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

Abb.	Abbildung
Anm.	Anmerkung
AWS	Amazon Web Services
dt.	deutsch
engl.	englisch
GC	Google Cloud
GCP	Google Cloud Platform
HTML	Hypertext Markup Language
IAM	Identity and Access Management
S3	Simple Storage Service
SSE	Server-Side Encryption
SSE-C	Server-Side Encryption with Customer-Provided Keys
SSE-KMS	Server-Side Encryption with AWS Key Management Service
SSE-S3	Server-Side Encryption with S3 Managed Keys
URL	Uniform Resource Locator
USA	United States of America, dt. Vereinigte Staaten von Amerika

Abbildungsverzeichnis

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 wird. Leomedia GmbH ist ein Unternehmen, das Software für Medienunternehmen wie Zeitungsverlage, Radiosender, Veranstalter, Künstler und Kulturvereine entwickelt. leomedia-web Leomedia (Bachelor- und Master-Themen, [10])
Leoticket ist eines der vielen Produkte von Leomedia, dass Services wie Online-Kartenvorverkäufe, Abendkassen, den Einlass bei der Veranstaltung, Statistiken, Abrechnungen und die Planung der Veranstaltung realisiert.leomedia-web Leomedia (ebd.)

Das vorliegende Problem von leoticket betrifft die Speicherung und Bereitstellung von Daten wie Tickets und Rechnungen. Durch die Nutzung von Galera Cluster handelt es sich um replizierte Daten. Durch die geringe Speichergröße von 200GB RAM kommt die Datenbank bereits an ihre Grenzen, was zu einer hohen Belastung des Systems führt. zu hohe Daten müssen verschoben und repliziert werden, wobei die Bandbreite bei der Übertragung begrenzt ist. Ein weiteres Problem ist die Bereitstellung der Daten über Email Anhänge. Anhänge dürfen eine bestimmte Speichergröße nicht überschreiten. Wenn Ticketkäufer mehrere Tickets in einer Bestellung tätigen, dann müssen diese über Email Anhänge bereitgestellt werden.

Leomedia plant eine Neugestaltung der leoticket-Anwendung, bei der sie sich von der Galera Cluster Technologie lösen möchten. In dieser Untersuchung werden keine relationalen Datenbanken berücksichtigt, da die Nachfrage nach neuen Speicherlösungen steigt. Es werden spezifische Anforderungen an die neue Speicherlösung gestellt.

1.2 Zieldefinition und Vorgehensweise

Ziel dieser Arbeit besteht darin, anhand der Anforderungen von leoticket eine geeignete Speicherlösung zu empfehlen und einen Prototypen basierend auf den ausgewählten Cloud-Providern zu erstellen. Der Prototyp soll die Bereitstellung von Binärdaten durch sichere und zeitlich begrenzte URLs ermöglichen. Dabei werden folgende Fragen untersucht:

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

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-Produkten 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 zu empfehlen. Die Ergebnisse werden anschließend ausgewertet.

Im Zuge der prototypischen Umsetzung werden die ausgewählten Technologien von den Cloud Providern 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 von Speichersystemen kaum abgesehen werden, da Big Data mehr 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 eignen. Die Wahl des richtigen Speichersystems hängt von den Anforderungen des Unternehmens ab, beispielsweise 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 folgenden 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 leoticket mehr entspricht. Hierbei fließen die Kosten- und Performance-Analysen mit in die Entscheidung ein.

2.1 Arten von Speichersystemen

In folgenden Unterabschnitten 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 Block 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 Block Storage.

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

2.1.1 File Storage

File Storage, auch dateiebenen- oder dateibasierter Storage genannt RedHat: File storage, block storage, or object storage? (2018), [12], 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 Verzeichnissen 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.

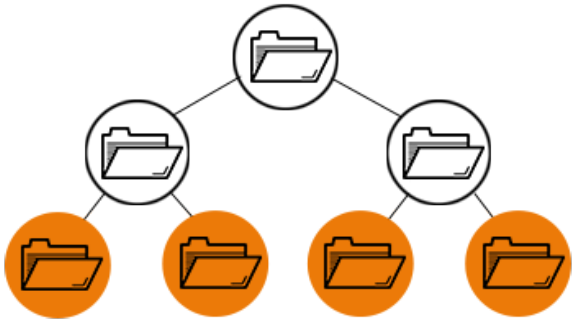

Abbildung 2.1.1: File Storage: Aufbau des Hierarchiesystems, RedHat

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), [19], 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 Datei wird schwierig, wenn viele Dateien in Tausenden von Verzeichnissen auf Hunderten von Speichergrößen gespeichert werden.

2.1.2 Block Storage

Block 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?, [7].

Daten werden in Blöcken auf dem Datenträger gespeichert, die unabhängig voneinander adressiert sind. Jeder Block hat 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?, [8].

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. 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), [9], 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.

Fazit

Aufgrund der Anforderungen von leoticket, Daten wie Rechnungen und Tickets zu speichern und abzurufen, erweist sich Object Storage als die geeignete Speicheroption. Da keine spezifischen Anforderungen hinsichtlich der Latenzzeit bestehen und auch keine Dateien im Petabyte- oder Exabyte-Bereich gespeichert werden müssen, scheidet die Block Storage-Variante aus. Viele Cloud-Anbieter stellen Methoden für Object Storage bereit, die Sicherheitsfunktionen, hohe Verfügbarkeit, gute Performance und eine API-Integration umfassen. Darüber hinaus ermöglichen Cloud Storage-Systeme die Bereitstellung von Dateien als Links. Die Skalierbarkeit ist ein weiterer entscheidender Faktor, der für die Wahl von Object Storage spricht.

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 der Ermittlung der Speicherart, die sich am besten auf die genannten Anforderungen fokusiert. Als Kontrast 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. GCP bietet ebenfalls eine Vielzahl von Speicherlösungen an, darunter Google Cloud Storage (GC Storage). Letztere ist ein Object Storage-Service, der für die Speicherung von unstrukturierten Daten wie Bildern, Videos und Dokumenten ausgelegt ist.

