Continuous and Client-centric Trust Monitoring in Multi-cloud Storage

Dimitri Van Landuyt, Luuk Raaijmakers, Ansar Rafique and Wouter Joosen
imec-DistriNet, Dept. of Computer Science, KU Leuven, B-3001 Heverlee, Belgium

Keywords: Trust Monitoring, Continuous Monitoring, Black-box Monitoring, Multi-cloud Storage, Federated Cloud Storage.

Abstract: Multi-cloud storage is the practice of composing the data tier of an application with heterogeneous cloud storage technologies, resources and services. In a federated cloud storage architecture which involves multiple cloud storage providers, both the complexity and the importance of trust management increases drastically. A trust relation is established between a data owner and a cloud storage provider when the data owner subscribes to the service and service level agreements (SLAs) are established. In practice, this trust relation is seldom revised, only when serious infractions are discovered and made public.
In this paper, we evaluate the potential of continuous and client-centric trust monitoring of cloud storage services. This approach leverages upon the statistical correlations between black-box performance metrics and reported white-box metrics, and identifies significant deviations between both. We evaluate in terms of (a) the effectiveness of correlating black-box and white-box measurements, and (b) the incurred performance overhead of the approach to continuously monitor for trust.

1 INTRODUCTION

Cloud storage services provide scalable and on-demand storage facilities. Due to the heterogeneity inherent to the underlying database technologies (NoSQL), many organizations are adopting a federated, multi-storage strategy (Bermbach et al., 2011), in which different storage services from different cloud storage providers (CSP) are combined.

The requirement of a trust relationship between service consumers and cloud service providers is a key impediment to cloud adoption (Rong et al., 2013; Habib et al., 2012). Lack of physical access to the storage infrastructure, information asymmetry, and diverging economical interests are at the foundation of this lack of trust. Indeed, cloud providers have economic incentives to be dishonest, e.g. by deliberately allocating less CPU or memory resources than agreed upon to maximize economic benefit (Zhang et al., 2014), or by not disclosing unexpected service disruptions to their clientele to avoid reputational harm.

Arguably, the issue of trust in cloud storage services is even more stringent, as the cloud storage paradigm involves entrusting third-party CSPs with data assets that are of crucial value to the data owner. One common example is a Software-as-a-Service (SaaS) offering that is built upon existing cloud storage resources to improve their operational efficiency. To alleviate the information asymmetry, CSPs commonly provide access to white-box monitoring services, allowing data owners to consult a number of measurements and performance indicators such as the delivered uptime and read/write latency. However, similarly to the scenarios described above, this approach still requires data owners to have a degree of trust in the reported values as such measurements are collected, reported and controlled exclusively by the CSP. In practice, if there is a discrepancy between the reported and the delivered service, regardless of it being deliberate or not, data owners have no simple way to knowing. CSPs commonly provide monitoring APIs and dashboard to trust between application providers/data owners and CSPs. However, since this is the only source of information, there is from a client-centric perspective no straightforward way to verify the correctness of this information.

A number of different approaches have been proposed and investigated to deal with the problem of limited trust in a cloud computing context. Reputation and recommender frameworks (Li et al., 2012; Khan and Hamlen, 2012; Habib et al., 2014; Habib et al., 2013; Muchahari and Sinha, 2012) collect positive and negative experiences from many service consumers and calculate trust ranks. Broker-

Van Landuyt, D., Raaijmakers, L., Rafique, A. and Joosen, W.
Continuous and Client-centric Trust Monitoring in Multi-cloud Storage.
DOI: 10.5220/0007713201000110
In *Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER 2019)*, pages 100-110
ISBN: 978-989-758-365-0

based approaches (Uikey and Bhilare, 2013) involve a trusted-third party reseller of cloud services that assume part of the liability or trust or assume an active rule, such as third-party auditors (Zhang et al., 2014). Other approaches rely on standardization and certification (Cloud Security Alliance (CSA), 2018) and periodic audits (Popa et al., 2011). These approaches essentially rely extensively on trust in external parties.

In this paper, we evaluate the potential of a complementary approach that focuses on the issue exclusively from the point of a single data owner. This approach is similar to the continuous monitoring and trust assessment solution of Li et al. (Li and Du, 2013). This approach involves continuously monitoring both the reported measurements (through white-box metrics) and delivered service (through black-box metrics) and attributing trust scores based on statistically-relevant discrepancies between both. This approach is based upon leveraging known (and thus expected) statistical correlations between white-box and black-box measurements (Schoonjans et al., 2015). Continuously checking the accuracy and veracity of the reported white-box measurements as such enables continuously assessing the trustworthiness of the CSP. We focus on functional validation of this approach and evaluating the performance cost of continuous monitoring. The results indicate that the performance overhead is acceptable in a realistic multi-cloud scenario. Additional results confirm the expected statistical correlations between both metric types.

The remainder of this paper is structured as follows: Section 2 provides a discussion of the background of this article. Section 3 discusses a prototype implementation of continuous black-box monitoring which is evaluated in Section 4. Section 5 discusses related work and finally, Section 6 concludes the paper.

2 BACKGROUND

Section 2.1 first provides background information on multi-cloud storage and then Section 2.2 discusses the notion of trust in this specific context.

2.1 Multi-cloud Storage

Cloud storage allows data owners and service providers to acquire storage resources on demand and store their data in the cloud. Apart from providing flexible data storage, it also alleviates the burden of maintaining an expensive in-house storage infrastructure (Rafique et al., 2017). As shown in Table 1, there is a wide variety of cloud storage providers (CSPs) and Database-as-a-Service (DBaaS) providers in the market, each focusing on different SLA guarantees and with support for different database technologies (e.g., NoSQL databases) (Rafique et al., 2018).

However, the paradigm of online data hosting and data access introduces serious concerns about data security, performance (i.e. latency), availability, and vendor lock-in. In addition, due to the heterogeneity inherent in the underlying database technologies, limited application requirements can be satisfied by a single cloud storage provider.

