Technical Report
KN–2013–DiSy–01

Distributed Systems Group

Utilizing Cloud Storages for iSCSI: Is Security really expensive?

Sebastian Graf **Andreas Rain**
Daniel Scharon **Marcel Waldvogel**

Distributed Systems Group
Department of Computer and Information Science
University of Konstanz – Germany

Cloud storage promises unlimited, flexible and cheap storages, including all-time availability and accessibility with the help of various technologies. Free-of-charge offers for endusers allure customers the same way as professional, pay-as-you-go storages do. The delocalization of the data provokes security concerns especially regarding the confidentiality of the data. Even though encryption offers a straight-forward solution to this problem, the performance questions its applicability when it comes to the utilization of professional storage-approaches like iSCSI. In this white-paper, we propose a utilization of NoSQL-based cloud-storages like Amazon S3 or Microsoft Azure for iSCSI. We evaluate the costs of a direct, bucket-based encryption and show, that in complex systems like iSCSI, the distance to the cloud represents the bottleneck instead of the encryption. Performance-boosting techniques like prefetching and caching improve the access and result in no practical overhead within such an utilization. Based on our own developed fully Java-based iSCSI target (jSCSI) and jClouds, our prototype represents, to the best of our knowledge, the first, free available, cloud-deployable iSCSI.

Table of Contents

1 Introduction

Cloud-Speicher repräsentieren bei Endnutzern ebenso wie bei IT-Service Providern eine skalierende, einfache Lösung für komplexe Fragestellungen wie Synchronisierung, Erweiterbarkeit, Verfügbarkeit und Wartung. Die physikalische Verteilung der Daten stellt allerdings die Verwendung von cloud-basierten Speichermöglichkeiten in Frage, insbesondere da viele Cloud-Provider ihre Daten selbst weiterverteilen, zum Teil über Landesgrenzen und geltende Datenschutzgesetze hinaus.

Verschlüsselungslösungen wie z.B. BoxCryptor oder EncFS stellen Lösungen bereit, zumindest die Vertraulichkeit der Daten auf technischer Ebene zu garantieren, nur zu welchem Preis? Technische Verschlüsselung auf Dateisystemebene wie bspw. mit EncFS oder FileVault verlangsamt spürbar das System und erschwert Backups ebenso wie Indexierung der Daten.

Bei der Verwendung von Cloud-Speichern wirken sich diese Probleme weniger drastisch aus:

- Die Entfernung zur Cloud stellt einen Flaschenhals dar, so dass sich angewendete Verschlüsselungtechniken nicht signifikant auf die Performanz auswirken.
- Intelligente Techniken wie Caching oder Prefetching wirken sich in komplexen Architekturen zusätzlich günstig auf die Verschlüsselung entfernt gespeicherter Daten aus.

Die generierten Ergebnisse vergleichen die Kosten einer lokalen Verschlüsselung und die einer sicheren Verwendung der Amazon S3-Cloud. Während bei einem lokalen Zugriff die Performanzeinbußen durch Ver- und Entschlüsselung direkt sichtbar sind, wirken sich diese bei einer entfernten Speicherung nicht negativ aus. Anhand eines komplexeren Anwendungsfalls, in unserem Fall die Kombination von iSCSI[1] mit der Amazon S3-Cloud, zeigen wir, dass Verschlüsselung in diesem Fall keinen signifikanten Verlust darstellt.

2 Techniken zur sichere Verwendung von Cloud-Speichern

Cloud-Speicher bieten unterschiedliche Angebote zur Abspeicherung von Daten. Neben den Anwendungen für private Endverbraucher wie Dropbox, Google Drive und Wuala bieten viele Cloud Service Anbieter ein breites Portfolio an Diensten für professionelle Nutzer an. Für professionelle, anwendungszentrierte Datenspeicherung stehen verschiedene Arten von No-SQL-Stores zur Verfügung. Diese Speichermöglichkeiten bestehen in der Regel aus sog. Buckets welche anhand von eindeutigen Schlüsseln gespeichert und abgefragt werden können. Ironischerweise basieren viele Cloudspeicher-Angebote für Endnutzer auf diesen No-SQL-Speichern: Dropbox verwendet beispielsweise Amazon S3[2] als Speicherort und es ist davon auszugehen, dass Google Drive auch auf das professionellere Google Cloud Storage-System zugreift.

