Fakultät Druck und Medien
Studiengang Medieninformatik

Bachelor Thesis
zur Erlangung des akademischen Grades Bachelor of Science

Evaluierung von Systemen zur Speicherung und Bereitstellung von Binärdaten im Kontext von Web Services

Gamze Isik

19. Juni 2023

Matrikelnummer: 39307
Bearbeitungszeitraum: 20. März - **19. Juni 2023**

Betreuer

Erstbetreuer: Prof. Martin Goik
Hochschule der Medien

Zweitbetreuer: Thomas Maier
Leomedia GmbH

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich (mit Ausnahme dieser Erklärung) als solches kenntlich gemacht.

Ort, Datum	Unterschrift

Zusammenfassung

Das Ziel der vorliegenden Arbeit ist die Evaluierung eines geeigneten Systems zur Speicherung und Bereitstellung von Binärdaten im Kontext von Web Services. Dabei werden die Anforderungen wie Performance, Verfügbarkeit, Sicherheit und API Anbindung gestellt. Durch die Realisierung eines Prototyps anhand der ausgewählten Speicherlösung werden die Binärdaten durch sichere, zeitlich begrenzte URL's bereitgestellt. Folgende Fragen werden gestellt: Welches Speichersystem ist im Hinblick auf Kosten, Performance und Verfügbarkeit für die Persistenz von Binärdaten besonders geeignet? Wie kann man Daten durch sichere, zeitlich begrenzte URL's bereitstellen? Verschiedene Speichersysteme werden verglichen und anhand von Kostenkalkulationen bewertet. Durch die Auswertung des Vergleichs wird der Prototyp implementiert und Messungen auf Testdaten durchgeführt, welches die wissenschaftliche Arbeit stützt.*Das Ergebnis kann dazu genutzt werden, auf neue Speicherlösungen mit höherer Performance und Sicherheit mit akzeptablen Kosten umzusteigen.

Abstract

The aim of this thesis is to evaluate a suitable system for storing and providing binary data in the context of web services. Requirements such as performance, availability, security, and API integration are set. By implementing a prototype based on the selected storage solution, binary data is provided through secure, time-limited URLs. The following questions are addressed: Which storage system is particularly suitable for persisting binary data in terms of cost, performance, and availability? How can data be provided through secure, time-limited URLs? Different storage systems are compared and evaluated based on cost calculations. By evaluating the comparison, the prototype is implemented and measurements are taken on test data, which supports the scientific work. The result can be used to switch to new storage solutions with higher performance and security at acceptable costs.

Inhaltsverzeichnis

Akronyme **4**

Abbildungsverzeichnis **5**

Tabellenverzeichnis **6**

1 Einleitung **7**
1.1 Problemstellung und Motivation 7
1.2 Zieldefinition und Vorgehensweise 7

2 Speichersysteme **9**
2.1 Arten von Speichersystemen 9
2.1.1 File Storage 10
2.1.2 Block Storage 11
2.1.3 Object Storage 12
2.2 Aktuelle Speichertechnologien im Markt 13
2.2.1 Eigenschaften 14
2.2.1.1 Sichere Speicherung 14
2.2.1.2 Hochverfügbarkeit 17
2.2.1.3 Performance 17
2.2.1.4 Kosten 17
2.2.1.5 API Anbindung 17
2.2.2 Bereitstellung der Dateien 17
2.3 Auswahl des Speichersystems 17
2.3.1 Kostenanalyse 17
2.3.2 Performance Analyse 17
2.3.3 Kalkulationsergebnisse 17

3 Prototypische Umsetzung **18**
3.1 Überblick und Vorgehensweise 18
3.2 Technologien 18
3.3 Speicherung von Binärdaten 18
3.4 Bereitstellung der Binärdaten 18
3.5 Messung der Performance auf Testdaten 18
3.5.1 Messungsergebnisse 18
3.6 Zusammenfassung der Implementierung 18

4 Ergebnisse dieser Arbeit **19**
4.1 Beschreibung und Funktionalität des Prototyps 19
4.2 Zusammenfassung der Ergebnisse 19
4.2.1 Kalkulationsergebnisse 19
4.2.2 Messungsergebnisse 19

5 Diskussion **20**
5.1 Analyse und Interpretation der Ergebnisse 20
5.2 Bewertung des Prototyps 20

6 Fazit **21**
6.1 Zusammenfassung der Ergebnisse 21
6.2 Beantwortung der Forschungsfrage 21
6.3 Potenzielle Anwendung des Prototyps 21

