# Cloud Storage Cost Modeling for Cryptographic File Systems

Mauro Storch
Sul-rio-grandense Federal Institute of
Education, Science and Technology
Gravatai, Brazil
Email: maurostorch@ifsul.edu.br

César A. F. De Rose
Pontifical Catholic University of Rio Grande do Sul
Computer Science School
Porto Alegre, Brazil
Email: cesar.derose@pucrs.br

***Abstract*—Nowadays, data security is a demand for companies when adopting storage services on public clouds. From long term persistence services, such as Amazon Glacier[1], to online block storage systems for virtual machines disks, security principles can be part of the cloud context, especially for customer's sensitive data. The confidentiality of storage services considers aspects such as data life-cycle, location, and size, besides that, this principle is often provided by a cryptography mechanism applied in one of the persistence layers, such as in the File-System (FS). However, to add cryptography for data security demands extra CPU cycles for ciphering the data during its persistence. Although these extra CPU cycles are not considered on current cloud costs estimations, it should be part of the total application execution's costs. This paper presents the architectures for Cryptography File Systems (CFS) adoption for data storing in cloud computing. Furthermore, a mathematical model is presented and discussed as an estimation tool of cryptography overhead when using CFSs in the cloud storage stack. The model is verified in a real scenario for estimating the total cost when adding security for storage in a cloud environment. As main result, the model could estimate the overhead within 90% to 92% of accuracy for the AES algorithm, according to real cases traces, considering available memory, I/O throughput and workload size.**

***Index Terms*—Cost Modeling, Cryptographic File System, Cloud Storage**

## I. Introduction

Storage performance is critical in most data-intensive application due to the nature of the I/O sub-system in computer architectures [1]. By adding any extra delay in the persistence work-flow should be carefully analyzed and measured to estimate either application performance impact or extra computational resources demand.

In order to provide the confidentiality security principle to data stored in the cloud, it is necessary to either encrypt data before sending to the cloud storage or encrypt data during the persistence phase using extra cloud resources. Once an user's application runs in the cloud, it is possible to consider two scenarios for data confidentiality, handling encrypted data through techniques such as homomorphic encryption [2], or encrypt and decrypt the data every time it is used by the user's application inside the cloud.

Most providers support cloud side encryption for data persistence in a transparent manner for users with no extra costs. However, this transparency gives users no control of the cryptography keys, which is not considered trustful [3]. Alternatively, customers could encrypt data inside their virtual machine instance before persisting it into storage layer. The encryption could be provided by either the application itself, which demands application modification, or provided transparently by a cryptography file system [4] (also know as CFSs).

The CFSs allow standard cryptography algorithms and its main roles are encrypting and decrypting data during the application's I/O operations. It is transparent for the application and still under control of customer, since it is handled by virtual machine internal software stack.

However, the CFSs need extra CPU cycles to provide confidentiality and it impacts in the whole virtual machine's CPU consumption. In other words, the CFSs add extra costs to customers renting the cloud. In doing so, a model representing the CFSs overhead in cloud environments will be part of customers' decisions for confidentiality adoption, cloud costs estimation, and so on. The model presented in this paper considers aspects such as the I/O sub-system throughput, the available memory in the host for handling the ciphering, and also the relation of those features with the total data size. Those axis are measured and a formula is produced for security overhead estimation.

The rest of the paper is organized as following, in next Section a Background and the Related Work are presented, following, in Section III, Cloud Storage principles and the security principles of storage systems are presented, focused in the Cryptographic File Systems. In Section IV the security cost model for cloud storage is presented and following, in Section V, its validation is described. Finally the conclusions and future works are presented in the last Section.

## II. Background and Related Work

Security has been in evidence since information leakage facts in recent years like NSA and Wikileaks cases [5]. For offline systems, a few techniques, such as keys and passwords, make most systems secure. For online systems, it is necessary to adopt cryptography, digital certificates or even complex authentication mechanisms. By considering software deployment on cloud environments it is necessary to review the security techniques to support companies' requirements.

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

2377-5750/17 $31.00 © 2017 IEEE
DOI 10.1109/PDP.2017.36

The principles of cloud computing made the security studies consider multi-tenancy scenarios, where different users could share the same virtualization stack to run their VMs. This scenario allows inter-VM attacks or even information leakage [6]. In addition to that, many other threats make managers consider cloud environments an unsafe solution.