Die geschachtelte Architektur cloud-basierter Speicherung resultiert in folgenden Aspekten Performanz und Sicherheit betreffend: Die Daten sind in jeder Ebene abgreifbar und sollten daher verschlüsselt werden, um Vertraulichkeit zu wahren. Des Weiteren steigt die Komplexität der Systeme und damit der Overhead mit jedem Zugriff auf den Speicher mit jeder Schicht an, mit der Zugriffe entsprechend zugeordnet werden müssen.

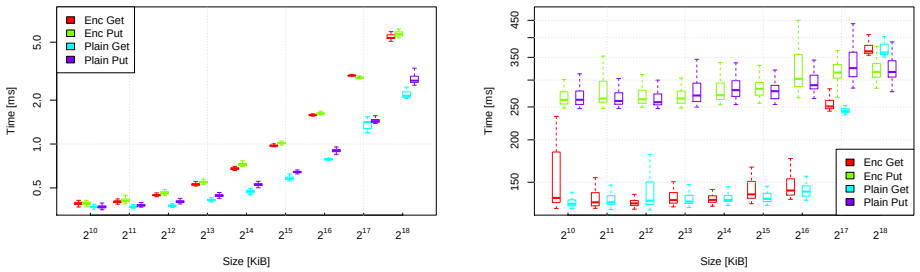
Um konsistente und durchgehende Verschlüsselung zu gewährleisten, sollten daher die Buckets direkt vor dem Schreibzugriff extern verschlüsselt werden.

Die Anwendung der Verschlüsselung sollte dabei nach Shannon's Maxim[1] offen verfügbar sein. Zusammengefasst, sollten folgende Eigenschaften eine Lösung für einen sicheren Cloud-Speicher aufweisen:

- Plattformunabhängigkeit:
  Jedes Betriebssystem sollte auf den Cloud-Speicher möglichst mit nativen Schnittstellen zugreifen können. Sofern eine zusätzliche Schnittstelle benötigt wird, sollte diese frei verfügbar und open-source sein.
- Anbieterunabhängigkeit:
  Die Implementierung soll nicht von einem einzelnen Cloud Storage Anbieter, z.B. AWS oder Azure abhängig sein. Die freie Wahl des Providers schließt vendor lock-ins aus und erlaubt einfache Speicherung der Daten in mehreren Clouds.
- Sicherheitskonformität:
  Die etablierten Sicherheitsmechanismen sollten transparent aber automatisch auf die Daten angewendet werden. Alle Buckets sollten verschlüsselt in den Cloud-Speicher geschrieben werden.

Unsere Tests basieren auf der Verwendung von iSCSI und Amazon S3 als Cloud-Speicher in verschlüsselter und unverschlüsselter Weise. Unseren Prototyp haben wir hierbei mit unserer eigenen, frei verfügbaren Java-Implementierung "jSCSI"[3, 4] und mit "jClouds" [5] implementiert. "jClouds" ist eine Java-basierte Bibliothek welche verschiedene REST-Dialekte von verschiedenen Cloud-Providern auf gemeinsame Methoden abbildet. Durch die Verwendung von "jClouds" wird dadurch eine einfache Anbietung an verschiedene Cloud-Speicher möglich.

2.1 Verschlüsselung von Buckets



(a) Direkter Zugriff auf Buckets im lokalen Dateisystem

(b) Direkter Zugriff auf Buckets in AWS

Fig. 1. Lesen und Schreiben lokal und auf AWS

Grafik 1 zeigt die Zeit von 100 Schreib- und Lesezugriffen mit verschiedenen Bucketgrössen von 2^{10} KiB bis zu 2^{18} KiB. Die Buckets werden hierbei verschlüsselt als auch unverschüsselt über "jClouds" abgelegt und angefragt.

[1] "One ought design systems under the assumption that the enemy will immediately gain full familiarity with them."

Grafik 1a zeigt die Boxplots von den 100 Durchläufen bei der Verwendung des lokalen Dateisystems unter "jClouds": Die Buckets sind hierbei einfache Dateien in einem festgelegten Ordner. Die Kosten der Verschlüsselung werden hierbei sowohl beim Schreibzugriff als auch beim Lesezugriff auf der logarithmisch skalierten y-Achse ersichtlich. Je grösser der Bucket, desto signifikanter steigen auch die Zugriffszeiten sowohl beim Schreiben, als auch beim Lesen an.