Therefore, a federated or multi-cloud storage architecture in which technologically heterogeneous storage services and database technologies from multiple cloud storage providers are combined within the same application, is becoming an increasingly popular tactic for service providers (Bermbach et al., 2011; Rafique et al., 2018).

2.2 Trust in Multi-cloud Storage

In the context of cloud storage, data owners have expectations from CSPs. These are typically codified and agreed upon in Service Level Agreements (SLAs). SLAs represent contracts between CSPs and data owners. In an SLA, guarantees offered by the CSP are clearly outlined. Next to this, an SLA also details the expected consequences when these promises aren't fulfilled. SLAs are an important component in the trust relationship between data owners and CSPs. When the CSP refrains from offering promised agreements of the SLA, this can seriously hurt the trust relationship.

Table 1 lists (in columns 3-5) the SLA guarantees offered by 10 representative commercial CSPs. It shows that these SLAs focus mainly on availability, while only two CSPs provide explicit performance guarantees (in terms of latency and throughput). However, although all CSPs guarantee some form of availability, definitions vary widely and differences can be observed regarding the employed calculation method: some CSPs calculate at the basis of unavailability, others in terms of downtime, and others in terms of error rate. Further investigation, however, indicates that each provider adopts a slightly different interpretation, making SLA promises in general rather difficult to compare fairly across providers.

The trust definition introduced in (EMC, 2011) equates trust to control and visibility. A certain degree of control and internal visibility or transparency is needed to establish trust. When outsourcing data to a CSPs, data owners trade in both in terms of visibility and trust. Information asymmetry (Akerlof, 1970)

Table 1: Illustrative overview of 10 different cloud storage providers (CSPs), the nature of the SLAs, and monitoring metrics they offer.

CSP (DB technology)	**SLA: Availability**		**SLA: Performance**		**Monitoring API**		
	Pct.	**Based on**	**Latency**	**Thr-oughput**	**Availability**	**Latency**	**Throughput**
InstaClustr (Instaclustr, 2017; Instaclustr, 2018) (Cassandra)	99,9% / 99,95%	Unavai-lability	- / 99%	-	`nodeStatus`	`clientRequestRead` `clientRequest-Write`	`reads` `writes`
ScaleGrid (ScaleGrid, 2016; MongoDB, 2018b; ScaleGrid, 2012) (MongoDB, Redis)	99,95%	Unavai-lability	-	-	`events.re-start`		
Redis Labs (Redis Labs, 2018; RedisLabs, 2018) (Redis)	99,95% / 99,99%	/	-	-		`Read latency` `Write latency`	`Reads/sec` `Writes/sec`
Datastax Managed Cloud (DataStax, 2018) (Cassandra)	99% / 99,9%	Unavai-lability	-	-	`nodetool status`	`nodetool cfstats`	`nodetool cfstats`
Compose (Compose, 2018b; Compose, 2018a) (MongoDB, Redis)	99,98%	/			`databa-ses.status`		
MongoDB Atlas (MongoDB Atlas, 2018; MongoDB, 2018a) (MongoDB)	99,95%	Downtime			`serverStatus` Uptime	`serverStatus` `OpLatencies`	`serverStatus` `OpLatencies.ops`
Google Cloud Storage (Google Cloud, 2016; Google Cloud, 2018)	99,9% / 99,95%	Error Rate	-	-	uptime checks	`request_latency`	
Microsoft Azure (Microsoft Azure, 2015; Microsoft Azure, 2018)	99,9% / 99,99%	Error Rate			`status & request errors`	`SuccessServerLa-tency`	
Amazon EBS (AWS - Amazon Web Services, 2018b; AWS - Amazon Web Services, 2018a; AWS - Amazon Web Services, 2018)	99% / 99,99%	Unavai-lability	-	99%	volume status check	`volumeTotalRead-Time/` `volumeReadOps`	`volumeReadOps` `volumeWriteOps`
Oracle Cloud (Oracle Cloud, 2018; Oracle, 2018)	99,9%	Error Rate	-	90%[*]	`gene-ral.status`		`performan-ce.throughput`

[*] 99,9% of the time.

is a key problem: data owners are only provided with a subset of the available information about the internal workings of the storage offering, while CSPs have full control and visibility. Increasing the amount of control consequently increases the amount of trust.

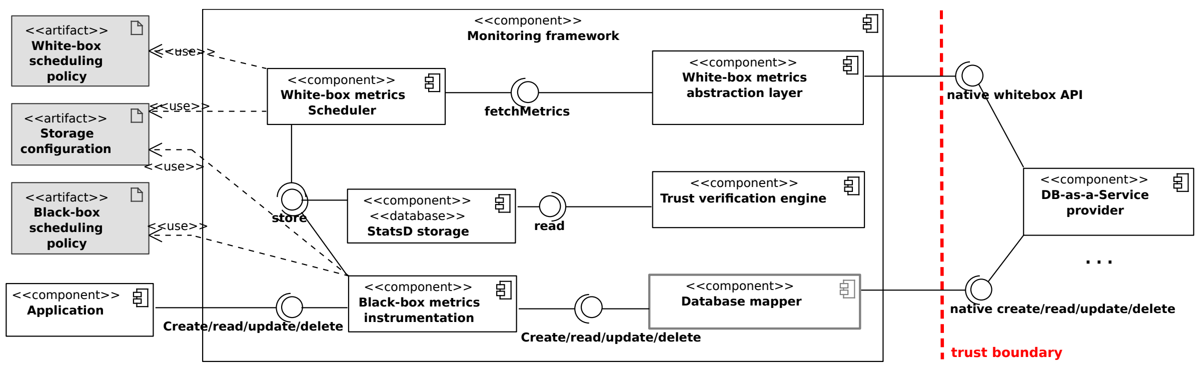
For this reason, many CSPs currently offer monitoring service interfaces and extensive data control dashboards, which are accessible to customers. Having an internal view of the system, which can for example be achieved by monitoring, increases visibility and in turn can increase the level of trust. Table 1 (final column) summarizes the metric types that can be accessed (programmatically or via these dashboards) in 10 investigated commercial cloud storage offerings. As these metrics are reported by the cloud providers themselves and are inherently based on the an internal view of the storage offering, they are in effect *white-box* metrics.