Insgesamt bieten AWS und GCP 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 GC Storage in Bezug auf verschiedene Kriterien untersucht. Die Auswahl der Kriterien erfolgt in Anlehnung an die spezifischen Anforderungen von leoticket. Dabei werden zunächst die Eigenschaften von Amazon S3 erläutert, gefolgt von einer Betrachtung von GC Storage. Ziel ist es, Funktionen und Angebote beider Cloud Provider zu untersuchen, die die Kriterien erfüllen und eine Entscheidungshilfe für leoticket zu bieten.

2.2.1.1 Sichere Speicherung

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.

Amazon S3

IAM (Identity and Access Management) ist ein wichtiger Bestandteil von AWS und ermöglicht Benutzer, Gruppen und Rollen zu erstellen, um den Zugriff auf S3 zu verwalten. Benutzern können individuelle Berechtigungen zugewiesen werden, während Gruppen und Rollen mehrere Benutzer mit denselben Berechtigungen zusammenfassen können. Auf S3-Buckets und Objekte kann eine granulare Zugriffssteuerung angewendet werden. 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, [2]

Eine weitere wichtige Sicherheitsfunktion von Amazon S3 ist die Datenverschlüsselung. S3 bietet eine Vielzahl von Verschlüsselungsoptionen für die serverseitige und clientseitige Verschlüsselung. Da die clientseitige Verschlüsselung für leoticket keine Anwendung findet, wird diese Methode nicht weiter in Betracht gezogen. Die clientseitige Verschlüsselung erfordert die Generierung und Kommunikation eines separaten Schlüssels für jeden Kunden, um auf die Dateien zugreifen zu können. Aus Gründen des Aufwands ist diese Vorgehensweise nicht praktikabel. Daher wird der Schwerpunkt auf die serverseitige Verschlüsselung gelegt. Es existieren drei Verschlüsselungsmethoden, nämlich die serverseitige Verschlüsselung mit Amazon S3-verwalteten Schlüsseln (SSE-S3), mit dem AWS Key-Management-Service-verwalteten Schlüsseln (SSE-KMS) und die vom Kunden verwalteten Schlüsseln (SSE-C). Durch diese Optionen haben Benutzer die Möglichkeit, die Verschlüsselung gemäß ihren individuellen Anforderungen anzupassen und somit die Datensicherheit 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 ohne weitere Zusatzkosten und keine Einbußung der Leistung automatisch verschlüsselt.

> 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), [3].

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, [2]

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

Auch Google Cloud bietet eine Vielzahl an Sicherheitsfunktionen, um sicherzustellen, dass Daten in der Cloud sicher gespeichert und geschützt sind. Unter anderem die Datenverschlüsselung. Google Cloud bietet genau wie Amazon S3 Serverseitige Verschlüsselung an. Hier unterscheidet man zwischen Google-managed Keys, Customer-managed encryption keys und die Customer-supplied encryption keys. Google-managed encryption ist die Standard Verschlüsselungsoption von Google Cloud. Cloud Storage verschlüsselt die Daten serverseitig, bevor die Daten auf die Festplatte geschrieben werden.

> Die Standard Variante verwaltet für den Nutzer die Encryption Keys in ihrem eigenen Key Management System. Auch Google verwendet für die Verschlüsselung die AES-256 wie AWS. Als Nutzer muss man bei dieser Varianet keine Einstellungen berücksichtigen. Daten werden automatisch verschlüsselt beim Abruf, vgl. `gcp-encrypt` Storage: Google-managed encryption keys, [17].

Wenn man mehr Kontrolle über die Schlüssel haben möchte, dann gibt es die Customer-managed encryption Option. Die Schlüssel werden von Cloud KMS (Cloud Key Management Service) erstellt und verwaltet. Diese Schlüssel werden vom Nutzer anschließend extern oder in einem HSM Cluster gespeichert. Customer Keys kann man für indivduelle Objekte benutzen oder einen Bucket so einstellen, dass er einen Standard Key für alle neuen Objekte verwendet. Der erstellte Schlüssel wird für die Verschlüsselung der Objektdaten, die CRC32C Checksum des Objekts und für die MD5 Hash verwendet. Als zusätzlichen Schutz gibt es die Service Agents, auch genannt als service accounts. Diesem Agent kann man bestimmte Rechte geben, sodass es Zugriff auf den gewünschten Encryption Key hat, um damit Objekte verschlüsseln zu können.

Zuletzt gibt es noch die Customer-supplied encryption keys. Als zusätzliche Sicherheitsebene zu Google-managed encryption keys können Nutzer ihren eigenen AES-256 Encryption Key bereitstellen, welches als Base64 encoded ist. Bei dieser Variante wird der Schlüssel nicht von Google Cloud gespeichert oder verwaltet. Die Nutzer müssen diese Schlüssel selber verwalten und speichern. Um zukünftige Requests zu validieren speichert Google Cloud einen kryptografischen Hash vom Schlüssel. Jedoch kann der Encryption Key nicht aus dem Hash regeneriert werden oder den Hash nicht zum Entschlüsseln anwenden.

Des weiteren gibt es auch in Google Cloud Zugriffskontrolleinstellungen, mit der Nutzer genau steuern können, wer auf ihre Daten zugreifen kann. Den Zugriff kann man auf Bucket- und Objektebene steuern und Benutzern und Gruppen bestimmte Rollen zuweisen. Dabei wird zwischen Uniform und Fine-grained unterschieden. Uniform Access Control (UAC) ermöglicht es, den Zugriff auf einen gesamten Bucket in Google Cloud Storage auf der Ebene von Rollen zuzuweisen. Dazu können verschiedene vordefinierte Rollen wie "Leser", SSchreiberöder "Besitzer"verwendet werden, die festlegen, welche Aktionen ein Benutzer auf dem Bucket ausführen darf. UAC ist eine einfachere Methode zur Zugriffskontrolle, da die Zugriffsrechte auf Bucket-Ebene vergeben werden. Das bedeutet, dass alle Objekte in dem Bucket automatisch dieselben Zugriffsrechte haben.