7 Danksagung **22**

8 Literaturverzeichnis **23**

9 Anhang **25**
9.1 Prototyp 25
9.1.1 Github Link 25
9.1.2 Dokumentation 25
9.1.3 Code Snippets 25

Akronyme

Abb.	Abbildung
Anm.	Anmerkung
AWS	Amazon Web Services
dt.	deutsch
engl.	englisch
GCP	Google Cloud Platform
HTML	Hypertext Markup Language
URL	Uniform Resource Locator
USA	United States of America, dt. Vereinigte Staaten von Amerika

Abbildungsverzeichnis

2.1.1 File Storage: Aufbau des Hierarchiesystems, RedHat redHat-storage 10
2.2.1 Einstellungen für Object Ownership, https://docs.aws.amazon.com/de_de/AmazonS3/latest/userguide//access-control-overview.html aws-iam-s3 16

Tabellenverzeichnis

1 Einleitung

Das folgende Kapitel dient der Einführung in die Problemstellung, Motivation sowie Ziele und Vorgehensweisen der vorliegenden Arbeit.

1.1 Problemstellung und Motivation

Die steigende Menge an Binärdaten im Kontext von Web Services, die in verschiedenen Anwendungen generiert werden, stellt eine große Herausforderung dar. Dabei ist es von großer Bedeutung, dass diese Daten sicher, zuverlässig und schnell gespeichert und abgerufen werden können. Vor diesem Hintergrund stellen sich Fragen nach der Auswahl eines geeigneten Speichersystems, das die Anforderungen wie Performance, Verfügbarkeit, Sicherheit und API-Anbindung erfüllt. Zudem müssen Mechanismen bereitgestellt werden, um den Zugriff auf die Daten zu beschränken durch sichere, zeitlich begrenzte URL's.

Diese Bachelorarbeit richtet sich auf das Problem einer Full-Service-Ticketing Software „leoticket", die vom Unternehmen Leomedia GmbH entwickelt wurde. Leomedia GmbH ist ein Unternehmen, das Software für Medienunternehmen wie Zeitungsverlage, Radiosender, Veranstalter, Künstler und Kulturvereine entwickelt. leomedia-web Leomedia (Bachelor- und Master-Themen [o.D.], [6])
Leoticket ist eines der vielen Produkte von Leomedia, dass Services wie Online-Kartenvorverkäufe, Abendkassen, den Einlass bei der Veranstaltung, Statistiken und Abrechnungen und die Planung der Veranstaltung realisiert.leomedia-web Leomedia (ebd.)

Das Problem liegt bei der Speicherung und Bereitstellung der Daten. Da es sich um Replikationen der Daten handelt, ist die Belastung des Systems hoch. Hohe Daten werden herumgeschoben. Die Bandbreite ist bei der Übertragung begrenzt. Ein weiteres Problem ist die Bereitstellung der Daten über Email Anhänge. Anhänge dürfen eine bestimmte Speichergröße nicht überschreiten. Wenn Ticketkäufer zehn oder 100 digitale Tickets kaufen, dann müssen diese über Email Anhänge bereitgestellt werden.

1.2 Zieldefinition und Vorgehensweise

Ziel dieser Arbeit ist die Realisierung eines Prototypen anhand der ausgewählten Speicherlösung die Binärdaten durch sichere, zeitlich begrenzte URLs bereitstellt. Dabei werden folgenden Fragen gestellt:

- Welches Speichersystem ist im Hinblick auf Kosten, Performance und Verfügbarkeit für die Persistenz von Binärdaten besonders geeignet?
- Wie kann man Daten durch sichere, zeitlich begrenzte URL's bereitstellen?

Im Rahmen der vorliegenden Bachelorarbeit werden verschiedene Arten von Speichersystemen untersucht, um die Forschungsfragen zu beantworten. Dabei erfolgt eine Analyse der aktuell verfügbaren

Speichertechnologien auf dem Markt hinsichtlich ihrer Eigenschaften wie Sicherheit, Verfügbarkeit, Performance und Kosten. Bei der Berücksichtigung der Integration des Speichersystems in Software-Produkte wird auch die API-Anbindung des Speichersystems betrachtet. Zur sicheren Bereitstellung von Dateien werden zudem geeignete Cloud-Provider miteinander verglichen. Kosten- und Performance-Kalkulationen werden durchgeführt, um eine geeignete Speicherlösung auszuwählen. Die Ergebnisse werden anschließend ausgewertet.

Im Zuge der prototypischen Umsetzung werden die ausgewählten Technologien implementiert und Testdateien zur Verfügung gestellt. Nach der Durchführung von Messungen zur Performance auf Testdaten erfolgt eine Zusammenfassung der Implementierung.