Motivation. CSPs retain full control on the information they provide. Dishonest CSPs could be deliberately selective in the information they provide, tamper with or alter the information offered to their customers, e.g. to avoid reputational damage in case of service disruption. Since CSPs are profit-based entities, they have an economic incentives to be dishonest because offering data owners less than agreed upon allows them to support more users and consequently gain more profit (Zhang et al., 2014).

3 APPROACH AND IMPLEMENTATION

Many approaches exist to maintain trust in a multi-cloud context. In this paper, we explore and evaluate a continuous, client-centric monitoring approach. More specifically, measurements obtained in a client-centric, black-box fashion are compared to the white-box measurements reported by the CSPs. By exploiting knowledge on existing (and thus expected) statistical correlations (Schoonjans et al., 2015) between values obtained from both perspectives, trust rankings of CSPs can be created and continuously revised.

The overall architecture is depicted in Figure 1. As shown, it is comprised of three major subsystems: (i) an abstraction layer for fetching white-box measurements from a wide range of CSPs and storage nodes (`White-box metrics abstraction layer`), which includes a scheduler to obtain specific white-box measurements according to an application-specific monitoring policy (`White-box metrics scheduler`), (ii) an instrumentation layer for obtaining black-box measurements (`Black-box metrics`



<<artifact>> White-box scheduling policy
<<artifact>> Storage configuration
<<artifact>> Black-box scheduling policy
<<component>> Application
<<use>>
<<use>>
<<use>>
<<use>>
<<component>> Monitoring framework
<<component>> White-box metrics Scheduler
fetchMetrics
<<component>> White-box metrics abstraction layer
store
<<component>> <<database>> StatsD storage
read
<<component>> Trust verification engine
<<component>> Black-box metrics instrumentation
<<component>> Database mapper
Create/read/update/delete
Create/read/update/delete
native whitebox API
<<component>> DB-as-a-Service provider
. . .
native create/read/update/delete
trust boundary

Figure 1: The overall architecture of the monitoring framework (*UML component diagram*).

instrumentation) through observation on regular application calls, and (iii) the Trust verification engine.

We have implemented a prototype implementation which is available on-line (Raaijmakers, 2018).

The White-box metrics abstraction layer component provides built-in abstraction support for in total 18 white-box provided by PostgreSQL, Redis, MongoDB, and Cassandra.

The Black-box metrics instrumentation component builds upon a database mapper framework (also called *Object-NoSQL Database Mapper* or ONDM (Cabibbo, 2013; Reniers et al., 2017)) to collect black-box measurements through instrumentation of regular application calls. Metrics are recorded for (i) uptime, (ii) number of read requests, (iii) number or failed requests (yielding an error or time-out), (iv) average read latency (10-second interval, 1-hour interval and 24-hour interval), and (v) average write latency (10-second interval, 1-hour interval and 24-hour interval).

The Trust verification engine implements a number of trust ranking algorithms that essential look at the discrepancies between the reported white-box measurements and the black-box metrics obtained from a client perspective. Many trust ranking models can be adopted to calculate trust scores, ranging from very straightforward methods such as calculating the difference between black-box and white-box measurements, to more advanced models to calculate and predict trustworthiness based on the obtained inputs (for example, using machine learning-based classifiers). The prototype implementation currently provides the following trust models, but is extensible in this regard.

- Percentage deviation. The percentage deviation method calculates the difference between black-box and white-box measurements as the percentage deviation based on the black-box measurement. A black-box measurement of 210 ms and a white-box measurement of 200 ms results in a trust score of 95% using the following calculation: $(210-200)/200 = 5\%$, a 5% deviation means that this measurement is $100\% - 5\% = 95\%$ trustworthy.
- Mean Square Error (MSE). The MSE of the two measurements can also be used as a trust score. Using the example above yields: $(210-200)^2 = 100$. The closer the value is to zero, the more trustworthy the storage node proves to be. This method results in more extreme values as the deviation between the two values increases, due to the power used in the calculation.
- Threshold. The threshold method treats all deviations equally, regardless of their size, as long as they are above a certain threshold. This method can prove to be useful in scenarios where each deviation above a certain level has the same severity, such as mission-critical applications that can not tolerate any violations.

In the most simple deployment, the monitoring framework is configured as *application middleware*. In this deployment, the framework runs entirely in the same execution environment as the application. For more advanced scenarios, the three main components can each be deployed on separate nodes: the black-box and white-box components are not coupled to each other, but they store their data on the same data store which is used by the trust ranking component.

4 EVALUATION

This section evaluates the presented approach in terms of (i) the effectiveness of correlating black-box and white-box measurements, and (ii) the additional performance overhead caused by continuous client-centric black-box monitoring.

4.1 Experiment Setup

The evaluations described in this section are performed on a machine with a 2.6 GHz Intel Core i7 processor and 16GB 2133 MHz LPDDR3 RAM, running MacOS 10.13.4. The performance measurements are obtained using the YCSB benchmark system[1] which supports a number of reusable workloads out of the box. The tests are executed in a configuration that involves a native Cassandra instance and each of the presented experiments were executed again over a Redis database to confirm the findings. The executed workloads consisted of 1 million insert operations and 1 million read operations, and a 5-second monitoring interval was used to obtain both white-box and black-box measurements for read and write latency.

4.2 Verification Accuracy

The approach presented in this paper builds upon the existence of statistical correlations between white-box measurements and black-box measurements obtained from a client perspective. These have been shown in earlier work (Schoonjans et al., 2015). In this first set of experiments, we evaluate the accuracy of the proposed method, by comparing the obtained black-box measurements with the white-box measurements.

Table 2 displays the average latencies after each workload. Column 3 shows the reported white-box latencies, obtained via the Cassandra client API. Column 4 presents the black-box latency obtained with the prototype, whereas the fifth column shows the average latencies as obtained by YCSB, which are also measured in a black-box fashion and were included to have a third perspective on the latencies.