Fine-Grained Access Control (FGAC) hingegen ermöglicht es, die Zugriffskontrolle auf die Ebene von Objekten oder sogar auf Teile von Objekten herunterzubrechen. Das bedeutet, dass jeder

Benutzer oder jede Gruppe individuelle Zugriffsrechte auf bestimmte Objekte oder Teile von Objekten haben kann. Mit FGAC kann man sehr granulare Zugriffssteuerungen implementieren. Es ist eine mächtige Methode, aber auch komplexer und zeitaufwändiger zu implementieren als UAC. Mit Cloud IAM können auch rollenbasierte Zugriffssteuerungen für Bucket- unObjektebene ähnlich wie bei AWS umgesetzt werden.

Weitere Funktionen die Google Cloud anbietet ist Objekt Versioning, Audit Logging, Bucket Locks, VPC Service Controls, Data Loss Prevention und Identity-Aware Proxy.

2.2.1.2 Hochverfügbarkeit

Amazon S3

Amazon S3 ist ein skalierbarer und hochverfügbarer Objekt Storage, der eine Verfügbarkeit von 99.99 Prozent garantiert.„Designed to provide 99.999999999 percent durability and 99.99 percent availability of objects over a given year.“, AWS: Data Protection in Amazon S3, [1]

aws-availability

Dies wird durch die Verwendung von Multi-Availability Zone Architekturen erreicht, die eine automatische Replikation von Daten in verschiedenen physischen Standorten ermöglichen. Die Multi-Availability Zone Architektur von Amazon S3 basiert auf der Aufteilung von Daten in mehrere geografisch getrennte Verfügbarkeitszonen (AZs). Jede AZ besteht aus mehreren physischen Rechenzentren, die sich in einem geografisch getrennten Gebiet befinden. Jede AZ ist vollständig unabhängig und bietet eine hohe Redundanz und Verfügbarkeit. Wenn ein Benutzer ein Objekt in Amazon S3 hochlädt, wird es automatisch in mehrere AZs repliziert.um sicherzustellen, dass das Objekt auch bei AUsfällen in einer AZ weiterhin verfügbar ist. Im Falle eines Ausfalls einer AZ wird Amazon S3 automatisch die Anfragen auf eine andere AZ umleiten, um eine ununterbrochene Verfügbarkeit des Objekts sicherzustellen. Durch die Cross-Region Replikation werden Daten automatisch in andere AWS-Regionen repliziert. Dadurch kann eine hohe Verfügbarkeit der Daten im Falle eines Ausfalls einer gesamten AWS-Region gewährleistet werden.

Amazon S3 bietet auch eine hohe Verfügbarkeit von Metadaten, die für den Zugriff auf Objekte verwendet werden. Metadaten werden automatisch in mehrere AZs repliziert, um sicherzustellen, dass die Metadaten auch im Falle eines Ausfalls einer AZ verfügbar bleiben. Zusätzlich zu Multi-Availability Zone Architekturen verwendet Amazon S3 auch Fehlerkorrektur- und Erkennungsmechanismen wie CRC-Prüfungen, um die Integrität von Daten sicherzustellen und sicherzustellen, dass die gespeicherten Daten stets korrekt sind.

Durch Aktivierung der Versionierung wird jeder Objektversion, die in einem Amazon S-Bucket gespeichert ist, eine eindeutige Versions-ID zugewiesen. Wenn eine Objektversion versehentlich gelöscht oder überschrieben wird, kann die vorherige Version wiederhergestellt werden.

Um eine hohe Verfügbarkeit bereitzustellen, stell Amazon S3 außerdem noch die S3-Transfer Acceleration zur Verfügung. Durch die Verwendung von Amazon S3-Transfer Acceleration können Benutzer die Übertragung großer Datenmengen beschleunigen, indem ein optimierter Netzwerkpfad genutzt wird. Dadurch kann de Verfügbarkeit der Daten verbessert werden, indem Verbindungsprobleme minimiert werden.

Um die Leistung von Amazon S3 zu überwachen und auf mögliche Probleme reagieren zu können wird CloudWatch bereitgestellt. Dadurch können Ausfallzeiten minimiert werden und herausfinden, an welchen Momenten die Verfügbarkeit am geringsten ist.
Insgesamt bietet Amazon S3 eine hochverfügbare und zuverlässige Speicherlösung, die durch Multi-Availability Zone Architekturen und Fehlerkorrekturmechanismen eine Verfügbarkeit von 99,99 Prozent gewährleistet.

Google Cloud Storage

Laut Cloud: Cloud Storage Service Level Agreement (SLA), [4] wird die Verfügbarkeit von Google Cloud Storage als Service Level Agreement (SLA) angegeben, das die garantierte Verfügbarkeit für den Dienst definiert. Gemäß dem aktuellen SLA von Google Cloud Storage beträgt die garantierte Verfügbarkeit für Multi-Regionaler Speicher 99,95 Prozent und für Regionale Speicher 99,9 Prozent. Diese Zahlen geben an, dass Google Cloud Storage darauf ausgelegt ist, eine sehr hohe Verfügbarkeit zu gewährleisten. In der folgenden Tabelle werden diese Werte nochmals belegt.

