Pick Your Choice in HBase: Security or Performance

Frank Pallas, Johannes Günther, David Bermbach
Information Systems Engineering Research Group
TU Berlin
Berlin, Germany
Email: {fp,jg,db}@ise.tu-berlin.de

***Abstract*—When analyzing sensitive data in a cloud-deployed Hadoop stack, data-in-transit security needs to be enabled, especially in the underlying storage tier. This, however, will affect the performance of the system and may partially offset the cost benefits of the cloud.**

In this paper, we discuss two strategies for securing HBase deployments in the cloud. For both, we present benchmarking results which show performance impacts that significantly exceed the suggested 10% from the official documentation. These results demonstrate (i) that security configurations should follow a rational decision process based on benchmarking results and (ii) that the security architecture of HBase/HDFS should be redesigned with an emphasis on performance.

***Keywords*-Benchmarking; HBase; Performance; Security**

I. Introduction

Big data technology can provide key business insights to enterprises of all sizes. However, until a few years ago, this was only available to big corporations that could afford the necessary infrastructure for a data warehouse or big data cluster in their local data center.

For smaller players, (public) cloud-based deployments can serve as low risk door openers to the world of big data. However, even big businesses can benefit from cloud-based deployments: big data use cases which need compute resources periodically (e.g., a bank running complex risk analyses once a month) or for variably-sized data sets (e.g., a retail company analyzing sales details right before Christmas and a week later) are a natural fit for cloud deployments that offer affordable, pay-as-you-go resources with instant scalability and high availability wherever and whenever needed.

At the same time, though, cloud-based deployments require additional security precautions due to data privacy regulations or simply for protecting core business interests. However, the performance impact of securing the big data cluster may offset the original cloud benefits – an aspect that has not been focused on in big data research yet. For instance, the official Apache HBase documentation only mentions an estimated performance impact of approximately 10% arising from the activation of built-in native security mechanisms [1, Section 58.3] which, as we will later see, is usually incorrect and also considers HBase native security as the only option. At the same time, cloud computing research, e.g., [2], has found interesting effects when enabling transport security – comprising a broad spectrum of both negative *and positive* performance impacts depending on the specific configuration.

In this paper, we focus on HBase, – the NoSQL system underlying the Hadoop ecosystem – and the performance impact of enabling security features therein, specifically of enabling transport encryption to ensure confidentiality of data in transit. We explicitly consider the big data engines on top of HBase, e.g., Hadoop or Spark, beyond the scope of this paper. We, therefore, present the following contributions:

- A thorough discussion of different strategies for ensuring data confidentiality for cloud-based HBase deployments
- A comprehensive experimental evaluation of the performance impact incurred by following a specific strategy
- A discussion of lessons learned and recommendations for big data deployments in public clouds

This paper is structured as follows: Based on a realistic application scenario and some foundations on cloud security and the HBase architecture provided in section II, we present two possible approaches for securing data in transit to, from, and within an HBase cluster deployed in a public cloud in section III. Our experiments for determining the performance impact of these approaches and the respective results are presented in section IV and discussed in section V. We close with related work (section VI) and a conclusion (section VII).

II. Foundations

In this section, we will give an overview of the application scenario behind our work and resulting security requirements in cloud environments, before presenting a brief introduction to HBase and its relevant communication channels.

A. Application Scenario

For our considerations, we assume the scenario of a mid-sized DIY store chain that already operates a comparably small centralized data center that serves both the ERP system as well as data analysis tasks. As storage backend, the store uses HBase. Due to recent growth in business, the ERP is affected by increased latency leading to delays at checkout and thus to decreased customer satisfaction. Furthermore, the insufficient capacity of the storage backend also impairs business intelligence through significantly longer data analysis runtimes.

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

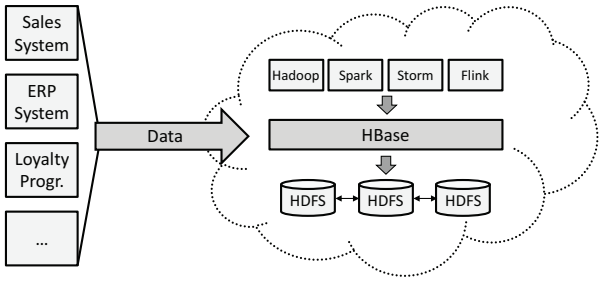

Figure 1. Considered application scenario

Instead of investing in additional hardware, the store chain plans to partially migrate to the cloud. Besides other benefits, this supports a cost-efficient mode of operation in which only a few active instances keep running HBase continuously while additional instances are provisioned for data analysis jobs with Hadoop, Spark, Storm, or Flink whenever necessary.