In terms of absolute values, there is a large discrepancy between the white-box and black-box measurements. This is indicated by column 6, showing the ratio of black-box over white-box latencies. For insert operations in Cassandra, the black-box latencies (measured from an external perspective and thus including network latency) are almost 26 times higher than the self-reported white-box latencies. Comparing the obtained black-box measurements with the YCSB latencies (in the final column), which are also measured from a black-box perspective, the ratio is much smaller: between 1.04 and 1.33.

This gives a strong indication that the measured black-box measures are at least not *incorrect*: they follow almost the same ratio as YCSB for each storage technology. As mentioned. the obtained black-box values include more operations, e.g. the additional latency towards the client, which are not counted in the white-box latency measurement which is obtained nearer to the database node. In this experiment, the discrepancy between both measurement types in terms of absolute values is specific to the setup (network latency) and incidental.

Figure 2 shows the detailed write latencies for Cassandra, during the workload involving 1 million insert operations, measured every 5 seconds. Despite the large discrepancy between the two measurements (factor 26), these graphs illustrate that they both reflect the same changes. At times t=13, t=20, t=36 and t=55 small changes in the white-box latency were observed, which are also reflected in the black-box latencies. These results confirm that this black-box metric can be used for verification, even though the absolute values do not correspond.

In a realistic deployment setting, these measurements will additionally be impacted by natural variations in the network latency between the storage service and the client. This is most relevant in a multi-cloud configuration that involves multiple CSPs, and plays a less significant role in a local, on-premise deployment (e.g., in a local data center context) such as in our evaluations. Trust ranking models that tolerate such variations will necessarily involve statistical prediction models of end-to-end latency (Kim and Yi, 2010; Yamamoto et al., 2015; Madhyastha et al., 2006). Integration of such models in the prototype and further validation of the approach outside of lab settings are considered part of our future work.

4.3 Performance Overhead of Black-box Measurements

In this second experiment, identical workloads were executed on both a native Cassandra instance (the baseline for comparison) and the prototype implementation that (i) measures black-box latency and (ii) stores these measurements in a central database. The experiment focuses on quantifying the performance overhead introduced by instrumenting regular read and insert requests with support for black-box monitoring. This is done by systematically comparing the results of a benchmarking workload of the baseline with the performance results of the prototype, focusing on the increase in latency.

For this experiment, we have executed workload D of YCSB, which is a combined read-write workload, with 95% read operations and 5% inserts. Each workload was executed with $N = 1$ million operations, and each experiment was repeated $M = 3$ times to ensure fair comparison and remove the influence of

[1] YCSB: https://github.com/brianfrankcooper/YCSB

Table 2: Latency measurements obtained in a black-box and white-box way. The ‘WB’ data series represent white-box measurements, whereas the ‘BB’ and ‘YCSB’ series represent black-box measurements obtained by respectively the prototype and YCSB.

Database	**read / insert**	**WB (ms)**	**BB (ms)**	**YCSB (ms)**	**ratio BB/WB**	**ratio BB/YCSB**
Cassandra	insert	0.034	0.874	0.657	25.70	1.33
	read	0.079	0.780	0.593	9.87	1.31
Redis	insert	0.113	0.409	0.359	3.62	1.14
	read	0.012	0.174	0.168	14.5	1.04

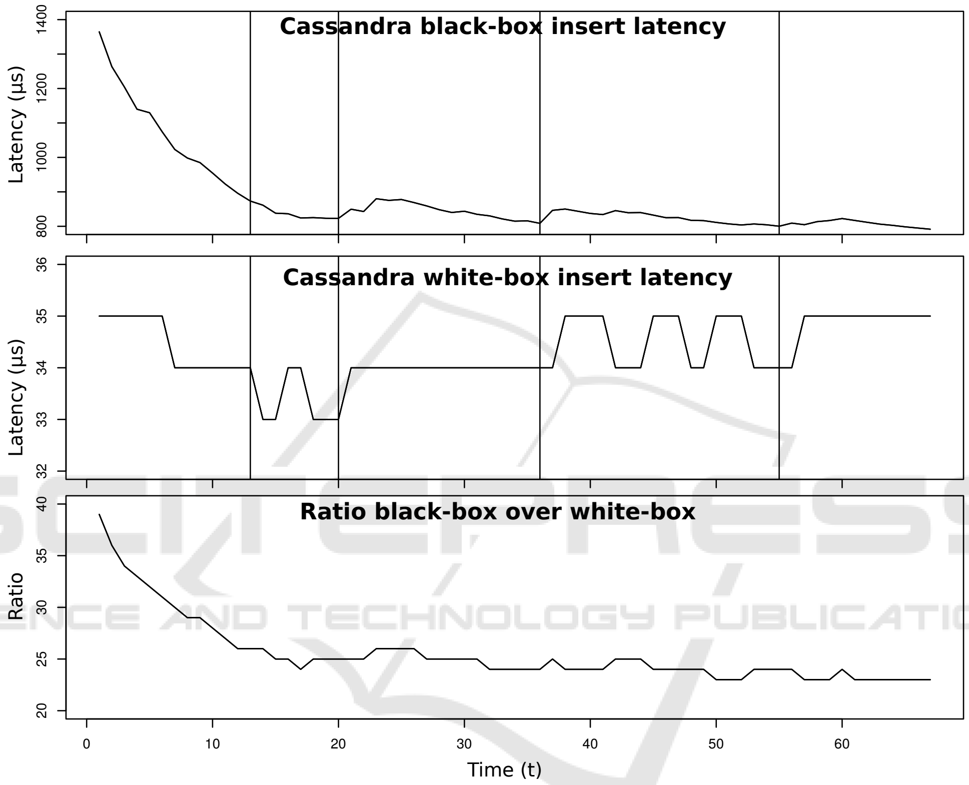


Figure 2: Cassandra insert latencies measured in a white-box and black-box way. Horizontal lines at t=13, 20, 36 and 55 indicate changes that are reflected in both black-box and white-box measurements.

accidental outliers.

