Schuepbach, and B. Metzler. Albis: High-performance file format for big data systems. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 615–630, 2018.
- [59] A. Uta, A. Custura, D. Duplyakin, I. Jimenez, J. Rellermeyer, C. Maltzahn, R. Ricci, and A. Iosup. Cloud Network Performance Variability Repository. <https://zenodo.org/record/3576604#.XfeXaOuxX0Q>, 2019.
- [60] A. Uta and H. Obaseki. A performance study of big data workloads in cloud datacenters with network variability. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*. ACM, 2018.
- [61] A. J. Viera, J. M. Garrett, et al. Understanding interobserver agreement: the kappa statistic. *Fam med*, 37(5):360–363, 2005.
- [62] C. Wang, B. Urgaonkar, N. Nasiriani, and G. Kesidis. Using burstable instances in the public cloud: Why, when and how? *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):11, 2017.
- [63] G. Wang and T. E. Ng. The impact of virtualization on network performance of amazon ec2 data center. In *INFOCOM*. IEEE, 2010.
- [64] T. White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.
- [65] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI'12*. USENIX, 2012.