Given its omnipresence in the big data context and the number of compatible connectors, components and tools available, the store chain plans to stick with the Hadoop ecosystem and thus to store all data in a cloud-deployed HBase cluster running on top of HDFS for both transaction processing as well as big data analytics. Figure 1 illustrates this scenario.

B. Cloud Security and the Need for Data in Transit Security

While a cloud-based approach promises obvious economic benefits in this scenario, it also raises additional concerns in matters of security. For example, customer data must be appropriately protected due to privacy/data protection laws, sales data represents valuable business secrets that must not be revealed to other players, and data integrity must be ensured to prevent misleading analyses. Cloud computing here necessitates a variety of specific security precautions. Depending on the particular use case and risk model, this can include measures for secure data storage (*"data at rest security"*), mechanisms explicitly addressing multi-tenancy issues, or even novel approaches to audit and compliance management [3].

In the context of cloud security, concepts based on fully homomorphic encryption [4] promise to facilitate the use of cloud computing even when the cloud provider is not trusted [5]. However, this comes with such a performance penalty that it typically offsets the entire economic cloud benefits [6]. We will, therefore, in the following assume the cloud provider to be sufficiently trusted.

A core challenge in our scenario but also for cloud deployments in general is to ensure *"data in transit security"*. Even with a trustworthy cloud provider, data in transit to and from the cloud as well as between different components of a system (e.g., different nodes of a cluster) must be protected against threats like eavesdropping or data manipulation as they make use of public networks.

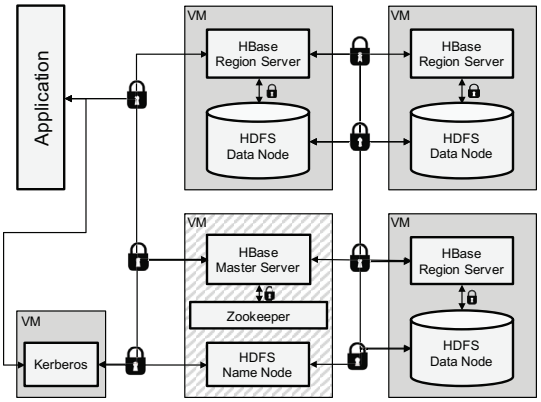

Figure 2. HBase deployment using HBase native security

C. HBase Architecture and Communication Channels

A typical HBase installation has a single *master server* that manages an arbitrary number of *region servers* which process client requests. Each region server controls a number of *data nodes* which store the actual data. To simplify maintenance, the HBase master commonly runs on top of *Apache Zookeeper*, a high level coordination service [7, p. 33]. Clients first request the corresponding region server's address from Zookeeper before directly interacting with the respective region server for reads and updates.

Master server, region servers and data nodes communicate with each other on two logical layers: the HBase-specific layer and the underlying HDFS-layer. Of these, the HBase-layer communicates via the HBase RPC protocol while the HDFS layer uses Hadoop RPC and the HDFS data transfer protocol. External clients communicate with the master server and the region servers either via native HBase RPC or through REST APIs, whereas the prevalent approach is to use tools that communicate via HBase RPC for reasons of efficiency. We will therefore explicitly not consider REST APIs herein.

The communication channels to be protected for cloud-based HBase deployments like the one assumed in our scenario are thus the cluster-internal traffic on the HDFS layer (Hadoop RPC and HDFS data transfer) and cluster-internal as well as -external communication via HBase RPC.

III. Securing HBase Communication in the Cloud

There are two main approaches for securing communication with and within cloud-based HBase clusters: HBase native security and an HBase-independent architecture based on virtual private clouds (VPC) and VPN. Both will now be introduced briefly.