The second row of Table 3 shows the results of the experiments for Cassandra: compared to the baseline variant, we observe an increased latency of on average 141 μs for read operations (corresponding to a 63% increase) and of on average 139 μs for the insert operations (corresponding to a 44% increase).

To confirm these findings, the experiment was repeated for the Redis database. The third row of Table 3 presents a summary overview of these results. It confirms that (a) the relative differences between latencies using the native APIs compared to the monitoring framework are substantial (40% to even 100%) and (b) no substantial differences can be observed across databases.

To pinpoint the specific cause of the extra overhead, additional experiments were conducted to assess the impact of storing the obtained measurements to the database. These experiments showed that the overhead increase is caused by storing the black-box measurements, which in effect introduces extra I/O operations for each request, regardless of it being a read- or insert-request. Figure 5 shows the results of a variant of the prototype that obtains the black-box measurements, but does not actually persists them to disk (the ‘without storing’ data series in Figure 5).

As shown, not actually storing the measurements has lead to a significant improvement in performance, more precisely latencies that are almost the same as for the native API. Having as such effectively pinpointed the main bottleneck allows to formulate a number of possible optimizations, such as a batch-

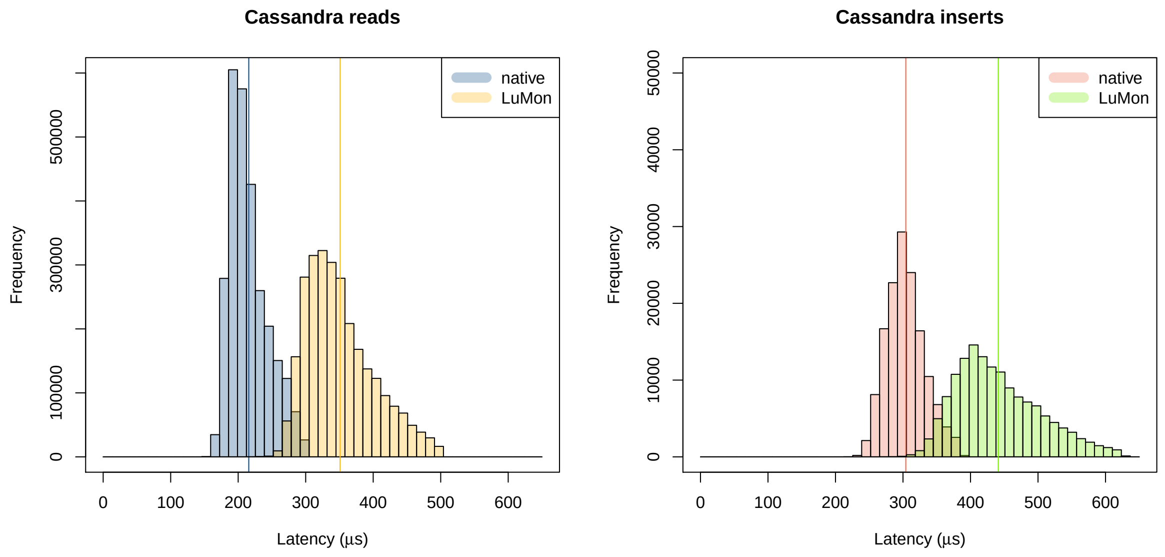


Figure 3: Cassandra read/insert latencies.

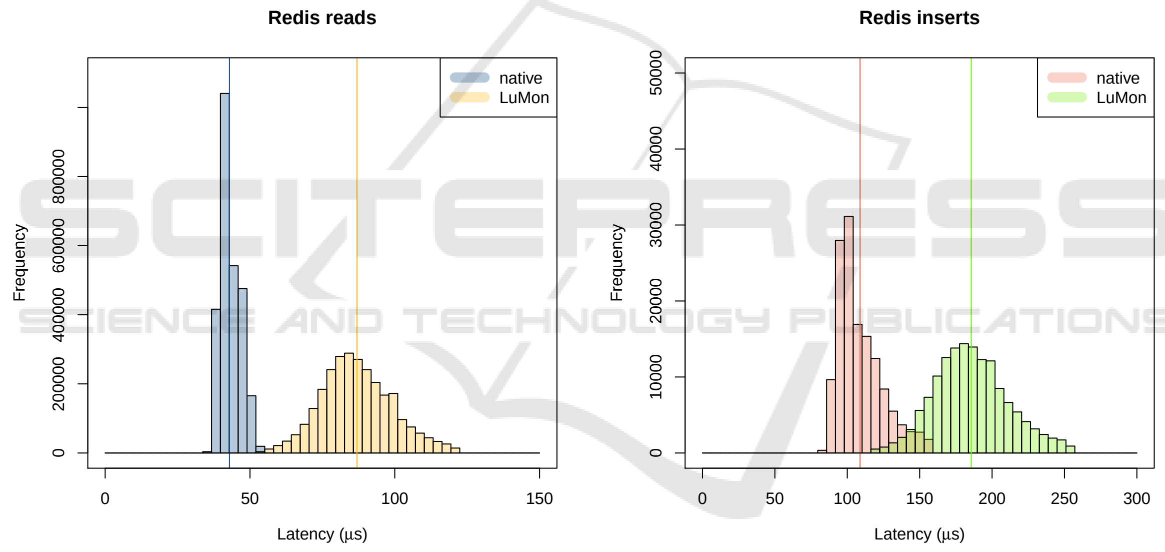


Figure 4: Redis read/insert latencies.

driven strategy for persisting the black-box measurements.

Furthermore, it is important to put these results in perspective: these experiments were performed on a local, single-node setup and thus does not take into account network latency and additional performance costs incurred in larger database clusters or complex inter-CSP federations. In comparison to the latencies of storage operations in more realistic cluster setups (Reniers et al., 2017), the absolute latency increases of 70-140 μs are not altogether that substantial.