Abschließend werden die Ergebnisse nochmals dargestellt und interpretiert sowie Schwächen und Grenzen des Prototyps aufgezeigt. Zur Einhaltung des roten Fadens der Arbeit werden die gestellten Forschungsfragen beantwortet und potenzielle Anwendungen des Prototyps aufgelistet.

2 Speichersysteme

Speichersysteme sind eine entscheidende Komponente der IT Infrastruktur eines Unternehmens. In der heutigen Zeit kann man von Speichersystemen kaum absehen, da Big Data immer an Wichtigkeit gewinnt. Sie bieten eine Möglichkeit, große Mengen an Daten zu speichern und zu verwalten, um den Zugriff und die Nutzung zu erleichtern. Es gibt eine Vielzahl an Speichersystemen, die für verschiedene Zwecke konzipiert sind. Durch die große Auswahl in der IT und die stetig anwachsende Innovation stellt sich die Frage, welche Speichersysteme sich für bestimmte Zwecke (Use Cases) eignen. Die Wahl des richtigen Speichersystems hängt von den Anforderungen des Unternehmens ab, wie zum Beispiel der Art der zu speichernden Daten, dem benötigten Zugriff und die Skalierbarkeit. Eine gründliche Analyse der Anforderungen und Kosten ist entscheidend, um die beste Lösung zu finden, die den Bedürfnissen des Unternehmens entspricht.

Im nachfolgenden Kapitel werden die unterschiedlichen Typen von Speichersystemen vorgestellt, wobei der Fokus auf den drei Speicherarten File-, Object- und Block Storage liegt. Im Anschluss daran erfolgt ein Vergleich von zwei Cloud-Providern in Bezug auf sicherer Speicherung, Hochverfügbarkeit, Performance, Kosten, API-Anbindung sowie der Dateibereitstellung. Auf Basis dieser Kriterien wird eine Entscheidung darüber getroffen, welcher Provider den Bedürfnissen von "Leoticketëntspricht. Hierbei fließen die Kosten- und Performance-Analysen mit in die Entscheidung ein.

2.1 Arten von Speichersystemen

Im folgenden Abschnitt werden die verschiedenen Arten von Speichersystemen vorgestellt, die für die Speicherung von digitalen Daten verwendet werden. Hierbei werden die drei gängigsten Speicherarten File-, Object-, und Blob Storage behandelt.

Die heutige IT-Landschaft bietet eine Vielzahl von Speichersystemen, die je nach Bedarf und Anforderungen ausgewählt werden können. Neben den traditionellen Speichermedien wie Festplatten und Bandlaufwerken gibt es heute auch verschiedene Arten von Speichersystemen, die in der Cloud oder als lokale Lösungen bereitgestellt werden können. Dazu gehören unter anderem File Storage, Object Storage und Blob Storage.

Jeder dieser Speicherarten hat ihre spezifischen Vor- und Nachteile und ist für bestimmte Anwendungsfälle besser geeignet als andere.

2.1.1 File Storage

File Storage, auch dateiebenen- oder dateibasierter Storage genannt RedHat: File storage, block storage, or object storage? (2018), [7], ist eine Speicherlösung bei der Dateien auf einem Dateisystem gespeichert werden.

> Dieses System wird auch als hierarchischer Storage bezeichnet und gilt als das älteste und am weitesten verbreitete Datenspeichersystem für Direct und Network-Attached Storage. Dateisysteme organisieren Daten in hierarchischen Ordnern und Unterverzeichnissen, ähnlich wie in einem Dateiordner auf einem Computer. Dateien werden in der Regel auf einem Server oder einer Festplatte gespeichert und können von mehreren Benutzern gleichzeitig gelesen und geschrieben werden. Hierbei werden die Informationen in einzelnen Verzeichnisse abgelegt und können über den entsprechenden Pfad aufgerufen werden. Um dies zu ermöglichen, werden begrenzte Mengen an Metadaten genutzt, die dem System den genauen Standort der Dateien mitteilen, vgl. RedHat.

In der folgenden Abbildung wird die hierarchische Struktur des Dateispeichers visualisiert.