A. HBase Native Security

The most important security mechanism in HBase is the so-called "wire encryption" between different nodes as well as between HBase nodes and external clients. HBase's wire encryption rests upon the well-established Kerberos protocol

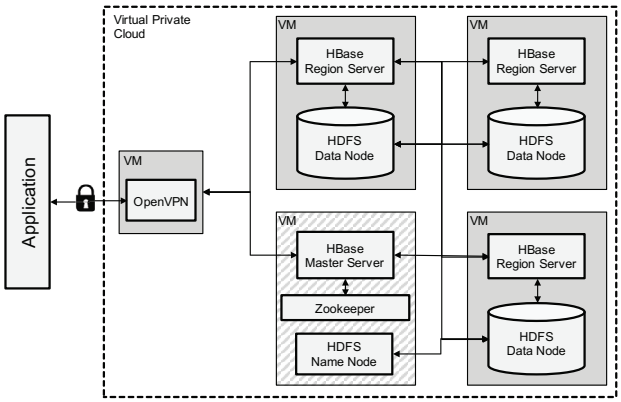

Figure 3. HBase deployment using a VPN-protected virtual private cloud

and architecture [8] for key distribution, authentication and ticket-issuing and thus necessitates to extent deployments by a dedicated Kerberos service.

A client that wants to access an HBase node in this setting obtains a ticket from the Kerberos service which delivers a session key to authenticated clients. Between nodes, in turn, shared secrets are established which are used for encrypting cluster-internal and traffic. In this regard, HBase-specific communication on the one and communication on the HDFS-layer (Hadoop RPC and HDFS data transfer) on the other hand are treated independently. Encryption can thus be activated and configured separately for the different layers [9, 209-210]. Activating encryption for external communication on the HBase layer automatically also enables cluster-internal encryption in this layer. As stated above, the secure operation of HBase in the cloud requires both layers to be protected and thus both mechanisms to be activated. Figure 2 illustrates this approach.

According to the official documentation, HBase wire encryption is supposed to have a performance impact of approx. 10% [1, Section 58.3]. However, this impact has so far not been backed by experiments and must be considered a rough estimate. In any case, such a performance impact has (in addition to the necessity for a separate Kerberos service) to be factored into decisions on the aspired system setup.

B. VPN-protected Virtual Private Cloud

Another approach to securing HBase deployments in the cloud is to operate the entire cluster within a VPC that is logically isolated from the rest of the cloud provider's infrastructure and only accessible through a dedicated network connection. Any traffic between the cluster and external clients uses this dedicated connection and can easily be secured against eavesdropping and manipulation through standard VPN solutions. Due to the strong isolation provided within VPCs, cluster-internal traffic between different nodes can be considered sufficiently secure without additional measures and, in particular, without using HBase native security (see figure 3).

Table I
APPROXIMATE CPU LOAD OBSERVED IN OUR EXPERIMENTS

Experiment Class	Data Nodes	Master	YCSB	VPN
No Security	80%	<5%	40-50%	–
HBase Native	85%	<5%	50-65%	–
VPC + VPN	20-25%	<5%	50%	30%

This deployment model promises several advantages over the native security approach discussed above: First, it reduces complexity as well as deployment and maintenance efforts, particularly regarding the operation and management of Kerberos and its counterparts on the other nodes of the cluster. Second, traffic encryption is then realized through specialized, mature solutions like OpenVPN, which have been extensively tested and optimized over the years. Regarding reliability and performance, this is an expectable advantage over HBase native security, which was only retrofitted after the main architecture had already been implemented.

On the other hand, a VPN-protected VPC requires additional resources for a dedicated VPN server and adds two additional intermediaries – VPN client and server – to the communication between clients and the cluster. Comparable to HBase native security, it is unclear how these will affect the overall performance of HBase. However, since both approaches can be considered sufficiently secure, factors like the performance impact imposed by the presented approaches should be one of these core decision factors for selecting an approach: While it is broadly accepted that security always comes at a cost, an approach that excessively impairs performance (or, respectively, requires significantly more provisioned resources for achieving the same performance) will be a worse option than another approach providing the same security level at a significantly smaller footprint. Making a rational decision between the two approaches laid out above thus requires reliable evidence on their respective performance impact. Furthermore, the same knowledge can also be pivotal for the decision whether to deploy HBase in the cloud at all: In case of an excessive security overhead, doing so could suddenly become economically infeasible.