Grafik 1b stellt den gleichen Test mit der Amazon S3-Cloud dar. Im Gegensatz zu den lokalen Benchmarks sind keine signifikanten Unterschiede sichtbar. Dies liegt daran, dass der Performanzverlust bei der Ver- und Entschlüsselung in den Zugriffszeiten auf den entfernten Speicher untergeht. Zusätzlich sind die Lesezugriffe blockierend während bei den Schreibzugriffen der Speicher den Erhalt der Daten bestätigt ohne dass die Daten schon verarbeitet worden sind. Dieses nicht-blockierende Verhalten ist insbesondere bei grösseren Bucket-Grössen ersichtlich.

Einen signifikanten Performanzverlust bei der sicheren, direkten Verwendung von Cloud-Speichern ist daher nicht sichtbar und wirkt sich auch bei komplexeren Applikationen nicht aus, wie wir in Kapitel 2.2 am Beispiel von iSCSI und "jClouds" zeigen werden.

2.2 iSCSI und Cloud-Speicher

iSCSI[1] stellt eine Kombination von SCSI-Kommandos auf TCP-Ebene dar. Durch die breite, native Verwendung in praktisch allen aktuellen Betriebssystemen, ist es Nutzern erlaubt, block-basierte Speichermedien über ein Netzwerk zu alloziieren und mit jedem beliebigem Dateisystem zu verwenden. iSCSI benötigt durch das Versenden von SCSI-Befehlen via TCP/IP keine zusätzliche, spezielle Hardware, im Gegensatz zu beispielsweise Fiber Channel. Dadurch kann nicht nur bereits vorhandene Netzwerktechnik verwendet werden, sondern auch bestehende cloud-basierende Infrastrukturen. Zusätzlich bietet iSCSI auch in Zukunft für bestimmte Anwendungsfälle Eigenschaften, die dateibasierte Verfahren wie beispielsweise WebDAV nicht erfüllen können wie beispielsweise das Vorhalten von Backups für Disaster Recovery, d.h. der Spiegelung der lokalen Produktionsumgebung für das Szenario einer Notfallwiederherstellung. Das entfernte Vorhalten von kompletten Blockgeräten über Cloud-Speicher ist hierbei nicht nur ein weiterer Beitrag zu Ausfallsicherheit sondern zudem auch noch deutlich günstiger als lokal physische Platten vorzuhalten. In solch einem Fall ist auch die Skalierbarkeit bei plötzlichem Wachstum des Bedarfs gewährleistet.

Eine Möglichkeit hierfür bietet "Amazons Storage Gateway". Die Verwendung dieses Angebots ist jedoch vergleichsweise kostspielig. Allein die monatliche Gebühr für die Bereitstellung beträgt derzeit 125 US-Dollar. Hinzu kommen monatliche Gebühren für das verwendete Volumen und die Menge der übertragenen Daten.

In der Kombination des Setups von "jClouds" mit der Java-Implementierung "jSCSI" besteht jedoch die Möglichkeit, das reguläre Angebot von Amazon S3 zu benutzen. Die Grundidee ist ein Mapping von der vom Betriebssystem via iSCSI übertragenen Blöcken zu den Buckets im Amazon Cloud-Speicher . Das Resultat, welches wir in "jSCSI" implementiert haben[2] repräsentiert eine weitaus komplexere Konfiguration als der triviale Bucket-Zugriff dargestellt in Grafik 1.

[2] Frei verfügbar unter `http://jscsi.org`

Dementsprechend sollte sich der Performanzverlust durch Verschlüsselung auch nicht signifikant auswirken.

Die Benchmarks wurden jeweils mit dem für unixoide Betriebssysteme geschriebenen Programm *bonnie++* durchgeführt. Bei den jeweiligen Tests wurden die Standardparameter genommen, und Daten von 100 Durchläufen gesammelt. Benchmarks werden bei bonnie++ auf Dateisystemebene ausgeführt, d.h. es finden verschiedene Lese- und Schreiboperationen auf Dateien innerhalb eines Ordners statt.