Covered Service	Monthly Uptime Percentage
Standard storage class in a multi-region or dual-region location of Cloud Storage	>= 99.95%
Standard storage class in a regional location of Cloud Storage; Nearline, Coldline, or Archive storage class in a multi-region or dual-region location of Cloud Storage	>= 99.9%
Nearline, Coldline, or Archive storage class in a regional location of Cloud Storage; Durable Reduced Availability storage class in any location of Cloud Storage	>= 99.0%

Abbildung 2.2.2: Verfügbarkeit der Speicherklassen gemäß Google Cloud Storage SLA https://cloud.google.com/storage/sla

Es ist jedoch zu beachten, dass die tatsächliche Verfügbarkeit von Google Cloud Storage von mehreren Faktoren abhängt, einschließlich der spezifischen Konfiguration, dem Datenzugriffsmuster, der Netzwerkverfügbarkeit und anderen betrieblichen Variablen. Es ist daher möglich, dass die tatsächliche Verfügbarkeit in der Praxis leicht von der garantierten Verfügbarkeit abweicht.
Auch Google Cloud Storage bietet die Multi-Region Storage an. Dies ermöglicht die Speicherung von Daten in mehreren Regionen weltweit. Dadurch werden die Daten redundant repliziert und bleiben auch im Falle eines Ausfalls einer Region verfügbar.

Object Versioning bietet wie AWS die Möglichkeit Objektversionen beizubehalten. Dadurch können vorherige Versionen von Objekten wiederhergestellt werden, falls sie versehentlich gelöscht oder überschrieben werden.

Mit der Bucket- und Objekt Replikation können Daten automatisch in andere Regionen oder Projekte repliziert werden. Dies ermöglicht eine geografische Redundanz und verbessert die Verfügbarkeit der Daten. Google Cloud Storage verwendet verteilte Speichersysteme eine redundante Infrastruktur, um eine hohe Datensicherheit- und Verfügbarkeit zu gewährleisten. Die Daten werden in mehreren Datenzentren repliziert, um Ausfälle zu vermeiden.

Ein weiterer Punkt ist die Monitoring und Fehlererkennung wie Stackdriver Monitoring, um die Leistung und Verfügbarkeit von google Cloud Storage zu überwachen und auf potenzielle Probleme zu reagieren.

2.2.1.3 Kosten

In diesem Abschnitt erfolgt eine Untersuchung der Kostenstruktur von Amazon S3 und Google Cloud Storage. Dabei werden die generellen Kostenfaktoren betrachtet, für die die Cloud-Anbieter Gebühren erheben. Im Rahmen der Kostenanalyse werden die Daten von leoticket herangezogen, um eine grobe Einschätzung der zu erwartenden Kosten durchzuführen.

Amazon S3

Amazon S3 erhebt Gebühren für verschiedene Leistungsbereiche, darunter die Speicherung, Anfragen, Datenabrufe, Datenübertragung, Verwaltung, Analyse und die Replikation. Die Speicherungsgebühr richtet sich nach der Objektgröße, der Speicherdauer innerhalb eines Monats und der gewählten Speicherklasse. Es stehen verschiedene Speicherklassen zur Verfügung, die für unterschiedliche Anwendungsfälle geeignet sind, nämlich S3 Standard, S3 Intelligent-Tiering, S3 Standard – Infrequent Access, S3 One Zone – Infrequent Access, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval (ehemals S3 Glacier) und S3 Glacier Deep Archive. In dieser Arbeit wird S3 Glacier nicht behandelt, da leoticket keine Anwendung für diese Speicherklasse hat. Stattdessen werden die Speicherklassen S3 Standard, S3 Intelligent-Tiering, S3 Standard - Infrequent Access und S3 One Zone - Infrequent Access näher untersucht. Zur Veranschaulichung werden die Preise in der folgenden Tabelle dargestellt:

	Standard	Intelligent Tiering	Standard-IA	One Zone - IA
Speicher/GB im Monat	0.023 EUR	0.023 EUR	0.013 EUR	0.010 EUR
PUT-, COPY-, POST-, LIST-Anforderungen (Pro 1000 Anfragen)	0.0050 EUR	0.0050 EUR	0.0093 EUR	0.0093 EUR
GET-, SELECT- und alle anderen Anforderungen (pro 1000 Anfragen)	0.00040 EUR	0.00040 EUR	0.00093 EUR	0.00093 EUR
Datenabrufe	—	—	0.00093 EUR	0.00093 EUR
Monitoring und Automation pro Objekt	—	0,0000023 EUR	—	—
Datenübertragungen aus Amazon S3 in das Internet	0.084 EUR pro GB			

Abbildung 2.2.3: Übersicht der Kosten der AWS S3 Speicherklassen

In der vorliegenden Tabelle werden die Standardpreise für verschiedene Speicherklassen von Amazon S3 aufgeführt. Die Preise für PUT-, COPY-, POST- und LIST-Anforderungen basieren jeweils auf einer Staffelung von 1000 Anfragen, ähnlich wie bei GET-, SELECT- und anderen Anforderungen. Sowohl die Speicherklasse Standard als auch IA haben eine Speicherungsgebühr von 0,023 Euro und Anforderungsgebühren von 0,0050 Euro für POST und 0,00040 Euro für GET-Anforderungen. Für die Speicherklassen Standard-IA und OneZone-IA liegen die Speicherungsgebühren bei 0,013 Euro bzw. 0,010 Euro. Die Anforderungsgebühren betragen 0,0093 Euro für POST und 0,00093 Euro für GET. Zusätzlich erheben beide Speicherklassen eine Gebühr von 0,00093 Euro für Datenabrufe. Es gelten auch Datenübertragungsgebühren von Amazon S3 zum Internet in Höhe von 0,084 Euro pro GB für alle Speicherklassen. Es ist erwähnenswert, dass nur die IA-Speicherklasse

zusätzliche Gebühren für das Verwalten von Objekten in Höhe von 0,084 Euro pro Objekt erhebt.