IV. EXPERIMENTS

In this section, we will describe the results of our experiments with the overall goal of quantifying the performance impact of enabling one of the above presented approaches for securing HBase. We will start by describing the experiment design, followed by the experiment setup before presenting experiment results. In addition, we also present the results of several supplementary experiments we conducted to better understand the impact of select influence factors.

A. Experiment Design

With our experiments we aimed to answer three basic questions:

- Q1: What is the performance impact in terms of maximum throughput of enabling native security in HBase?
- Q2: What is the performance impact in terms of maximum throughput of using a VPN-protected VPC deployment?
- Q3: Is this potential impact constant for different cluster sizes, i.e., is there a scalability impact through the different security options?

To answer these questions, we decided to measure maximum throughput of HBase in three configurations (no security as a baseline value, HBase native security, VPC) for three different cluster sizes. Each conducted experiment was repeated until we were sure to have found reproducible results. With extensive monitoring, we asserted that none of the "special" nodes, i.e., HBase master, HDFS name node, Kerberos, VPN server, Zookeeper, and (most importantly) the benchmark client, became a resource bottleneck during our experiments.

For benchmarking, we used YCSB [10], the current state of the art database benchmarking tool. As a workload we selected workload A which offers a 50:50 ratio of reads to writes and is thus fairly representative for our presented use case from figure 1, but which is also comparable to existing database benchmarking papers, e.g., [2], [11]–[13]. For larger cluster sizes, we increased the number of operations in each benchmark to assert that each experiment ran sufficiently long. We used OpenVPN for the VPC configuration.

B. Experiment Setup

For our experiments, we used Amazon EC2 and deployed all systems in the same availability zone of the region us-east-1. For the VPC experiments we used Amazon VPC. For the placement of components on virtual machines, we followed best practices from literature (e.g., [7], [11], [14], [15]) and deployed the components as described in figures 2 and 3; YCSB took the role of the "application" in those figures.

We used t2.small instances for Kerberos, m3.small for the VPN server, c4.xlarge (experiments with 3 and 6 data nodes) and c4.2xlarge (experiments with 12 data nodes) for YCSB, and m4.large for all other machines (data node/regionserver and master/Zookeeper/name node). Depending on the respective experiment we had either 3, 6, or 12 machines running the combination of data node and regionserver. Again, we used extensive monitoring to avoid having performance bottlenecks in any non-regionserver machines. Except for the VPC case, which we will discuss in some more detail later, the observed CPU loads approximately depicted in table I are as intended.

C. Experiment Results

Following the basic questions laid out above, we first wanted to know the performance impact raised in terms of maximum throughput by activating HBase native security. For this aim, we started with our medium cluster size (6 data nodes) and measured an overall average throughput of 18,215 ops/s for the baseline setting without security. With HBase native security enabled we observed an average throughput of 9,643 ops/s – a decrease of 47% that goes way beyond the 10% mentioned in the official HBase documentation [1, Section 58.3]. Figure 4 illustrates this result.

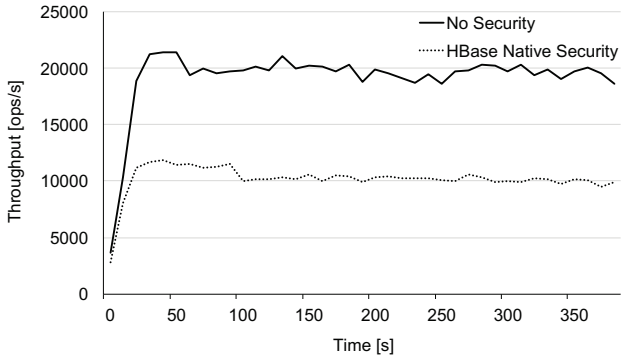

Figure 4. Exemplary throughput impact of HBase native security for a cluster with 6 data nodes