5 DISCUSSION AND RELATED WORK

In a cloud context, the most common approach to trust involves trusted third parties (T3P) and trust brokerage. For example, the approach of Zhang et al. (Zhang et al., 2014) features T3P that audits cloud service providers in terms of memory usage. Muchahari et al. (Muchahari and Sinha, 2012) have presented a dynamic trust monitor that performs continuous calculation based on SLA and QoS and reports it to a central registry (Muchahari and Sinha, 2012). Analogously, recommender (Li et al., 2012), marketplace (Habib

Table 3: Summary of average latencies.

Database	**ops**	**P (μs)**	**native (μs)**	**abs diff**	**% diff**
Cassandra	read	364	223	141	63
	insert	453	314	139	44
Redis	read	90	45	45	100
	insert	190	118	72	61

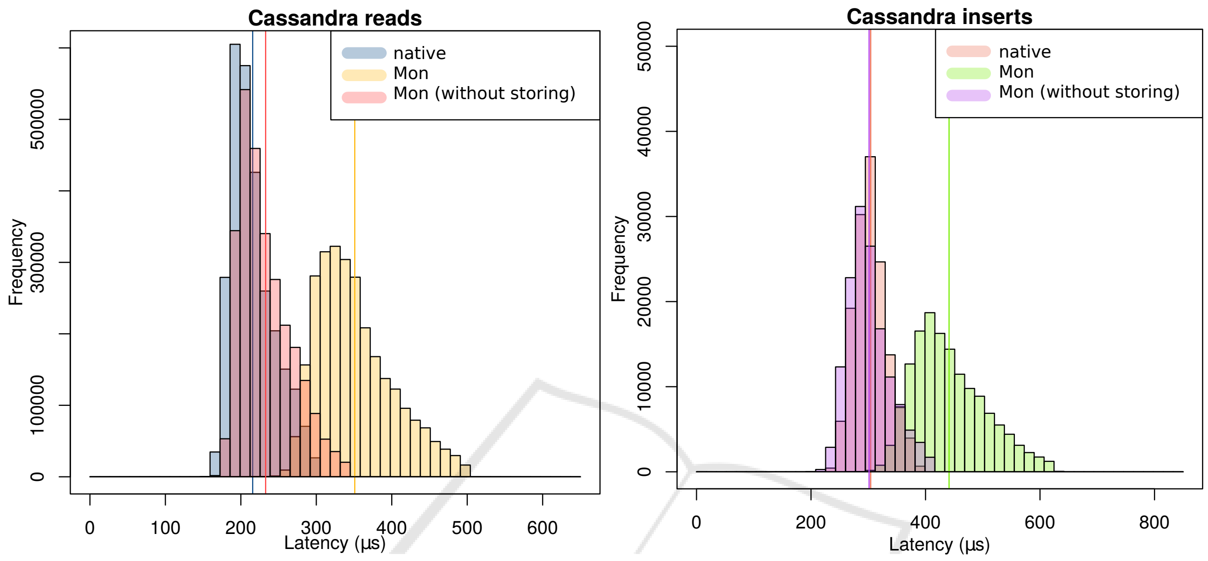


Figure 5: Cassandra read/insert overhead.

et al., 2013; Habib et al., 2014) and reputation systems (Khan and Hamlen, 2012) rely on a centralized trusted entity that collects and shares experiences between data owners and CSPs and facilitates matchmaking and service brokerage.

The solution discussed in this article is complementary, in the sense that it assumes the point of view of a single data owner and does not build upon the precondition of a trusted third party.

The industry survey of Lins et al. (Lins et al., 2016) stresses the necessity of continuous auditing. In terms of the architectural styles explored in this survey, the approach presented in this paper is an example of the 'Monitoring and Control Layer'-type system, but then from an externalized perspective. Key arguments against systematic benchmarking of cloud services (Li et al., 2010) are that these activities commonly introduce (i) additional cost, and (ii) may have side-effects, for example benchmark systems introduce artificial data which may hinder a production system. In our approach, the impact on a production system is reduced by instrumenting regular application operations with the acquisition of black-box measurements. Similar approaches have been implemented to continuously audit the privacy-preservation and data security aspects of a CSP, leveraging techniques of trusted computing (Kai et al., 2013). In such protocols, trust guarantees w.r.t. the correctness of the reported service levels are constructed by means of cryptographic proofs. Yang et al. (Yang et al., 2013) have proposed an architecture that supports these protocols in a multi-cloud storage architecture.

Increasing trust between data-owners and CSPs can be done in several ways. Since trust is based on lack of sufficient information, the solution is to increase the visibility of the system, by gathering and sharing information. In (Zhang et al., 2014) this approach is taken. While these approaches increase the trustworthiness of CSPs, they essentially shift the trust problem to the T3P.

Quantifying and ranking trust has been an active topic of research in reputation systems (Kamvar et al., 2003). The Cloud Security Alliance (CSA) (Cloud Security Alliance (CSA), 2018) proposed the CAIQ (Consensus Assessments Initiative Questionnaire), at the basis of which rankings and trust information scores can be calculated. This scoring scheme has been used extensively in the context of trust-based systems, for example in the work of Habib et al. (Habib et al., 2014; Habib et al., 2013). Cloud-trust, the security assessment model of Gonzalez et al. (Gonzales et al., 2017) involves defining a reference model for multi-tenant IaaS services that explicitly offers security controls and trust zones, and derives trust scores from lower-level security-related metrics. Although we have only worked with initial and simplistic trust ranking schemes, in future work, we will investigate more sophisticated trust ranking models that deal with problems such as hysteresis between metrics or other causal or temporal relations that may hamper the accuracy of the proposed trust models.

Client-centric approaches to benchmark or assess distributed systems have been successfully applied in the context of system properties for which a global view is hard to construct, such as consistency (Golab et al., 2014; Bermbach et al., 2014). Similar approaches have been successfully adopted to benchmark Infrastructure-as-a-Service (IaaS) cloud services from an externalized perspective (Folkerts et al., 2012; Xiong et al., 2013). Wood et al. (Wood et al., 2007) discuss the efficacy of unobtrusive black-box performance profiling in comparison to gray-boy performance profiling for the purposes of dynamic migration of virtual machines. The framework of Sakr et al. (Sakr and Liu, 2012) implements client-centric monitoring of cloud-hosted databases for the purpose of adaptive SLA-based provisioning and cost management. This framework demonstrates how adopting a client-centric monitoring perspective can improve flexibility and trust. Schoonjans et al. (Schoonjans et al., 2015) have shown the statistical correlations between some black-box and white-box metrics, showing that these metrics can indeed be indicative of actual service delivered. Related approaches employ black-box metrics predictive for finding faults, anomalies, and deviations in performance (Nguyen et al., 2013).