Auch für die Replikation fallen in Amazon S3 Gebühren an. Bei der Replikation von Daten fallen zusätzlich zu den Speicherungskosten Kosten für die primäre Kopie der Daten, für Replikations-PUT-Anforderungen und für anwendbare Gebühren für den Speicherabruf mit seltenem Zugriff. Bei CRR wird auch für den regionenübergreifenden Datentransfer OUT von S3 zu jeder Zielregion gezahlt. Die Preise für Speicher- und PUT-Anfragen für die replizierte Kopie basieren auf den ausgewählten AWS-Zielregionen, während die Preise für Datenübertragungen zwischen den Regionen auf der AWS-Quellregion basieren. Wenn man S3 Replication Time Control nutzt, wird eine Datenübertragungsgebühr für die Replikationszeitsteuerung sowie Gebühren für S3-Replikationsmetriken erhoben, die zum selben Tarif abgerechnet werden wie angepasste Amazon-CloudWatch-Metriken. Die Kosten für die S3 Replication Time Control-Datenübertragung beträgt pro GB 0.015 USD. Unter AWS Preise können diese Informationen abgerufen werden.

Im Zusammenhang mit der Objektversionierung entstehen Kosten für die Beibehaltung älterer Versionen von Objekten, da zusätzlicher Speicherplatz benötigt wird. Es fallen keine direkten Kosten für das Tagging von benutzerdefinierten Metadaten an, die den Objekten zugeordnet sind. Es ist jedoch möglich, dass Kosten für das Abrufen von Tags über die S3-API (z. B. mit den ListObjects- oder GetObject-Operationen) anfallen, da dies als Datenabruf betrachtet wird und entsprechende Gebühren gemäß den AWS-Preisen für Datenzugriff erhoben werden. Es sollte beachtet werden, dass die genauen Preise für Objektversionierung und Tagging in Amazon S3 im Laufe der Zeit Änderungen unterliegen können.

Die genauen Preise und Kostendetails für S3 können sich im Laufe der Zeit ändern. Es wird empfohlen, die aktuellsten Informationen auf der offiziellen AWS-Preisseite oder im AWS-Kostenrechner unter AWS Calculator zu überprüfen, um eine grobe Kosteneinschätzung zu erhalten.

GC Storage

GC Storage setzt Preise für die Komponenten Datenspeicher, Datenverarbeitung und Netzwerknutzung. Beim Datenspeicher ist wie bei Amazon S3 die Menge der in den Buckets gespeicherten Daten bedeutend. Preise für Speicher hängen von der Speicherklasse der Daten und dem Standort der Buckets ab. Bei der Datenverarbeitung werden Gebühren für die von Cloud Storage durchgeführten Verarbeitung, einschließlich Vorgangsgebühren, anwendbarer Abrufgebühren und Replikation zwischen Regionen erhoben. Bei der Netzwerknutzung werden Gebühren für die Menge der aus den Buckets gelesenen oder zwischen diesen verschobenen Daten erhoben. Google Cloud Storage bietet vier Speicherklassen an von denen drei untersucht werden. Die Standard, Nearline und Coldline. Diese drei Speicherklassen haben unterschiedliche Leistungseigenschaften und Kosten. Die Preise variieren je nach gewählter Speicherklasse.

Um den Vergleich zwischen Amazon S3 und Google Cloud Storage zu veranschaulichen wird eine Tabelle in der unteren Abbildung dargestellt:

	Standard	Nearline	Coldline
Speicher/GB im Monat	0.021 EUR	0.012 EUR	0.0054 EUR
Vorgänge Klasse A pro 1000 Vorgänge	0.0047 EUR	0.0093 EUR	0.0093 EUR
Vorgänge Klasse B pro 1000 Vorgänge	0.00037 EUR	0.00093 EUR	0.0047 EUR
Abrufgebühren	—	0.0093 EUR pro GB	0.019 EUR pro GB
Ausgehender Traffic von Google Cloud ins Internet	0.11 EUR pro 0 bis 1TB monatlich		

Abbildung 2.2.4: Vergleich der Kosten von Google Cloud Storage Speicherklassen

Ähnlich wie bei Amazon S3 sind die Speicherkosten der Nearline und Coldline geringer als die Standard Speicherklasse. Dafür werden Gebühren für die Datenabrufe bei Nearline und Coldline erhoben. Die verschiedenen Speicherklassen sind für unterschiedliche Anwendungsfälle ähnlich wie bei Amazon S3 konzipiert. Die Standard Storage-Klasse bietet hohe Performance, niedrige Latenzzeiten und hohe Verfügbarkeit. Sie eignet sich gut für häufig genutzte Daten, bei denen schneller Zugriff und gerine Latenz von Bedeutung sind. Beispiele dafür sind aktive Anwendungen, Datenbanken oder Inhalte mit hohem Durchsatz. Die Nearline Storage-Klasse ist für seltener genutzte Daten konzipiert, auf die jedoch mit niedriger Latenzzeit zugegriffen werden muss. Es bietet niedrigere Speicherkosten als die Standard Storage, jedoch mit einer etwas längeren Zugriffszeit. Es eignet sich für Backup-Daten, Archivierung und lange Speicherung. Letztere Speicherklasse, die Coldline Storage ist für Daten ausgelegt, auf die selten zugegriffen wird und bei denen eine längere Zugriffszeit akzeptabel ist. Es bietet die niedrigsten Speicherkosten an. Sie eignet sich gut für langfristige Archivierung, Compliance-Daten und Backup-Daten.

Zusätzliche Kosten, die nicht in der Tabelle aufgelistet sind, sind die Kosten für die Replikation der Daten. Auf Wunsch können Nutzer in Cloud Storage Daten innerhalb einer Region, in Dual-Regionen oder Multiregionen replizieren. Zusätzlich zu den Daten, die in den hochgeladenen Objekten enthalten sind, werden benutzerdefinierte Metadaten auf die monatliche Speichernutzung

angerechnet. Für die benutzerdefinierten Metadaten "NAME:VALUEërfasst Google Cloud Storage beispielsweise jedes Zeichen in NAME und VALUE als Byte, das mit dem Objekt gespeichert wird.