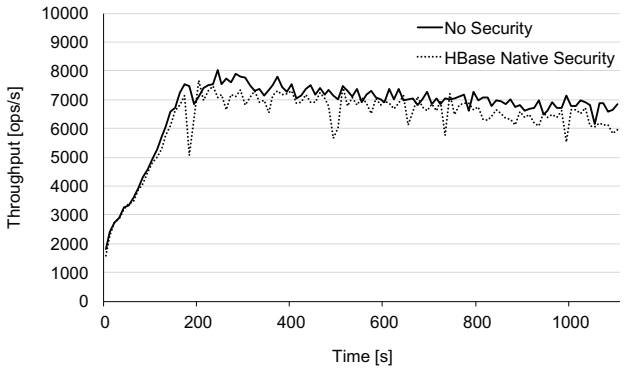

Figure 5. Exemplary throughput impact of HBase native security for a cluster with 3 data nodes

Given this significant impact of HBase native security, the second approach of a VPN-protected VPC becomes increasingly interesting. For this configuration, however, we got even worse results with an average throughput of 5,243 Ops/s for the same cluster size with 6 data nodes. In comparison with the baseline configuration, this corresponds to a performance impact of more than 70%. As such impacts will simply be inacceptable in practice, we abstained from benchmarking this configuration for other cluster sizes at all.

For HBase native security, however, we still wanted to know whether the performance impact is constant or varies for different cluster sizes. Especially in the light of the officially communicated 10%, we thus conducted respective experiments for 3 and 12 data nodes as initially planned.

For the cluster with 3 data nodes, we got an overall average throughput of 6,688 ops/s in the baseline configuration, which dropped to 6,205 ops/s (a decrease of only 7%) with HBase native security enabled (see figure 5). At least for this cluster

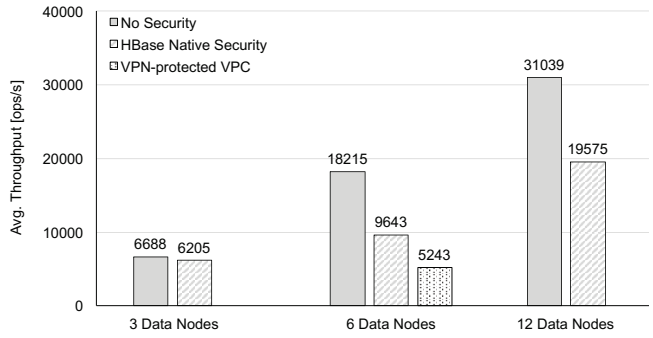

Figure 6. Compared average throughput results for no security, HBase native security, and VPN-protected VPC

size, our results are thus roughly in line with the official HBase documentation. Notably, our cluster with 3 data nodes and HBase native security enabled still performed better in matters of maximum throughput than our VPN-protected VPC configuration with 6 data nodes. Finally, the cluster with 12 data nodes showed a drop in average throughput from 31,039 to 19,575 ops/s or 37% with HBase native security.

Altogether, we thus found the impact of HBase native security in matters of maximum throughput to strongly depend on the chosen cluster size. With larger, more realistic cluster sizes, impact becomes significantly higher than the 10 % from the official documentation. Interestingly, however, the jump from 7% (3 data nodes) to 47% (6 data nodes) does not perpetuate but rather softens down to 37% for the cluster with 12 data nodes. The configuration with a VPN-protected VPC, in turn, showed a throughput decrease of more than 70% for 6 data nodes and was thus not pursued further for other cluster sizes. Figure 6 summarizes these results.

D. Additional Experiments

In order to learn more about the effect of further factors like instance types and to better localize the source of the observed performance impacts, we conducted a set of additional experiments within a more flexible and interactive – yet more inaccuracy-prone – setting. This setting consisted of 3 instances each of which again held a region server and a data node. Different from our above-mentioned experiments, however, the "master" components (name node, Zookeeper, HBase master) were also placed on one of these instances in addition to the region server and the data node. To heighten flexibility and to ease the collection of metrics, we configured, deployed, and monitored our nodes through Apache Ambari in this setting. The Ambari server was also placed on the "master" instance while the Ambari metrics collector was installed on one of the two instances that ran the region server and the data node. As in our above experiments, however, YCSB and Kerberos were deployed on separate instances.

Due to the higher flexibility provided within this setting, we were able to easily benchmark multiple variants of this configuration. In particular, we conducted experiments on m3.medium and r3.large instances. To identify whether the observed performance impact arises on the HDFS- or on the HBase-layer (see section II-C above), we furthermore tested the additional security configuration with HDFS-layer security turned on but HBase RPC encryption left inactive. As in our main experiments, we repeated each experiment several times to avoid random effects and monitored the YCSB and Kerberos instances to avoid bottlenecks here.