The approach evaluated in this paper exploits these correlations to find and monitor for significant deviations between delivered and observed measurements, and attribute a level of trust. The main benefit of adopting a client-centric, black-box approach is that we effectively measure the end-to-end service delivered to the customer, including external factors such as network latency which are not reported in whitebox measurements.

6 CONCLUSION

In this paper, we have evaluated an approach of continuous trust monitoring that leverages upon the existing statistical correlations between black-box metrics obtained from a client perspective, and white-box metrics which are reported by cloud storage providers. By continuously comparing the delivered service in terms of key performance and availability metrics (latency, throughput, uptime), trust rankings are created and continuously maintained or revised, allowing data owners to react to service disruptions or SLA violations in a timely manner.

We have shown the validity and feasibility of the proposed approach in a prototype which we in turn evaluated in terms of the incurred performance overhead.

ACKNOWLEDGEMENTS

This research is partially funded by the Research Fund KU Leuven (project GOA/14/003 - ADDIS) and the DeCoMAdS SBO project.

REFERENCES

Akerlof, G. A. (1970). The market for “lemons”: Quality uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84(3):488–500.

AWS - Amazon Web Services (2018a). Amazon compute service level agreement. https://aws.amazon.com/ec2/sla/[Online; accessed March 26, 2018].

AWS - Amazon Web Services (2018b). Amazon ebs product details. https://aws.amazon.com/ebs/details/[Online; accessed March 26, 2018].

AWS - Amazon Web Services (2018). Monitoring the status of your volumes. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring-volume-status.html. [Online; Accessed April 22, 2018].

Bermbach, D., Klems, M., Tai, S., and Menzel, M. (2011). Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 452–459. IEEE.

Bermbach, D., Zhao, L., and Sakr, S. (2014). Towards comprehensive measurement of consistency guarantees for cloud-hosted data storage services. In Nambiar, R. and Poess, M., editors, *Performance Characterization and Benchmarking*, pages 32–47, Cham. Springer International Publishing.

Cabibbo, L. (2013). Ondm: an object-nosql datastore mapper. *Faculty of Engineering, Roma Tre University. Retrieved June 15th*.

Cloud Security Alliance (CSA) (2018). The cloudtrust protocol (ctp). https://cloudsecurityalliance.org/group/cloudtrust-protocol/#_overview.

Compose (2018a). Compose scalegrid monitoring. https://help.compose.com/docs/essentials-compose-monitoring. [Online; Accessed April 22, 2018].

Compose (2018b). Enhanced sla - compose. https://help.compose.com/docs/enhanced-sla[Online; accessed March 26, 2018].

DataStax (2018). Datastax managed cloud service level agreement. https://www.datastax.com/dmc-service-level-agreement[Online; accessed March 26, 2018].

EMC (2011). Proof, not promises: creating the trusted cloud. http://www.emc.com/collateral/emc-perspective/11319-tvision-wp-0211-ep.pdf [Online; accessed December 28, 2017].

Folkerts, E., Alexandrov, A., Sachs, K., Iosup, A., Markl, V., and Tosun, C. (2012). Benchmarking in the cloud: What it should, can, and cannot be. In *Technology Conference on Performance Evaluation and Benchmarking*, pages 173–188. Springer.

Golab, W., Rahman, M. R., AuYoung, A., Keeton, K., and Gupta, I. (2014). Client-centric benchmarking of eventual consistency for cloud storage systems. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, pages 493–502. IEEE.

Gonzales, D., Kaplan, J. M., Saltzman, E., Winkelman, Z., and Woods, D. (2017). Cloud-trust—a security assessment model for infrastructure as a service (iaas) clouds. *IEEE Transactions on Cloud Computing*, 5(3):523–536.

Google Cloud (2016). Google cloud storage sla. https://cloud.google.com/storage/sla[Online; accessed March 26, 2018].

Google Cloud (2018). Stackdriver monitoring. https://cloud.google.com/monitoring/. [Online; Accessed April 22, 2018].

Habib, S. M., Hauke, S., Ries, S., and Mühlhäuser, M. (2012). Trust as a facilitator in cloud computing: a survey. *Journal of Cloud Computing: Advances, Systems and Applications*, 1(1):19.

Habib, S. M., Ries, S., Mühlhäuser, M., and Varikkattu, P. (2014). Towards a trust management system for cloud computing marketplaces: using caiq as a trust information source. *Security and Communication Networks*, 7(11):2185–2200.

Habib, S. M., Varadharajan, V., and Mühlhäuser, M. (2013). A framework for evaluating trust of service providers in cloud marketplaces. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1963–1965. ACM.

Instaclustr (2017). Service level agreements - cassandra services. https://www.instaclustr.com/company/policies/service-level-agreements/[Online; accessed March 26,2018].

Instaclustr (2018). Instaclustr - monitoring api. https://www.instaclustr.com/support/api-integrations/api-reference/monitoring-api/. [Online; Accessed April 22, 2018].

Kai, H., Chuanhe, H., Jinhai, W., Hao, Z., Xi, C., Yilong, L., Lianzhen, Z., and Bin, W. (2013). An efficient public batch auditing protocol for data security in multi-cloud storage. In *ChinaGrid Annual Conference (ChinaGrid), 2013 8th*, pages 51–56. IEEE.

Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 640–651, New York, NY, USA. ACM.

Khan, S. M. and Hamlen, K. W. (2012). Hatman: Intra-cloud trust management for hadoop. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 494–501. IEEE.