Beim vorzeitigen Löschen eines Objektes aus den Speicherklassen Coldline und Nearline werden Gebühren erhoben, da eine Mindestspeicherdauer angesetzt ist. Das vorzeitige Löschen beim Nearline beträgt pro GB pro Tag ca. 0.00043 USD und beim Coldline 0.0002 USD. Auch für Tags fallen Gebühren pro Monat von 0.005 USD an, dass für Buckets angehängt werden.

Auch hier fallen Kosten für die Objektversionierung an. Die Kosten setzen sich aus zwei Hauptkomponenten zusammen. Einmal den Speicher und die Anfragen.Jede Version eines Objekts belegt Speicherplatz im Bucket. Die kosten basieren auf der Größe der Objekte und der Anzahl der Versionen, die gespeichert sind. Das Hochladen, Herunterladen oder LÖschen von Objektversionen führt zu Anfragen an den Storage-Dienst. Für diese Anfragen können GEbühren anfallen, die sich nach der Anzahl der Anfragen richten. Es ist wichtig zu beachten, dass die Kosten für die Objektversionierung zusätzlich zu den regulären Kosten für die Speicherung und den Datenverkehr in Google Cloud Storage anfallen. Daher sollten die potenziellen Kosten der Objektversionierung in den Kalkulation einbezogen werden.

2.2.1.4 Performance

Amazon S3

Amazon S3 bietet verschiedene Funktionen und Dienste, die die Performance und Skalierbarkeit der Datenzugriffe verbessern. In der offiziellen Dokumentation S3: Performance Guidelines for Amazon S3, [15] werden verschiedene Best Practices Guidelines empfohlen, die die Leistung erhöhen können. Es wird empfohlen HTTP Analyse Tools zu verwenden, um die Leistung von DNS lookup times, Latenzen und die Datentransfergeschwindigkeiten zu messen. performance-guide

Eine andere Methode, die die Leistung erhöhen kann, ist die horizontale Skalierung von Connections. Da Amazon S3 als ein sehr großes verteiltes System gilt, können dadurch Requests über getrennte Verbindungen verteilt werden, um auf die maximale Bandbreite zu kommen. „Amazon S3 doesn't have any limits for the number of connections made to your bucket.“, ebd. performance-guide
Außerdem verspricht Amazon S3, dass Requests beim wiederholten Mal wahrscheinlicher erfolgreich und schneller sind, da sie einen anderen Pfad als beim ersten Request nehmen.„[...] if the first request is slow, a retried request is likely to take a different path and quickly succeed.“, ebd. performance-guide

Weitere Funktionen die für die Leistungserhöhung beitragen sind die S3 Transfer Acceleration, S3 Select und die S3 Cross-Replication. Die S3 Transfer Acceleration ermöglicht schnelle, einfache und sichere Übertragungen von Dateien über große geografische Distanzen hinweg zwischen dem Client und S3 Buckets. Die Daten werden über eine optimierte Route an Amazon S3 weitergeleitet. Diese Funktion ist für Daten in Größe von Gigabytes zu Terabytes geeignet, die regelmäßig verschickt werden müssen. Um die Geschwindigkeit zu messen, bietet Amazon S3 die S3 Transfer Acceleration Speed Comparison Tool an, um beschleunigte und nicht-beschleunigte Uploads zu messen.

S3 Select ermöglicht das effiziente Abrufen von spezifischen Daten aus Objekten in Amazon S3. Anstatt ein gesamtes Objekt herunterladen zu müssen, können mit S3 Select nur die benötigten Daten abgefragt werden. Dies reduziert den Datenverkehr und beschleunigt den Abrufvorgang erheblich, insbesondere bei großen Daten.

S3 Cross-Region Replication ermöglicht die Replikation von Daten zwischen verschiedenen AWS-Regionen. Durch die Replikation in geografisch entfernte Regionen kann die Leistung verbessert werden, indem Daten näher an den Benutzer oder Anwendungen bereitgestellt werden. Dadurch werden niedrigere Latenzzeiten und eine bessere Verfügbarkeit erzielt.

Die AWS SDK stellt eine einfache API und wird regelmäßig geupdatet, um die neuesten Technologien anzubieten. Die SDK beinhaltet die automatische Retry Request bei HTTP 503 Fehlern. Es beinhaltet auch den Transfer Manager, der dafür sorgt Connections automatisch horizontal zu skalieren. Damit können tausende von Requests pro Sekunde erreicht werden.

Google Cloud Storage

Google Cloud Storage ermöglicht die Auswahl des geeigneten Speicherorts für Daten, um die Latenzzeiten zu minimieren. Man kann zwischen multi-regionalen Speicherstandorten oder regionalen Speicherstandorten wählen, um Daten näher an den Benutzer oder Anwendungen zu speichern und den Zugriff zu beschleunigen.

Durch die Integration mit Content Delivery Networks (CDNs) kann die globale Verteilung von Daten optimiert und die Bereitstellung beschleunigen. Das CDN stellt eine Zwischenspeicherung von Inhalten in Edge-Servern weltweit bereit, um den Zugriff auf Daten schneller und effizienter zu machen.

Bei der Performance spielt die Request Rate auch eine bedeutsame Rolle. Das bietet das Auto-Scaling von GCP an. Cloud Storage ist ein multi-tenant Service, was bedeutet, dass Benutzer die zugrunde liegenden Ressourcen gemeinsam nutzen. Um die gemeinsam genutzten Ressourcen optimal zu nutzen, haben Buckets eine anfängliche I/O Kapazität."Cloud Storage is a multi-tenant service, meaning that users share the same set of underlying resources.[...]", gcp-autoscale Cloud: Request rate and access distribution guidelines: Auto Scaling, [5] (Übersetzung aus Google Cloud)

> Diese Kapazitäten betragen etwa 1000 Schreibzugriffsanfragen pro Sekunde für Objekte, einschließlich Hochladen, Aktualisieren und Löschen von Objekten. Es ist zu beachten, dass Cloud Storage auch eine kleinere Begrenzung für wiederholte Schreibvorgänge mit demselben Objektnamen hat. Auf Lesezugriffsanfragen po Sekunde betragen sie bei 5000, einschließlich Auflisten von Objekte, Lesen von Objektdaten und Lesen von Objektmetadaten, vgl. gcp-autoscale ebd.