Again, we experienced a significant performance impact for the configuration with both security layers turned on for both instance types. In contrast to our main experiments outlined above, however, this performance impact already arose for clusters with 3 data nodes: Throughput dropped from 2,071 to 1,293 ops/s (38%) on m3.medium instances and from 12,958 ops/s to 7,243 ops/s (44%) on r3.large instances. While this basically confirms our finding that HBase native security entails a substantial performance impact, it raises the question why this impact can already be observed for clusters with three data nodes here. Possible explanations include the different component distribution as well as varios characteristics of the used instance types (processor generation, storage subsystem, etc.). Identifying the ultimate bottlenecks could be done through more sophisticated monitoring tools such as AISLE [16] in future experiments.

The experiments with the additional security configuration of only HDFS-layer encryption turned on, in turn, provided extremely illuminative insights: On m3.medium instances, this configuration resulted in a performance impact of only 11% (1,842 instead of 2,071 ops/s). The remaining 27% must thus be attributed to HBase's RPC encryption. On r3.large instances, this imbalance is even more prominent, with only 8% (11,936 instead of 12,958 ops/s) of the overall throughput reduction being caused by HDFS encryption and the remaining 36% arising because of HBase RPC encryption. Most of the observed performance impact is thus raised in the security implementation of HBase RPC, which is therefore an obvious candidate for closer examination in the future.

V. Discussion and Implications

Returning to the initial scenario from section II-A, choosing one of the strategies for securing HBase will affect the DIY store chain with extensive performance degradation. For cases requiring a given target throughput, this requires to provision more or bigger virtual machines (scale out or up).

This, however, can be rather costly: While we did not test scale up strategies explicitly, we could see that the maximum throughput of the 12 node cluster using HBase native encryption was similar to the throughput of the unsecured 6 node cluster. Using the prices of our setup on EC2, this corresponds to a cost increase of 90%. This sounds like a lot, however, the actual amounts are not that high for cloud deployments: Considering only the cost for the virtual machines, our unsecured 6 node cluster costs 0.84 USD/hour (i.e., 605 USD/month) while the secured 12 node

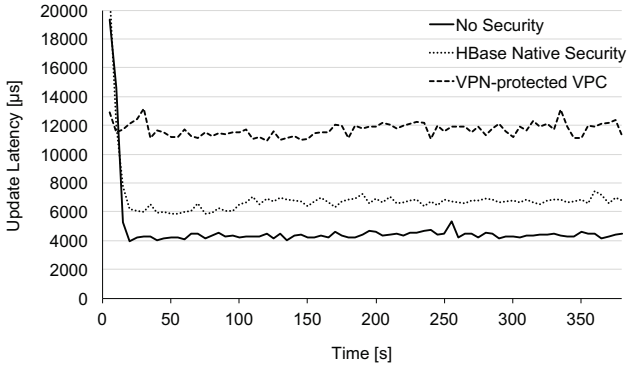

Figure 7. Update Latencies for 6 Data Node Clusters

cluster costs 1.59 USD/hour (i.e., 1145 USD/month). Both values should still be compared to another baseline, that of an on-premise deployment. This demonstrates that – at least for small to mid-sized clusters – a cloud deployment is still the more economical option for relatively short-running analysis tasks; even in the face of horrendous performance penalties when enabling then necessary data in transit encryption.

Another conclusion we draw from this is that it seems about time (a) to consider security of the Hadoop stack a key design goal – maybe more important than new analysis features – and (b) to consider rewriting or replacing the various RPC protocols of the Hadoop stack with a proven standard technology solution. As a comparison, Apache Cassandra uses a much simpler secure communication architecture based on standard TLS implementations [2]. We believe that the significant performance degradation observed for HBase native security could be avoided by using a standard and, thus, highly optimized TLS implementation instead of custom RPC protocols.