The security solutions for data storage in cloud environments have been following standard patterns, such as using cryptography. The solution choice should provide confidentiality (where provider does not learn any information from users' data), integrity (any unauthorized modification of stored data should be detected by the customer), availability (customer's data should be available from anywhere at any time), reliability (customers' data should be backed up), efficient retrieval (data retrieval should be as efficient as in a standard public cloud), and data sharing (customers would share their data with trusted parties) [7]. Some cryptography techniques also provide the auditing support for cloud environment using a public key design for third-party auditing with no local data damage and adding minimal overhead [3]. Also, even the data deletion could be verifiable to avoid its distribution or leakage after company stops using external resources [8].

In order to support security for storage system, the Cryptography File Systems are solutions that apply standard ciphers algorithms in a transparent manner to upon layers such as the user's application. The concept of the CFSs are simple to understand, yet difficult to apply, often due to overhead impact in application's performance.

The privacy preserving techniques may be applied in different layers of the computational stack and each of them have their own specification for pros and cons [9]. In the Application layer, although it could be easy to deploy and adopt several kind of encryption algorithms, the performance is clearly damaged compared to lower layers (which is critical in storage systems). Following, if the privacy is applied in the File System layer there are still several algorithms and it is possible to offer it to upper layer in a transparent manner, however, once the file structure is created it becomes difficult to modify or revoke keys since it would be necessary to modify the entire list of files and directories. One layer bellow, the Block structure is responsible for storing raw data and yet it is possible to apply several different algorithms with higher performance comparing to upper layers, however nothing is kept in plain text in the File System which require longer operations for key revocation or modification. The lower layer in this stack is the hardware embedded cryptography. Although this layer offer higher performance then above layers, the cryptography characteristics, such as keys and algorithms, they are pre-defined reducing the security and privacy aspects. As a main conclusion, presented by the survey of Diesburg [9], the File System layer offer transparency and flexibility for the application layer, however the performance should be careful analysed in order to reduce overall impacts in the user's software execution.

Some variations on cryptography algorithm application could reduce the impact in terms of performance. The standard AES algorithm could be applied using the CTR mode which could work in parallel [10]. The AES adoption could be improved using processors with the AES-NI instruction set [11].

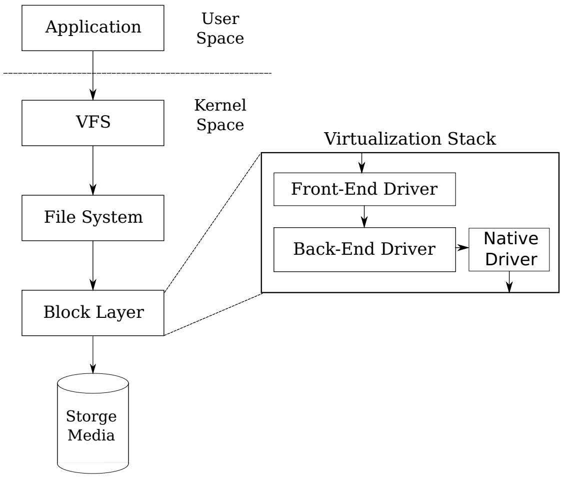

Fig. 1. I/O stack in Unix-like systems over a virtualized environment.

Even choosing the best setup for the encryption layer, it is clear the impact in performance when handling stored data. For Big Data environments [12], small datasets has low impact of the performance due to in-ram data processing, however in bigger datasets a high CPU demand is observed in nodes which handle the persisted data. This overhead needs to be clearly observed in order to measure and estimate the impact of cryptography in the I/O-dependent operations. To combine algorithms in a hybrid model is also an option trying to reduce the processing time in data access, but it is clear that an amount of CPU time is needed for storing data with confidentiality [13].

## III. Cloud Storage with CFS

Most cloud providers build their service on top regular data centers. The storage system is often provided by centralized per-rack storage unit, and on-server discs. The hardware storage is managed by the virtualization layer that places the VMs' (virtual machine) disc images according to rules of the virtualization player. These disc images are attached to VMs and they are accessed as regular disc by the operating system hosted inside that VM. Figure 1 shows a regular I/O stack for Unix-like systems, where in Block Layer the Virtualization technology add extra layers in order to handle shared resource among virtual machines. Even in a Virtualized environment the CFSs act in layers between the application and the Block Layer.