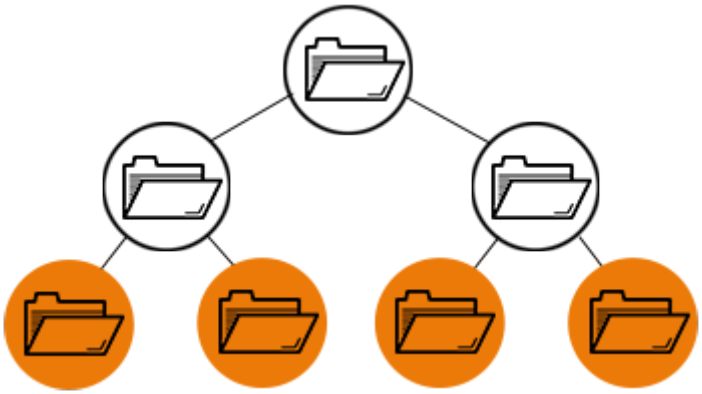

Abbildung 2.1.1: File Storage: Aufbau des Hierarchiesystems, RedHat

File Storage wird häufig in Unternehmen und Organisationen eingesetzt, um gemeinsame Dateiserver bereitzustellen oder Daten in Cloud-Speicherdiensten wie Dropbox oder Google Drive zu speichern. Auch wenn es von Betriebssystemen und Anwendungen gut unterstützt wird, kann die Performance und Skalierbarkeit von File Storage bei sehr großen Dateisystemen beeinträchtigt werden, was insbesondere bei stark frequentierten Anwendungen oder bei der Verarbeitung großer Datenmengen zum Problem werden kann.

> Mit zunehmenden Datenvolumen erfordert das Skalieren von Dateispeichern das Hinzufügen neuer Hardwaregeräte oder den Austausch vorhandener Geräte durch solche mit höherer Kapazität. Dies kann im Laufe der Zeit teuer werden. „As data volumes expand, scaling file storage requires [...]“, (Wahlman: Data Storage: Exploring File, Block, and Object Storage (2022), [8], Übersetzung des Autors)

Laut Wahlmann (2022, Übersetzung des Autors) wird die Datenspeicherung bei zu vielen Daten nicht nur teuer, sondern auch unhandlich und zeitaufwändig. Der schnelle und einfache Zugriff auf jede einzelne Datei wird schwierig, wenn man zig Millionen von Dateien in Tausenden von Verzeichnissen auf Hunderten von Speichergrößen speichert.

2.1.2 Block Storage

Block Storage, auch genannt als Block-level Storage speichert Dateien auf SAN(Storage Area Networks) basierten Netzwerken oder auf Cloud-basierten Speicherumgebungen. Das System teilt Daten in Blöcke auf und speichert die separaten Teile jeweils mit einer eindeutigen Kennung, vgl. IBM: What is block storage? (o.D.), [3].

Daten werden in Blöcken auf dem Datenträger gespeichert, die unabhängig voneinander adressiert werden können. Jeder Block ist eine feste Größe, typischerweise im Bereich von einigen Kilobytes bis hin zu einigen Megabytes. Diese Blöcke können im System an jeder Stelle gespeichert werden.

> Wenn auf Block Storage gespeicherte Daten abgerufen werden, verwendet das Server-Betriebssystem die eindeutige Adresse, um die Blöcke wieder zusammenzufügen und so die Datei zu erstellen. Der Vorteil besteht darin, dass das System nicht durch Verzeichnisse und Dateihierarchien navigieren muss, um auf die Datenblöcke zuzugreifen. Dadurch werden Effizienzen erzielt, da der Abruf von Daten schneller erfolgen kann, vgl. IBM: What is object storage? (o.D.), [4].

Typische Anwendungsbereiche des Block Storage sind Datenbanken, Virtualisierungsumgebungen und Anwendungen für Big Data-Analysen. Speicherung von strukturierten Daten wie Datenbanken, Virtuelle Maschinen und Betriebssysteme eignen sich besonders bei der Verwendung von Block Storage. Diese Art von Daten erfordert schnellen und direkten Zugriff auf bestimmte Bereiche des Speichers und muss häufig in Echtzeit ausgeführt werden. Block Storage eignet sich daher am besten für Anwendungen mit hohen Anforderungen an die Leistung und niedriger Latenzzeit.

2.1.3 Object Storage

Object Storage hat sich als Speichertechnologie in den letzten Jahren immer stärker etabliert und wird von vielen Unternehmen als Alternative zu traditionellen Speicherlösungen wie Block- oder File Storage angesehen. Die ersten Object Storage Systeme wurden bereits in den 1990er Jahren entwickelt, aber erst mit dem Aufkommen von Big Data, IoT und der Cloud-Nutzung hat es einen breiteren Einsatz gefunden. Heute bieten viele Cloud Provider wie Amazon Web Services (AWS) und Google Cloud Platform (GCP) Object Storage als einen ihrer Haupt-Cloud-Services an.