Wenn die Anfragehäufigkeit für einen bestimmten Bucket steigt, skaliert Cloud Storage automatisch und erhöht die I/O Kapazität für diesen Bucket, indem die Anfragelast auf mehrere Server verteilt wird.

Cloud Storage ermöglicht den parallelen Upload und Download von Daten, um die Übertragungsgeschwindigkeit zu maximieren. Es können mehrere Threads oder Prozesse verwenden werden, um Daten gleichzeitig hochzuladen oder herunterzuladen und so die Leistung zu verbessern.

Der Google Cloud Storage Transfer Service ermöglicht den schnellen und effizienten Transfer von großen Datenmengen. Daten können aus anderen Cloud-Speicherlösungen, On-Premise-Speichern oder öffentlichen Datensätzen in Google Cloud Storage übertragen werden, um Zeit und Bandbreite zu sparen.

Während die out-of-the-box Performance bereits stabil ist, gibt es einige Einstellungen und Empfehlungen von Google Cloud, um die Performance auf den Anwendungsfall zu optimieren. Umm die Leistung messen zu können, bietet Google Cloud die "perfdiag"Tool an. Der Autor McAnlis empfiehlt in seinem Block [gcp-blog] Optimizing your Cloud Storage performance: Google Cloud Performance Atlas dieses Tool zu verwenden, dass eine Reihe von Tests durchläuft, die die aktuelle Performance von einem Cloud Bucket protokolliert. Des Weiteren empfiehlt er die Nutzung des "gsutil"Tools von Google Cloud, um kleinere Dateien schneller hochzuladen. Wenn man 20 000 Dateien, die jeweils 1kb groß sind, hochladen möchte, dann dauert der Overhead bei jedem individuellen Upload länger als die gesamte Uploadzeit gemeinsam. Deshalb sollte man Batch Operationen verwenden, da diese den Overhead reduzieren und die Leistung verbessern. Das gsutil Tool bietet die Option an Batch Operationen durchzuführen. Auf dem folgenden Diagramm wird ein Test mit 100 mal 200 000 Dateien mit individuellem Upload und Batch Upload angezeigt, das mit "gsutil -m cpïn ein Storage Bucket hochgeladen wird.

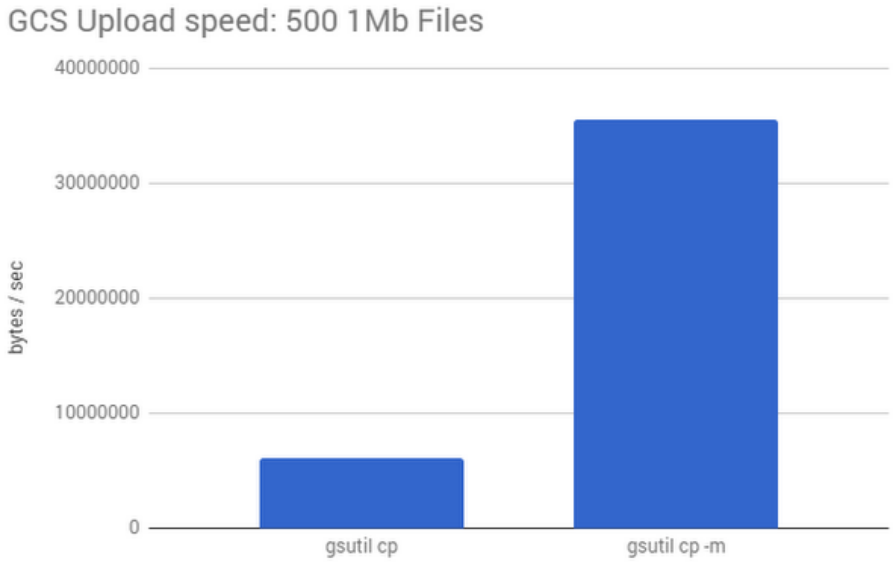

Abbildung 2.2.5: GCS Upload Geschwindigkeit, [gcp-blog] https://cloud.google.com/blog/products/gcp/optimizing-your-cloud-storage-performance-google-cloud-performance-atlas/

Hier sieht man, wie die Leistung sich dabei um das fünffache erhöht im Gegensatz zu den inidivduellen Uploads.

Das Auto-balancing von Google Cloud sorgt für die Verteilung der Upload-Connections auf "Backend Shards". Das funktioniert durch die Name/Pfad der Datei und kann die Upload Geschwindigkeit erhöhen, wenn sich die Dateien in unterschiedlichen Ordnern befinden oder auch verschiedene Namensgebungen haben. Falls sich die Dateien zu sehr ähneln, kann dies die Upload Geschwindigkeit reduzieren, da die Connections alle in den gleichen Shard übergehen.

Eine weitere Methode, die die Performance steigern kann, sind Requests in Größe ab 1MB. Bei kleineren Requests sollte man die Requests versuchen zu parallelisieren, damit die fixen Latenzkosten überlappen.

Es ist zu beachten, dass die Performance auch von anderen Faktoren abhängt, wie die Anwendungsarchitektur, die Netzwerkverbindung und die Implementierung in der Anwendung. Es können spezifische Konfigurationsoptionen genutzt werden, um die Leistung weiter zu optimieren, wie Caching, asynchrone Operationen oder die Nutzung von optimierten Bibliotheken oder Frameworks.

2.2.1.5 API Anbindung

Amazon S3

Die API-Anbindung von Amazon S3 ist umfangreich und bietet Entwicklern eine Vielzahl von Möglichkeiten zur Interaktion mit dem Speicherdienst. Amazon S3 bietet eine RESTful API (Application Programming Interface) sowie verschiedene SDKs (Software Development Kits) und Tools, die die Integration und Nutzung erleichtern.