In the same hand, cloud providers offer storage as a service through internet-based communication, e.g. HTTP. Those systems are also called WebServices and they are also handled by the virtualization layer (to support service elasticity). Some of these services support security principles and they often uses cryptography.

Cloud storage systems are composed of layers which are implemented according to the application, i.e. a guest operating system hosted into a virtual disc which is managed by a virtualization layer over a storage service managed by a Network-Attached Storage. The confidentiality mechanisms can be applied to different levels of the storage stack, but one is enough to avoid data leakage.

When applying confidentiality for storing data inside the virtual machine, the user is responsible for choosing the cryptography algorithm and also for managing the keys, given to user total control of the security. The cryptography mechanism could be applied either internally in the user's software or using a persistence mechanism with cryptography support, i.e. the Cryptographic File Systems.

The Cryptographic File Systems, earlier introduced by *A Cryptographic File System for Unix*[4] and *The Design of a Cryptography Based Secure File System* [14], have been used to provide persistent data with confidentiality. The main feature of this mechanism consists in provide encrypted stored data with no application modification. The application uses standard I/O instructions, since the CFS works in lower system's layers.

Most CFSs uses symmetric cryptography due to performance issues and could be build as the following level of abstraction [1]:

### A. Block-based System

Block-based storage system have cryptographic support by handling one disk block at time. There is no knowledge of upper layers, such as files or directories, and it could even be used by software that needs to access raw partitions (such as databases). In other words, the persistence flow is intercepted by an cryptography phase which applies the ciphering to each block in the raw device, as shows Figure 2. In the Linux systems, the Device-mapper crypt [15] is a kernel module which provides block devices using the kernel's crypto API [16]. This API runs in the kernel space memory area and all instructions and keys are placed in the kernel space controlled area.

### B. Stackable File System

Stackable cryptography file systems are placed on top of different regular file systems and they are often portable for multiples operating systems, since the software layer which intercepts the data flow for encryption is independent of the persisted data. The ciphering process uses regular file system as the lower layer for hosting the directory-file structure and modify only the files content and optionally the file name (or directory name). Figure 3 shows a Cryptography Module running in User Space layer that handle application's files storing them into another File System. The EncFS [17] is one of this kind and was created as a virtual encrypted view for a directory in a regular file system in the user-space. It runs without any special permission and use the FUSE (File System in Userspace) library. The key management is made through user's configurations, but never stored, and it is prompted every time the encrypted directory is mounted. During the mounting phase, a new key is derived from the original within a certain number of rounds which is described in the user's configuration file. The primarily goal of this file system is to protect data off-line and support the usage of strong cryptography algorithms, such as AES and 3DES, for regular users. Similarly, the eCryptFS [18] build a cryptographic file system over a regular directory structure using algorithms such as the AES with different size of keys. In this case, the key is managed by a regular Linux keyring and the ciphering jobs is relayed to the kernel module Crypto API. Any algorithm supported by this kernel module is eligible to be used for the eCryptFS.

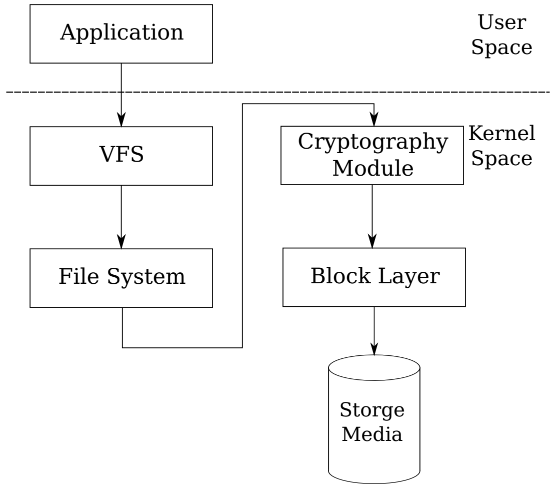

Fig. 2. I/O stack in Unix-like systems with Cryptography read/write directly in the block storage unit.

When comparing Block and Stackable models, the former could achieve better performance due to less layers between application and the storage device. However, a performance comparison between those two models should be evaluate according to the above application's behavior. Despite the model architecture impacts the overhead, the main portion of CPU load of their architectures is related to the cryptography algorithm. In doing so, following a model considering the algorithms used in CFSs is presented.