> Object Storage ist für den Umgang mit großen Datenvolumen und unstrukturierten Daten entwickelt wurden. Sie speichert Daten als eigenständige Objekte, die aus Daten und Metadaten bestehen und einen eindeutigen Identifier (UID) haben („Object storage (aka object-based storage) is a type of data storage used to [...]“, dataCore-OS Ivanov: File Storage vs. Object Storage: Understanding Differences, Applications and Benefits, What is Object Storage? (2020), [5], Übersetzung des Autors).

Im Gegensatz zu hierarchischen Systemen wie beim File Storage ist das Speichersystem flach strukturiert. Durch die einfache API Anbindung kann es mit vorhandenen Anwendungen integriert werden. Nutzer können detaillierte Informationen wie beispielsweise Erstellerangaben, Schlüsselwörter sowie Sicherheit-und Datenschutzrichtlinien hinterlegen. Diese Daten bezeichnet man als Metadaten.

Laut nx-fileScala Wahlman, 2022 ist Skalierbarkeit der Hauptvorteil, da bei der Speicherung von Petabyte und Exabyte alle Objekte in einem Namespace abgelegt werden. Selbst wenn dieser Namespace auf Hunderte von Hardwaregeräten und Standorten verteilt ist, können alle Objekte schnell abgerufen werden. Andere Vorteile von Objekt Storage beinhalten die Datenintegritätsprüfung, im Englischen bekannt als „erasure coding“und die Datenanalyse.

Auch Object Storage hat seine Nachteile. Laut redHat-storage RedHat muss das Objekt nach der Speicherung bei Veränderung komplett neu überschrieben werden. Sie sind für traditionelle Datenbanken nicht geeignet, da das Schreiben von Objekten Zeit beansprucht und man sich mit der API auseinandersetzen muss, vgl. redHat-storage RedHat.

Insgesamt bietet Object Storage eine skalierbare und flexible Methode zur Speicherung von unstrukturierten Daten. Organisationen sollten jedoch die Vor- und Nachteile von Object Storage im Kontext ihrer spezifischen Anwendungsfälle abwägen, um eine fundierte Entscheidung über die beste Speichermethode zu treffen.

Da leoticket Daten wie Rechnungen und Tickets als Dateien speichern und abrufen abruft, ist Object Storage die richtige Speicherart.
Für leoticket ist die Object Storage Variante die am Besten geeignetste Speicherlösung, da ...

2.2 Aktuelle Speichertechnologien im Markt

Die beiden größten Cloud-Speicheranbieter, Amazon Web Services (AWS) und Google Cloud Platform (GCP), bieten eine Vielzahl von Speicherlösungen an, die auf die Bedürfnisse von Unternehmen zugeschnitten sind.

In diesem Kapitel werden die aktuellen Speichertechnologien auf dem Markt untersucht, wobei der Fokus auf den Angeboten von AWS und GCP liegt. Um eine Vergleichsgrundlage zwischen AWS und GCP zu schaffen, werden die verschiedenen Aspekte wie sichere Speicherung, Hochverfügbarkeit, Leistung, Kosten, API Anbindung und die Bereitstellung der Dateien betrachtet. Dieses Vorgehen dient dazu, zu ermitteln, welche Speicherart sich am besten für welche Anforderungen eignet. Als Kontrast dazu wird das Open-Source-Objektspeichersystem MinIO betrachtet.

AWS bietet eine Reihe von Speicheroptionen an, darunter Amazon S3 (Simple Storage Service). Amazon S3 ist ein Object Storage-Service, der für die Speicherung und den schnellen Abruf von unstrukturierten Daten wie Videos, Fotos und Dokumenten ausgelegt ist.
Google Cloud Platform bietet ebenfalls eine Vielzahl von Speicherlösungen an, darunter Google Cloud Storage. Google Cloud Storage ist ein Object Storage-Service, der für die Speicherung von unstrukturierten Daten wie Bildern, Videos und Dokumenten ausgelegt ist.

Insgesamt bieten AWS und Google Cloud Platform eine Vielzahl von Speicherlösungen an, die auf die Bedürfnisse von Unternehmen zugeschnitten sind. Organisationen sollten jedoch die Vor- und Nachteile jeder Speicherlösung abwägen, um die beste Lösung für ihre spezifischen Anforderungen zu finden.

2.2.1 Eigenschaften

Im vorliegenden Abschnitt werden Amazon S3 und Google Cloud Storage in Bezug auf verschiedene Kriterien untersucht. Dabei werden zunächst die Eigenschaften von AWS S3 erläutert, gefolgt von einer Betrachtung von GC Storage. Ziel ist es, die Unterschiede zwischen den beiden Anbietern hervorzuheben und eine Entscheidungshilfe zu bieten, welche Plattform für die Anforderungen von „leoticket“am besten geeignet ist. Die Auswahl der Kriterien erfolgt in Anlehnung an die spezifischen Anforderungen von „leoticket“.