Depending on the respective use case, the VPC option should not entirely be disregarded: As illustrated in figure 7, the throughput reduction of the VPC option was mainly caused by dramatically increased latency values. This is due to the fact that YCSB internally uses a fixed-size thread pool for running the actual requests, i.e., these threads spent a lot of time waiting for responses when benchmarking the VPC option whereas they spent less time with idle waiting in the other two setups. We did not increase the thread pool size for YCSB to assert comparability across benchmark runs. However, for applications that communicate only asynchronously with the HBase cluster, the VPC option could in fact be feasible.

VI. Related Work

So far, there has not been a lot of work on quantifying the performance impact of using security mechanisms in HBase or other NoSQL systems: Müller et al. [2] developed TLSBench to analyze the performance impact of enabling transport layer encryption in Apache Cassandra and Amazon's DynamoDB service. Their findings indicate that potential performance impacts in DynamoDB are fully covered by the provider whereas performance impacts in Cassandra vary between low performance reductions, no effect at all, and even increased performance. Their approach, however, is not directly comparable to our approach since TLSBench is specifically designed for TLS-based communication to which the Hadoop stack uses a different, custom alternative.

The general security mechanisms available in Hadoop (including HBase) have been presented in [9], [17], albeit without an analysis of the resulting performance impact. Other benchmarks run against HBase, e.g., [11], [13], ignore security-related aspects. Configurations with client-side encryption enabled where only encrypted data is stored can only be used for a very limited number of use cases and it is unclear whether this includes big data scenarios. Existing benchmarking results of such setups, e.g., [18], effectively quantify the performance of the benchmarking client machine since client-side encryption is – aside of small size differences of the data items – fully opaque for underlying storage systems.

Other approaches for quantifying the performance impact of using data in transit encryption focus on different protocols and application domains such as web servers, e.g., [19], [20] or web services, e.g., [21]. Existing benchmarking approaches for NoSQL systems, in turn, do not consider security aspects, instead focusing on performance, e.g., [10], [22], novel cloud-specific benchmarks, e.g., [23], or consistency, e.g., [24], [25]. Finally, approaches like [16], [26] can complement our approach by providing insights into the quality of underlying infrastructure resources in the cloud.

VII. Conclusion

Running HBase in the cloud comes with a number of benefits. However, deploying HBase in the cloud requires additional security measures . In particular, data in transit within, to, and from an HBase cluster must be encrypted. Such security measures, though, come at a price – either in the form of reduced performance or as additional resources required for achieving a certain target performance. These costs are all too often unknown to system architects, preventing them from making rational choices between different design options and thus leading to undesirable outcomes.

Based on a realistic scenario comprising OLTP and OLAP aspects, we presented two approaches for data in transit security in cloud-deployed HBase clusters: one based on HBase native security and one based on a VPN with VPC. Through experiments we studied how these two approaches affect performance of HBase in various cluster sizes.

In particular, we found that using HBase native security leads to significant performance impacts of up to 47% for realistic cluster sizes of 6 or 12 data nodes. In terms of costs, this can result in an increase of up to 90% for achieving similar performance with security enabled. Being aware of

these numbers allows practitioners to make better-founded decisions whether to use HBase in the cloud or not.

When data in transit security is not mandatory, our results support better tradeoff decisions between risk-reduction and costs. Businesses considering cloud-based HBase deployments that significantly differ from ours could adopt our experiment-driven approach to make more rational decisions in the context of security based on own measurement results.

Finally, our results also provide guidance for future HBase development: Through additional experiments, we were able to attribute large portions of the observed performance degradation to the HBase layer, i.e., not to the underlying HDFS layer. The security implementations of HBase should thus be examined in more detail and probably be rewritten or replaced in the future. Given the ongoing trend towards cloud-based big data analysis and the necessity for data in transit security in most scenarios, reducing security overheads should be a top priority in future HBase development.

Until then, users of cloud-based HBase deployments have to pick their choice: Either they get a performant system – or a secure one.

ACKNOWLEDGMENT

The authors would like to thank Amazon Web Services who provided research grants for our experiments.

REFERENCES

[1] Apache Software Foundation. (2016) Apache hbase reference guide. [Online]. Available: https://hbase.apache.org/book.html#security.example.config

[2] S. Müller, D. Bermbach, S. Tai, and F. Pallas, "Benchmarking the Performance Impact of Transport Layer Security in Cloud Database Systems," in *Proc. of IC2E*. IEEE, 2014, pp. 27–36.