## IV. Storage Security Model

The storage privacy principles, supported by the cryptography file systems presented early, are often based in regular symmetric algorithm such as AES and BlueFish. However, the overhead measurement of storing data with privacy is not only base on the characteristics of the cryptography algorithm. It is necessary to consider the architecture of the cryptography file systems, the IO subsystem, as well as the encryption and decryption operations in such a system. The storage system often uses cache mechanisms for performance improvement that needs to be paired with the cryptography module.

Currently, the cryptography algorithms achieve higher performance specially using modern processors. In the other hand, the IO subsystem's performance still depends on physical

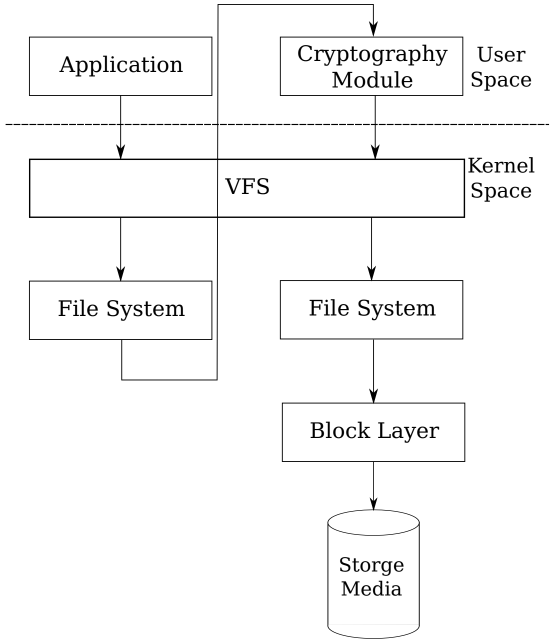

Fig. 3. I/O stack in Unix-like systems with Cryptography module read/write in a File System.

aspects for persisting data. These two aspects make the Cryptographic File Systems an alternative for storing data with confidentiality and with low CPU allocation.

But how much is the impact of CPU usage in such a system, how is the impact of Cryptography File Systems in a cloud environment, and what is the application's profile that match within a certain CPU costs are some of the questions that need to be answered for better allocation of cloud computing resources.

For this reason, one can consider to model the costs of using Cryptography File System in a cloud environment. The CPU usage estimation is important since cloud billing is commonly based in CPU units (such as cycles or hours).

As mentioned early, the CPU time used by the CFS is spread along the IO queue size. In doing so, one can first deduce a inverted relation of the total CPU time consumed by cryptography for a certain data amount and the time for persisting it. This relation can be writing as:

$$S_{(d,t)} = \frac{C(d)}{t} \quad (1)$$

where $d$ is a data amount, $C$ is the function for expressing the CPU time for that given data amount $d$, and $t$ is the total time of the persistence flow. The formula would be clear if a single one-way direction (reading or writing) is considered. However, it is necessary to identify and split the data amount $d$ in write and read operation to make it close to real scenarios. It is also necessary to consider a significant difference in reading and writing throughput to estimate the total time of the persistence process. In doing so the formula could be rewritten as:

$$S_{(w,r,TP_r,TP_w)} = \frac{E(w) + D(r)}{r \times TP_r + w \times TP_w} \quad (2)$$

were $w$ and $r$ are total data write and read, respectively, $E$ and $D$ are the function for estimating the CPU time of encrypting and decrypting an amount of data. These values will produce the total CPU time for ciphering process. Following, the $TP$ variables are related to the throughput of the reading and writing operations in IO subsystems. Although it seems to be clear spreading the cryptography's CPU time over the IO time, the machine's memory size has also a significant impact in the application's total time since the file system's cache subsystem allow better performance of the cryptography algorithms' utilization. So the memory size $m$ impacts the total time of the IO subsystem and it is represented as:

$$S_{(w,r,TP_r,TP_w,m)} = \frac{E(w) + D(r)}{\Delta(r/TP_r + w/TP_w)} \quad (3)$$

where the variable $m$ represents the available memory for the IO subsystem and needs to be considered for two scenarios: the total dataset size is smaller than the available memory, or not. The $\Delta$ is dependent of a condition within $m$ as:

$$\Delta = \begin{cases} F(r/TP_r + w/TP_w, m), & r + w < m \\ 1, & r + w >= m \end{cases} \quad (4)$$

where the $F$ function in the formula gives the index of the extra performance for in-memory operations. This behavior could be observed during experiments where file with size small than the host's available memory could achieve at least 3 times higher performance.