2.2.1.1 Sichere Speicherung

Amazon S3

Viele Anbieter von Object Storage-Lösungen bieten integrierte Verschlüsselungsmöglichkeiten an, um sicherzustellen, dass Daten sowohl in Ruhe als auch in Bewegung geschützt sind. Dabei können unterschiedliche Verschlüsselungsmethoden zum Einsatz kommen. In Bezug auf die sichere Speicherung bieten sowohl AWS als auch GCP verschiedene Optionen für die Verschlüsselung von Daten.Die Sicherheit der gespeicherten Daten ist von entscheidender Bedeutung, um die Integrität und Vertraulichkeit der Daten zu gewährleisten. In diesem Unterabschnitt werden die verschiedenen Sicherheitsfunktionen von Amazon S3 untersucht.

IAM (Identity and Access Management) ist ein wichtiger Bestandteil von AWS und ermöglicht es Benutzern, Gruppen und Rollen zu erstellen, um den Zugriff auf S3 zu verwalten. Benutzer können individuelle Berechtigungen zugewiesen werden, während Gruppen und Rollen mehrere Benutzer mit denselben Berechtigungen zusammenfassen können. Auf S3-Buckets und Objekte kann man eine granuläre Zugriffssteuerung anwenden. Benutzer, Gruppen oder Rollen können so berechtigt werden, den Zugriff auf bestimmte Buckets und Objekte zu beschränken oder zu erlauben. ”Beim Erteilen von Berechtigungen in Amazon S3 entscheiden Sie [...].”[aws-iam-s3] AWS: Security: Identity and Access Management (2023), [1]

Eine weitere wichtige Sicherheitsfunktion von Amazon S3 ist die Datenverschlüsselung. S3 bietet eine Vielzahl von Verschlüsselungsoptionen für die serverseitige und clientseitige Verschlüsselung. Da für leoticket die clientseitige Verschlüsselung nicht in Frage kommt, liegt der Fokus auf der serverseitigen Verschlüsselung. Es gibt drei Methoden, darunter die serverseitige Verschlüsselung mit Amazon S3-verwalteten Schlüsseln (SSE-S3), mit KMS-verwalteten Schlüsseln (SSE-KMS) und die SSE-C. Diese Optionen ermöglichen es Benutzern, die Verschlüsselung auf ihre spezifischen Anforderungen abzustimmen und so die Sicherheit der Daten zu gewährleisten.
Laut [aws-iam-s3] AWS nutzen Buckets standardmäßig die SSE-S3 Methode. Für die Verschlüsselung wird die 256-bit Advanced Encryption Standard (AES-256) verwendet. Seit dem 5. Januar 2023 sind alle neu erstellen Buckets auf SSE-S3 ausgelegt. Alle neuen Objekte sind beim Hochladen automatisch verschlüsselt, ohne weitere Zusatzkosten und keine Einbußung der Leistung.

> Die Serverseitige Verschlüsselung schützt die Daten at rest. Amazon S3 verschlüsselt jedes Objekt mit einem eindeutigen Schlüssel. Als zusätzliche Sicherheitsmaßnahme werden diese eindeutigen Schlüssel mit einem weiteren Schlüssel verschlüsselt, welches in regelmäßigen Abständen rotiert wird, vgl. [aws-iam-s3] ebd.

Amazon S3 stellt auch die SSE-KMS als Auswahl zur Verfügung. AWS-KMS ist ein Dienst, dass einen Schlüsselverwaltungssystem zur Verfügung stellt. Es verschlüsselt die Objekt Daten und speichert die S3 Checksum, das im Objekt Metadaten steckt, in verschlüsselter Form. Man kann die SSE-KMS in der AWS Management Konsole oder über die AWS KMS API verwalten. Jedoch gibt es zusätzliche Kosten bei der Verwendung der Methode. Dazu mehr im Kosten Abschnitt. Bei der Nutzung von AWS-SSE gibt es zwei Methoden. Einmal die AWS managed key oder die customer managed key. Es unterstützt die „envelope encryption“. Das bedeutet, das die Schlüssel für die Daten durch einen Master Key verschlüsselt wird. Dies erleichtert die Verwaltung der Schlüssel.