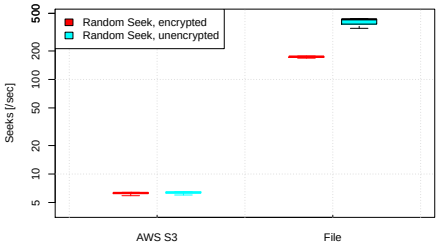


Fig. 2. Random Seek im Vergleich

Abbildung 2 zeigt den direkten Vergleich zwischen Random Seeks auf Cloud-Speicher via iSCSI und jenen auf einem lokalen Dateisystem. Auf der logarithmisch skalierten y-Achse werden die unterschiedlichen Seeks pro Sekunde dargestellt. Unsere Annahme wird hier deutlich bestätigt: Die Performanz ist beim lokalen Dateisystem wesentlich höher. Während aber die Random Seeks auf verschlüsseltem wie unverschlüsseltem Cloud-Speicher nahezu gleich sind, macht die Verschlüsselung beim lokalen Dateisystem mehrere hundert Seeks in der Minute aus.

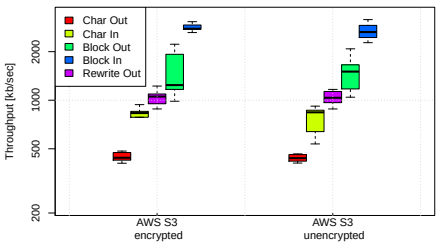


Fig. 3. Sequentieller Durchsatz auf AWS S3 via iSCSI

Abbildung 3 verdeutlicht nochmals die Insignifikanz des Unterschieds zwischen verschlüsseltem und unverschlüsseltem Cloud-Speicher. Der Durchsatz in

kB pro Sekunde ist beim sequentiellen Zugriff auf verschlüsselten wie unverschlüsseltem Speicher nicht wesentlich zu unterscheiden. Deutlich sichtbar ist auch der Performanzunterschied zwischen block-basierten Zugriffen und zeichenbasierten Zugriffen: Blockzugriffe sind deutlich performanter da hier das Mapping von Block-Offsets zu Buckets in weniger Zugriffen auf den entfernten Cloud-Speicher resultiert.

3 Zusammenfassung

Sicherheit ist gerade bei Cloud-Diensten wie Cloud-Speicher immens wichtig. Wie diese Arbeit gezeigt hat, ist Sicherheit jedoch nicht mit erheblichen Mehrkosten verbunden. Während die Performanzkosten der Verschlüsselung im lokalen Dateisystem mit der Dateigröße skalieren, verhält es sich bei der Cloud anders. Die Entfernung zur Cloud ist mit höherer Latenz und geringerem Durchsatz verbunden, wie auch schon in der Literatur gezeigt[6]. Die Distanz zum Speicher stellt daher einen Flaschenhals dar, welcher Performanzunterschiede bei der Verschlüsselung nahezu nivelliert, und das jeweils umso mehr, je komplexer die Gesamtarchitektur ist. Techniken wie Caching oder Prefetching wirken sich hierbei zusätzlich günstig auf die Verschlüsselung entfernt gespeicherter Daten aus wie unsere Ergebnisse mit unserer iSCSI-Implementierung "jSCSI" gezeigt haben. Nächste Schritte umfassen dabei das Versionieren der Daten ebenso wie entsprechende Integritätsprüfungen um eine weitergehende Sicherheit zu gewährleisten.

References

[1] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, "Internet small computer systems interface (iSCSI)," http://www.ietf.org/rfc/rfc3720.txt, 2004. 1, 2.2

[2] S. L. Garfinkel, "An evaluation of amazon's grid computing services: EC2, S3, and SQS," Harvard University, Tech. Rep., 2007. 2

[3] M. Kramis, V. Wildi, B. Lemke, S. Graf, H. Janetzko, and M. Waldvogel, "jSCSI – a Java iSCSI initiator," in *Paper, presented at: Jazoon 2007 – The International Conference on Java Technology*, 2007. 2

[4] S. Graf, P. Brend'amour, and M. Waldvogel, "jSCSI 2.0, Multithreaded Low-Level Distributed Block Access," University of Konstanz, Tech. Rep., 2010. 2

[5] "jClouds," http://www.jclouds.org/. 2

[6] Y. Chen and R. Sion, "To cloud or not to cloud? musings on costs and viability," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, 2011. 3

List of Figures