As a summary, the formula application would need the knowledge of variable values such as the amount of data read (r) and written (w), the function costs of encrypting (E) and decrypting (D) each kind of persistence flow, and the throughput of the IO subsystem. The throughput needs to be observed in two main perspectives: the read and write operations in to raw-external devices (virtual disc of virtualization layer is also included), and the read and write operations in to memory which will fit the $F$ function, as presented in the formula 4.

The presented model aims to produce as result the CPU allocation of a host (or virtual cloud host) for a certain data amount. This value helps to predict the host allocation when choosing to add security in the persistence for confidentiality guarantees.

The system administrator could trace company's application I/O data amount and use this formula to estimates the CPU allocation per node.

## V. Validation

In order to evaluate the proposed model, initially a set of experiments was conducted to fit the mathematical model, considering the impact of memory, the absolute CPU load and the relation between dataset's size and execution time. Finally, an experiment considering a real application scenario shows the impact of the cryptography for storing data in a cloud environment.

### A. Environment Description

The validation environment was composed by virtual machines running a standard Linux using a single virtual processor and variable memory sizes (256MB, 512MB, 768MB, and 1024MB). The virtual machines were hosted in a physical server with a 8-cores processor Intel Xeon X6550 2GHz, 96GB RAM, a local Sata disk with 128GB, and XenServer 6.2. The Cryptography File Systems used in the experiments are the dm-crypt [15], placed in the kernel level and acting as a Block CFS, and the EncFS [17] running in the user-space with FUSE library, acting as a Stackable CFS, and the algorithm is always set to AES with a 256 bits key. This algorithm is chosen due to it is recommended by security institutes and could be verified in the future using specialized processors [11]. These CFSs were evaluated by the IOZone [19] I/O benchmark tool.

### B. Evaluation

This evaluation defines the weights for the variables and functions from the Equation 3. The functions $E$ and $D$ in the formula should calculate the processing time of encrypting and decrypting an amount of data in bytes, respectively. Considering that the CFSs used in this validation use standard cryptography algorithm, this values could be captured form isolated test case (measured with command line tools, i.e.) and applied in a linear function. The values from the functions $E$ and $D$ for a 500MB file was respectively $4558ms$ and $3956ms$. One can notice the encryption operation expensive in terms of time compared to decryption. This operation is also essential during the writing processes which is far expensive than reading in terms of time for the I/O subsystem. When combining both encrypting and writing operations the worst scenario for this experiment is presented. So, next experiments would consider only writing operations in order to demonstrate the model application.

Following with the Equation 3, the throughput variables are determinant in the overall formula application. In order to figure out the behavior of such mechanism, an experiment was conducted to write files in a CFS from 10MB to 1000MB and the host CPU consumption is captured for four memory sizes 256MB, 512MB, 768MB and 1024MB. Figure 4 shows differences in CPU allocation according to the file size and the host memory size. In general, the CPU load was near $10\%$ for files with size up to $68\%$ of the available memory. This experiment makes clear the allocation of the memory during the encryption phase. Otherwise, for bigger files the I/O subsystem queue the work flow not allowing the progress of the CPU. In such scenario, the formula would be impacted by the host's memory size which is mapped in the mathematical model as the $\Delta$ factor. This factor is conditioned by the function in Equation 4, where the total execution time is affected by a factor $F$ when persisted data is bigger than than available memory size $m$. The reduced CPU load is also reflected in the total execution time, as it was observed during the tests executions.

The isolation of CPU load with and without using the memory as a support for improving the performance is demonstrated in the experiments when the CPU overhead chart was produced. Figure 5 presents a stabilized overhead of CPU for files bigger than 260MB and a increasing curve otherwise. This execution was produced in a virtual machine with 512MB RAM and the negative peaks are random and it was not considered in the model.

When modeling the overhead of CPU of the Cryptography File Systems, one may consider to draw a logarithmic curve for predicting this values, as presented in Figure 6.