Kim, J. and Yi, J. (2010). A pattern-based prediction: An empirical approach to predict end-to-end network latency. *Journal of Systems and Software*, 83(11):2317 – 2321. Interplay between Usability Evaluation and Software Development.

Li, A., Yang, X., Kandula, S., and Zhang, M. (2010). Cloudcmp: Comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 1–14, New York, NY, USA. ACM.

Li, W., Ping, L., Qiu, Q., and Zhang, Q. (2012). Research on trust management strategies in cloud computing environment. *Journal of Computational Information Systems*, 8(4):1757–1763.

Li, X. and Du, J. (2013). Adaptive and attribute-based trust model for service-level agreement guarantee in cloud computing. *IET Information Security*, 7(1):39–50.

Lins, S., Schneider, S., and Sunyaev, A. (2016). Trust is good, control is better: Creating secure clouds by continuous auditing. *IEEE Transactions on Cloud Computing*.

Madhyastha, H. V., Anderson, T., Krishnamurthy, A., Spring, N., and Venkataramani, A. (2006). A structural approach to latency prediction. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 99–104. ACM.

Microsoft Azure (2015). Sla for storage - azure. https://azure.microsoft.com/en-us/support/legal/sla/storage/v1_0/[Online; accessed March 26, 2018].

Microsoft Azure (2018). Overview of metrics in Microsoft Azure. https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-overview-metrics. [Online; Accessed April 22, 2018].

MongoDB (2018a). Monitoring a cluster. https://docs.atlas.mongodb.com/monitor-cluster-metrics/. [Online; Accessed April 22, 2018].

MongoDB (2018b). Using the MMS console. http://api.mongodb.com/mms/0.8/usage.html. [Online; Accessed April 22, 2018].

MongoDB Atlas (2018). Sla for mongodb atlas. https://www.mongodb.com/cloud/atlas/availability-sla[Online; accessed March 26, 2018].

Muchahari, M. K. and Sinha, S. K. (2012). A new trust management architecture for cloud computing environment. In *2012 International Symposium on Cloud and Services Computing*, pages 136–140.

Nguyen, H., Shen, Z., Tan, Y., and Gu, X. (2013). Fchain: Toward black-box online fault localization for cloud systems. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 21–30. IEEE.

Oracle (2018). Monitoring and tuning the database. https://docs.oracle.com/cd/B19306_01/server.102/b14196/montune.htm#ADMQS010. [Online; Accessed April 22, 2018].

Oracle Cloud (2018). Oracle cloud infrastructure service level agreement. https://cloud.oracle.com/en_US/iaas/sla[Online; accessed March 26, 2018].

Popa, R. A., Lorch, J. R., Molnar, D., Wang, H. J., and Zhuang, L. (2011). Enabling security in cloud storage slas with cloudproof. In *USENIX Annual Technical Conference*, volume 242, pages 355–368.

Raaijmakers, L. e. a. (2018). Monitoring platform: prototype implementation.

Rafique, A., Van Landuyt, D., and Joosen, W. (2018). Persist: Policy-based data management middleware for multi-tenant saas leveraging federated cloud storage. *Journal of Grid Computing*.

Rafique, A., Van Landuyt, D., Reniers, V., and Joosen, W. (2017). Towards an adaptive middleware for efficient multi-cloud data storage. In *Crosscloud'17 Proceedings of the 4th Workshop on CrossCloud Infrastructures & Platforms*.

Redis Labs (2018). What is the redis labs service level agreement? https://redislabs.com/faqs/what-is-the-redislabs-service-level-agreement-2/[Online; accessed March 26, 2018].

RedisLabs (2018). Redis enterprise software - operations and administration guide. https://redislabs.com/redis-enterprise-documentation/administering/monitoring-metrics/definitions/. [Online; Accessed April 22, 2018].

Reniers, V., Rafique, A., Van Landuyt, D., and Joosen, W. (2017). Object-nosql database mappers: a benchmark study on the performance overhead. *Journal of Internet Services and Applications*.

Rong, C., Nguyen, S. T., and Jaatun, M. G. (2013). Beyond lightning: A survey on security challenges in cloud computing. *Computers & Electrical Engineering*, 39(1):47–54.

Sakr, S. and Liu, A. (2012). Sla-based and consumer-centric dynamic provisioning for cloud databases. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 360–367.

ScaleGrid (2012). Monitoring mongodb instances using mongodb monitoring service (MMS). https://scalegrid.io/blog/monitoring-mongodb-instances-using-10gen-mongo-monitoring-service-mms/. [Online; Accessed April 22, 2018].

ScaleGrid (2016). Master services agreement. https://scalegrid.io/msa.html[Online; accessed March 26, 2018].

Schoonjans, A., Van Landuyt, D., Lagaisse, B., and Joosen, W. (2015). On the suitability of black-box performance monitoring for sla-driven cloud provisioning scenarios. In *The 14th Workshop on Adaptive and Reflective Middleware*. ACM.

Uikey, C. and Bhilare, D. (2013). A broker based trust model for cloud computing environment. *International Journal of Emerging Technology and Advanced Engineering*, 3(11):247–252.

Wood, T., Shenoy, P. J., Venkataramani, A., Yousif, M. S., et al. (2007). Black-box and gray-box strategies for virtual machine migration. In *NSDI*, volume 7, pages 17–17.

Xiong, P., Pu, C., Zhu, X., and Griffith, R. (2013). vperfguard: an automated model-driven framework for application performance diagnosis in consolidated cloud environments. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, pages 271–282. ACM.

Yamamoto, H., Ano, S., and Yamazaki, K. (2015). Modeling of dynamic latency variations using autoregressive model and markov regime switching for mobile network access on trains. *Journal of Information Processing*, 23:420–429.

Yang, K., Jia, X., et al. (2013). An efficient and secure dynamic auditing protocol for data storage in cloud computing. *IEEE Trans. Parallel Distrib. Syst.*, 24(9):1717–1726.

Zhang, H., Ye, L., Shi, J., Du, X., and Guizani, M. (2014). Verifying cloud service-level agreement by a third-party auditor. *Security and Communication Networks*, 7(3):492–502.