[3] Cloud Security Alliance, "Security guidance for critical areas of focus in cloud computing v3.0."

[4] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[5] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proc. of CCSW 2011*. ACM, 2011, pp. 113–124.

[6] Y. Chen and R. Sion, *Costs and Security in Clouds*. New York, NY: Springer, 2014, pp. 31–56.

[7] L. George, *HBase: The Definitive Guide*, 2nd ed. O'Reilly, 2015.

[8] C. Neumann, "The kerberos network authentication service (v5)," RFC 4120, 2005. [Online]. Available: https://www.ietf.org/rfc/rfc4120.txt

[9] B. Spivey and J. Echeverria, *Hadoop Security - Protecting Your Big Data Platform*. Sebastopol: O'Reilly, 2015.

[10] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," in *Proc. of SOCC*. ACM, 2010, pp. 143–154.

[11] J. Kuhlenkamp, M. Klems, and O. Röss, "Benchmarking Scalability and Elasticity of Distributed Database Systems," 2014, pp. 1219–1230.

[12] D. Bermbach, L. Zhao, and S. Sakr, "Towards Comprehensive Measurement of Consistency Guarantees for Cloud-Hosted Data Storage Services," in *Proc. of TPCTC*. Springer, 2014, pp. 32–47.

[13] T. Rabl, S. Gómez-Villamor, M. Sadoghi, V. Muntés-Mulero, H.-A. Jacobsen, and S. Mankovskii, "Solving Big Data Challenges for Enterprise Application Performance Management," 2012, pp. 1724–1735.

[14] G. Saloustros and K. Magoutis, "Rethinking hbase: Design and implementation of an elastic key-value store over log-structured local volumes," in *Proc. of ISPDC*, 2015.

[15] N. Dimiduk and A. Khurana, *HBase in Action*, 1st ed. Birmingham: Manning, 2012.

[16] J. Kuhlenkamp, K. Rudolph, and D. Bermbach, "AISLE: Assessment of Provisioned Service Levels in Public IaaS-based Database Systems," in *Proc. of ICSOC*. Springer, 2015, pp. 154–168.

[17] P. P. Sharma and C. P. Navdeti, "Securing big data hadoop: a review of security issues, threats and solution," *Int. J. Comput. Sci. Inf. Technol*, vol. 5, 2014.

[18] T. Waage and L. Wiese, "Benchmarking Encrypted Data Storage in HBase and Cassandra with YCSB," in *Proc. of FPS*. Springer, 2015.

[19] G. Apostolopoulos, V. Peris, and D. Saha, "Transport layer security: how much does it really cost?" in *Proc. of INFOCOM 1999*, 1999, pp. 717–725.

[20] L. Zhao, R. Iyer, S. Makineni, and L. Bhuyan, "Anatomy and performance of ssl processing," in *Proc. of ISPASS 2005*, march 2005, pp. 197–206.

[21] S. Shirasuna, A. Slominski, L. Fang, and D. Gannon, "Performance comparison of security mechanisms for grid services," in *Proc. of GRID 2004*, 2004, pp. 360–364.

[22] M. Klems, D. Bermbach, and R. Weinert, "A Runtime Quality Measurement Framework for Cloud Database Service Systems," in *Proc. of QUATIC*, 2012, pp. 38–46.

[23] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing, "How is the Weather Tomorrow?: Towards a Benchmark for the Cloud," in *Proc. of DBTEST*. ACM, 2009, pp. 1–6.

[24] D. Bermbach and S. Tai, "Benchmarking Eventual Consistency: Lessons Learned from Long-Term Experimental Studies," in *Proc. of IC2E*. IEEE, 2014, pp. 47–56.

[25] ——, "Eventual Consistency: How Soon is Eventual? An Evaluation of Amazon S3's Consistency Behavior," in *Proc. of MW4SOC*. ACM, 2011, pp. 1–6.

[26] A. H. Borhani, P. Leitner, B.-S. Lee, X. Li, and T. Hung, "WPress: An Application-Driven Performance Benchmark for Cloud-Based Virtual Machines," in *Proc. of EDOC*. IEEE, 2014, pp. 101–109.