Bei der AWS managed key Variante wird automatisch ein Schlüssel erstellt, sobald ein Objekt in ein Bucket hochgeladen wird. Dieser generierte Schlüssel wird dann für die Ver- und Entschlüsselung der Daten verwendet. Wenn man einen eigenen Schlüssel bei KMS verwenden möchte, dann erstellt man zuerst einen symmetrischen Key vor der KMS Konfiguration. Bei der Bucket Erstellung kann man anschließend den selbst-erstellten Key angeben. Die Nutzung von Customer Managed Keys hat einige Vorteile, die den Anforderungen von leoticket entspricht. Selbsterstellte Schlüssel bieten mehr Flexibilität und Kontrolle. Man kann sie selber erstellen, rotieren und deaktivieren. Hinzufügend kann man auch Zugriffskontrollen und Auditierung für den Schutz der Daten konfigurieren.

Wenn man die SSE-KMS Variante aussucht, kann man auch die S3 Bucket Key Funktion aktivieren. Dies kann die Request Kosten bis zu 99 Prozent reduzieren, indem die Request Traffic von Amazon S3 zu AWS KMS reduziert wird. Durch die Aktivierung der S3 Bucket Key für einen Bucket werden unique data keys für die Objekte im Bucket erstellt. Diese Bucket Keys werden für eine bestimmte Zeit verwendet, welches Abrufe zu Amazon S3 nach AWS KMS reduziert um Verschlüsselungsoperationen durchzuführen.

Zuletzt gibt es noch die SSE-C (server-side encryption with customer-provided keys). Bei der SSE-C stellt der Customer seinen eigenen Schlüssel zur Verfügung. Dieser Schlüssel wird nicht von Amazon S3 gespeichert, im Gegensatz bei AWS KMS. Amazon S3 übernimmt mit dem bereitgestellten Schlüssel die Datenverschlüsselung beim Schreiben sowie die Datenentschlüsselung beim Zugriff auf Objekte. Amazon S3 entfernt anschließend den Schlüssel aus dem Speicher. Da Amazon S3 den Schlüssel nicht speichert, sepeichert er den zufällig generierten HMAC (Hash-baded Message Authentication Code) Wert vom Encryption Key um zukünftige Requests zu validieren.”Note: Amazon S3 does not store the encryption key that you provide. [...]”, aws-sse-c. AWS: Using server-side encryption with customer-provided keys (SSE-C) (2023), [2].

Object Ownership ist eine weitere wichtige Sicherheitsfunktion von Amazon S3. Mit Object Ownership können Benutzer oder Gruppen die Eigentümerschaft von Objekten in S3-Buckets besitzen. Dies bedeutet, dass nur autorisierte Benutzer die Berechtigung haben, Objekte zu löschen oder zu ändern, was die Sicherheit der Daten erhöht.„S3 Object Ownership ist eine Einstellung auf Amazon-S3-Bucket-Ebene, mit der Sie Zugriffskontrolllisten (ACLs) deaktivieren und das Eigentum an jedem Objekt in Ihrem Bucket übernehmen können, [...].”aws-iam-s3 AWS: Security: Identity and Access Management (2023), [1]

AWS empfiehlt die ACL (Access Control List) auf Bucket-Ebenen deaktiviert zu lassen. Alle Objekte eines Buckets gehört so dem Bucket Owner. Laut aws-iam-s3 AWS verfügt Object Ownership über drei Einstellungen, mit denen man die Eigentümerschaft von Objekten steuern kann.

Einstellung	Gilt für	Auswirkung auf Object Ownership	Auswirkungen auf ACLs	Hochladen akzeptiert
Bucket-Eigentümer erzwungen (empfohlen)	Alle neuen und bestehenden Objekte	Bucket-Eigentümer besitzt jedes Objekt.	ACLs sind deaktiviert und wirken sich nicht mehr auf die Zugriffsberechtigungen für Ihren Bucket aus. Anfragen zum Festlegen oder Aktualisieren von ACLs schlagen fehl. Anfragen zum Lesen von ACLs werden jedoch unterstützt. Bucket-Eigentümer hat das volle Eigentum und die volle Kontrolle. Der Objekt-Writer hat nicht mehr das volle Eigentum und die volle Kontrolle.	Uploads mit ACLs mit vollem Zugriff des Bucket-Eigentümers oder Uploads, die keine ACL angeben
Bucket-Eigentümer bevorzugt	Neue Objekte	Wenn ein Objekt-Upload die `bucket-owner-full-control` vordefinierte ACL beeinhaltet, gehört dem Bucket Eigentümer das Objekt. Objekte, die mit anderen ACLs hochgeladen wurden, gehören dem Schreibkonto.	ACLs können aktualisiert werden und können Berechtigungen erteilen. Wenn ein Objekt-Upload die `bucket-owner-full-control` vordefinierte ACL enthält, hat der Bucket-Eigentümer Vollzugriff und der Objekt-Writer hat keinen Vollzugriff mehr.	Alle Uploads
Objekt-Writer (Standard)	Neue Objekte	Der Objekt-Writer besitzt das Objekt.	ACLs können aktualisiert werden und können Berechtigungen erteilen. Der Objekt-Writer hat vollen Kontrollzugriff.	Alle Uploads

Abbildung 2.2.1: Einstellungen für Object Ownership, aws-iam-s3 https://docs.aws.amazon.com/de_de/AmazonS3/latest/userguide//access-control-overview.html

Laut aws-iam-s3 AWS zeigt die obige Tabelle die Auswirkungen, die jede Einstellung für Object Ownership auf ACLs, Objekte, Objekteigentümer und Objekt-Uploads hat.