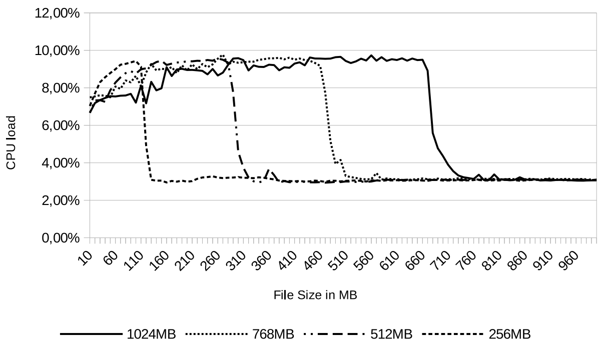

Fig. 4. Memory size

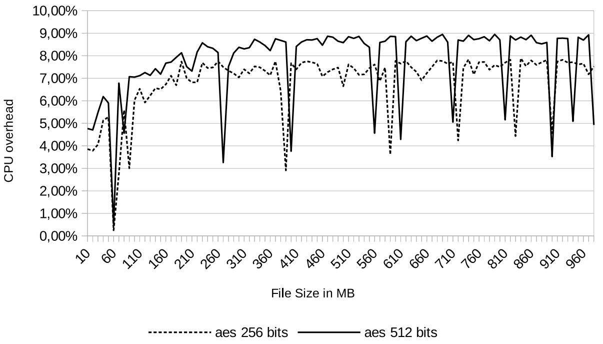

Fig. 5. CPU load

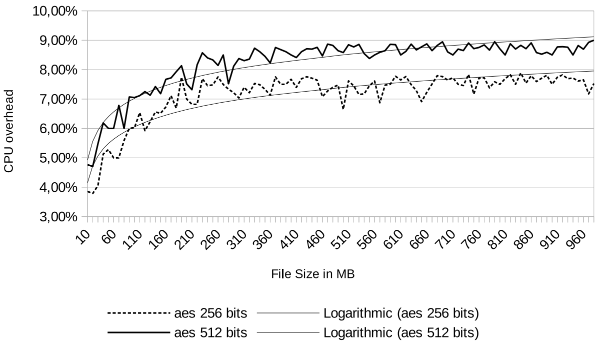

Fig. 6. Log CPU load

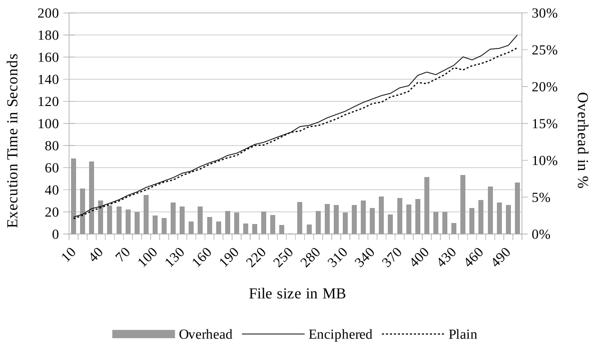

Fig. 7. Hadoop overhead by CFS: Word count running over different files sizes.

### C. Real Application validation

In order to evaluate the Cryptography File Systems supporting a real cloud application, a new environment was produced with two virtual machines (same image and CPU) with 2GB RAM each. The virtual machines hosts a set of Hadoop [20] components, enough for the execution of a map-reduce application. The datanodes was configured to read and write data into a CFS in all nodes. The workload was a 1MB file replicated with size from 10MB to 500MB with a set of words, which produces a linear workload. This linearity can be observed in the execution time line in Figure 7. It is also observed that the CPU overhead was no more than 10% in average. This overhead is an comparison of the environment with and without the usage of the CFS and confirms the estimation presented early in this paper.

## VI. Conclusions

Cryptographic File Systems have been used to provide data storing with privacy. However, they do not appear in cloud estimations since it is part of user's VM installation. Although most privacy solutions use regular cryptography algorithms to provide confidentiality for persisting data, there are more elements to be considered when estimating the extra overhead of such solutions in a cloud environment.

In doing so, this paper presented a mathematical model for extra CPU estimation when using Cryptographic File Systems. The model considers three main factors as essential in the overhead calculation: (a) the CPU load for encrypting and decrypting the data flow, (b) the memory size of the host machine, and (c) the throughput difference between raw-in-disk operations and in-memory operations.