Logging ist ein weiterer Aspekt der Amazon S3-Sicherheit. S3 bietet verschiedene Logging-Optionen, darunter Bucket Logging und Object-Level Logging, die es Benutzern ermöglichen, Zugriffe auf S3-Objekte aufzuzeichnen und zu überwachen. Diese Funktionen sind entscheidend, um Compliance-Anforderungen zu erfüllen und verdächtige Aktivitäten zu erkennen. AWS bietet eine Vielzahl von tools zur Überwachung der Amazon S3 Ressourcen. Darunter die Amazon CloudWatch Alarms, AWS CloudTrail Logs, Amazon S3 Access Logs und die AWS Trusted Advisor.

Google Cloud Storage

2.2.1.2 Hochverfügbarkeit

2.2.1.3 Performance

2.2.1.4 Kosten

2.2.1.5 API Anbindung

2.2.2 Bereitstellung der Dateien

2.3 Auswahl des Speichersystems

In diesem Abschnitt wird durch eine Kosten-, und Performance Analyse die Speichersysteme ausgewertet und eine Auswahl getroffen. Die Kalkulationsergebnisse werden zum Schluss dargestellt.

2.3.1 Kostenanalyse

2.3.2 Performance Analyse

2.3.3 Kalkulationsergebnisse

3 Prototypische Umsetzung

3.1 Überblick und Vorgehensweise

3.2 Technologien

3.3 Speicherung von Binärdaten

3.4 Bereitstellung der Binärdaten

3.5 Messung der Performance auf Testdaten

3.5.1 Messungsergebnisse

3.6 Zusammenfassung der Implementierung

4 Ergebnisse dieser Arbeit

4.1 Beschreibung und Funktionalität des Prototyps

4.2 Zusammenfassung der Ergebnisse

4.2.1 Kalkulationsergebnisse

4.2.2 Messungsergebnisse

5 Diskussion

5.1 Analyse und Interpretation der Ergebnisse

5.2 Bewertung des Prototyps

6 Fazit

6.1 Zusammenfassung der Ergebnisse

6.2 Beantwortung der Forschungsfrage

6.3 Potenzielle Anwendung des Prototyps

7 Danksagung

8 Literaturverzeichnis

Internetquellen

aws-iam-s3 [1] AWS: Security: Identity and Access Management. (2023). Abgerufen am 11.05.2023. URL: https://docs.aws.amazon.com/de_de/AmazonS3/latest/userguide//access-control-overview.html.

aws-sse-c [2] AWS: Using server-side encryption with customer-provided keys (SSE-C). (2023). Abgerufen am 12.05.2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/ServerSideEncryptionCustomerKeys.html.

ibm-topics [3] IBM: What is block storage? (o.D.) Abgerufen am 07.05.2023. URL: https://www.ibm.com/topics/block-storage.

ibm-storage [4] IBM: What is object storage? (o.D.) Abgerufen am 07.05.2023. URL: https://www.ibm.com/topics/object-storage.

dataCore-OS [5] Mike Ivanov: File Storage vs. Object Storage: Understanding Differences, Applications and Benefits, What is Object Storage? (2020). Abgerufen am 07.05.2023. URL: https://www.datacore.com/blog/file-object-storage-differences/.

leomedia-web [6] GmbH Leomedia: Bachelor- und Master-Themen. (o.D.) URL: https://www.leomedia.de/unternehmen/abschlussarbeiten/.

redHat-storage [7] RedHat: File storage, block storage, or object storage? (2018). Abgerufen am 06.05.2023. URL: https://www.redhat.com/en/topics/data-storage/file-block-object-storage.

nx-fileScala [8] Lauren Wahlman: Data Storage: Exploring File, Block, and Object Storage. (2022). Abgerufen am 06.05.2023. URL: https://www.nutanix.com/blog/block-storage-vs-object-storage-vs-file-storage.

9 Anhang

9.1 Prototyp

9.1.1 Github Link

9.1.2 Dokumentation

9.1.3 Code Snippets