The validation shows the behavior of CFSs considering the aspects explored in the model. Yet, it was possible to use the experimented values of CPU load, processing time, and workload size for fitting the model's variables. The fitted model was used to estimate the overhead of a CFS hosting files of an application execution (based on Big Data operations) and the model predictability accuracy was close to 90%.

As future experiments specialized file systems will be considered in order to investigate the best mechanism for data privacy storage for cloud computing, taking into account not only the throughput but also the cache hierarchy, synchronization mechanisms and different cryptography algorithms.

## Acknowledgment

The authors would like to thank CNPq (National Counsel of Technological and Scientific Development - Brazil), CAPES (Coordination for the Improvement of Higher Education Personnel - Brazil), FAPERGS and Sul-rio-grandense Federal Institute of Education, Science and Technology for financial support.

## References

[1] C. Wright, J. Dave, and E. Zadok, "Cryptographic file systems performance: What you don' t know can hurt you," in *Security in Storage Workshop, 2003. SISW '03. Proceedings of the Second IEEE International*, Oct 2003, pp. 47–47.

[2] C. Gentry, "Fully homomorphic encryption using ideal lattices." in *STOC*, vol. 9, 2009, pp. 169–178.

[3] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the 2010 IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[4] M. Blaze, "A cryptographic file system for unix," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, ser. CCS '93. New York, NY, USA: ACM, 1993, pp. 9–16. [Online]. Available: http://doi.acm.org/10.1145/168588.168590

[5] B. Toxen, "The nsa and snowden: Securing the all-seeing eye," *Queue*, vol. 12, no. 3, pp. 40:40–40:51, Mar. 2014. [Online]. Available: http://doi.acm.org/10.1145/2602649.2612261

[6] A. S. Ibrahim, J. H. Hamlyn-harris, and J. Grundy, "Emerging security challenges of cloud virtual infrastructure," in *Proc. of The Asia-Pacific Cloud Workshop – Co-located with APSEC2010*, Sydney, Australia, 2010.

[7] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Financial Cryptography and Data Security*. Springer, 2010, pp. 136–149.

[8] F. Hao, D. Clarke, and A. Zorzo, "Deleting secret data with public verifiability," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[9] S. M. Diesburg and A.-I. A. Wang, "A survey of confidential data storage and deletion methods," *ACM Comput. Surv.*, vol. 43, no. 1, pp. 2:1–2:37, Dec. 2010. [Online]. Available: http://doi.acm.org/10.1145/1824795.1824797

[10] M. J. Dworkin, "Sp 800-38a addendum. recommendation for block cipher modes of operation: Three variants of ciphertext stealing for cbc mode," Gaithersburg, MD, United States, Tech. Rep., 2010.

[11] I. Corp.", "Intel®Advanced Encryption Standard (AES) New Instructions Set." "Intel White Paper", 2006, https://software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf.

[12] L. Dupr and Y. Demchenko, "Impact of information security measures on the velocity of big data infrastructures," in *2016 International Conference on High Performance Computing Simulation (HPCS)*, July 2016, pp. 492–500.

[13] P. V. Maitri and A. Verma, "Secure file storage in cloud computing using hybrid cryptography algorithm," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, March 2016, pp. 1635–1638.

[14] E. Gudes, "The design of a cryptography based secure file system," *IEEE Transactions on Software Engineering*, vol. SE-6, no. 5, pp. 411–420, Sept 1980.

[15] C. Fruhwirth, "New methods in hard disk encryption," Tech. Rep., 2005. [Online]. Available: http://clemens.endorphin.org/nmihde/nmihde-A4-ds.pdf

[16] J. Corbet, "The 2.5 kernel gets crypto," 2002. [Online]. Available: http://lwn.net/Articles/14157/

[17] V. Gough, "EncFS: an Encrypted Filesystem for FUSE," 2014. [Online]. Available: https://vgough.github.io/encfs/

[18] M. Halcrow, "ecryptfs: A stacked cryptographic filesystem," *Linux J.*, vol. 2007, no. 156, pp. 2–, Apr. 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1242359.1242361

[19] W. D. Norcott and D. Capps, "Iozone filesystem benchmark," *URL: www. iozone. org*, vol. 55, 2003.

[20] Apache Foudation, "Hadoop project," http://hadoop.apache.org, 2013.