Amazon S3 bietet eine RESTful API, die auf dem HTTP-Protokoll basiert. Mit dieser API können Entwickler HTTP-Anfragen wie GET, PUT, POST und DELETE verwenden, um auf Buckets und Objekte in Amazon S3 zuzugreifen, sie zu erstellen, zu lesen, zu aktualisieren und zu löschen. Jedoch empfiehlt Amazon die Nutzung der AWS SDK, da man bei der REST API den nötigen Code für die Kalkulierung der validen Signatur schreiben muss, um sich zu authentifizieren.„It requires you to write the necessary code to calculate a valid signature to authenticate your requests.“, aws-api S3: Amazon S3 REST API Introduction, [13].

Die AWS SDK authentifziert Requests durch die Bereitstellung der Access Keys, so fällt das Schreiben des Codes dafür weg. Die SDK ist in verschiedenen Programmiersprachen zugänglich, daruner Java, Javascript, Python, .NET, Ruby, PHP, IOS und Android. Die folgende Tabelle zeigt alle unterstützten Programmiersprachen an.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for Swift	AWS SDK for Swift code examples

Abbildung 2.2.6: Unterstützte Programmiersprachen von AWS SDK, aws-sdk https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html

Sie bietet eine abstrakte Schnittstelle, um die Entwicklung von Anwendungen zu erleichtern und die Interaktion mit Amazon S3 zu vereinfachen. Darüber hinaus gibt es auch Drittanbieter-Tools und Open-Source Bibliotheken, die die Integration mit Amazon S3 unterstützen. Spring Boot stellt eine Bibliothek für die alte und neue Version von AWS SDK zur Verfügung. Intellij Idea bietet einen Plugin „AWS Toolkit“an, mit dem man seine Amazon Credentials zur Verfügung stellen kann. Diesen Plugin gibt es auch für Eclipse und Visual Studio Code.

Außerhalb der AWS SDK stellt Amazon S3 auch die AWS CLI (Command Line Interface) bereit, um API Calls durchzuführen. Amazon S3 ermöglicht die Durchführung von Batch-Operationen,

um mehrere Objekte in einem einzigen API-Aufruf zu verarbeiten. Dies erleichtert die effiziente Verarbeitung großer Datenmengen und reduziert die Anzahl der API-Anfragen.

Die API-Anbindung von Amazon S3 erfordert eine geeignete Authentifizierung und Autorisierung, um sicherzustellen, dass nur autorisierte Benutzer auf die Daten zugreifen können. Dies erfolgt in der Regel mithilfe von Zugriffsschlüsseln, die über AWS Identity and Access Management (IAM) verwaltet werden.

Google Cloud Storage

Auch Google Cloud Storage bietet umfangreiche API-Integrationen, um die Einbindung in eigene Anwendungen zu ermöglichen. Eine davon sind die Client Libraries für verschiedene Programmiersprachen wie Java, Python, Node.js, Go, Ruby und .NET. Diese Bibliotheken erleichtern die Integration von Google Cloud Storage in Anwendungen und bieten eine benutzerfreundliche Schnittstelle für die Interaktion mit den Speicherressourcen.
Auch in Cloud Storage wird eine RESTful API ähnlich wie in Amazon S3 angeboten. Diese ermöglicht es Entwicklern über HTTP-Anfragen auf die Speicherressourcen zuzugreifen. Sie unterstützt CRUD Operationen für Buckets und Objekte sowie erweiterte Funktionen wie das Setzen von Metadaten, das Verwalten von Zugriffssteuerungen und das Durchführen von Batch-Anfragen.

Google Cloud Storage bietet sowohl JSON- als auch XML-APIs für die Interaktion mit dem Speicherdienst. Die JSON-API ist die bevorzugte Option und bietet eine moderne, leichtgewichtige Syntax für die Kommunikation mit dem Dienst anstelle der RESTful API. Neben der direkten API-Anbindung stellt Google auch das Google Cloud SDK zur Verfügung, das verschiedene Befehlszeilentools enthält, mit denen auf Speicherressourcen zugegriffen werden und sie verwalten können. Das SDK bietet auch eine Reihe von Clientbibliotheken für verschiedene Programmiersprachen. Neben den Client-Bibliotheken werden auch Tools wie gsutil (ein Befehlszeilen-Dienstprogramm), das ermöglicht, Cloud Storage von der Befehlszeile aus zu verwalten, sowie Cloud Storage FUSE, das es ermöglicht, Cloud Storage als Dateisystem zu mounten und direkt darauf zuzugreifen.

Die API-Anbindung von Google Cloud Storage erfordert ebenfalls eine Authentifizierung und Autorisierung, um sicherzustellen, dass nur autorisierte Benutzer auf die Daten zugreifen können. Dies wird mithilfe von Google Cloud IAM (Identity and Access Management) verwaltet, das Zugriffsrichtlinien und Rollenverwaltung bietet.

2.2.2 Bereitstellung der Dateien

In diesem Abschnitt werden Methoden für die Bereitstellung der Dateien durch signierte URL's von Amazon S3 und Google Cloud präsentiert. Unternehmen stehen und Organisationen stehen vor der Herausforderung, Dateien sicher und effizient für ihre Benutzer bereitzustellen. Eine häufig verwendete Methode, um diesen Anforderungen gerecht zu werden, ist die Verwendung von signierten URL's.

Signierte URLs ermöglichen es, auf einfache Weise temporäre Zugriffsrechte für Dateien zu gewähren, ohne dass komplexe Zugriffskontrollmechanismen implementiert werden müssen. Sowohl Amazon Web Services (AWS) S3 als auch Google Cloud Storage bieten die Möglichkeit, signierte URLs für die Dateibereitstellung zu generieren.

Amazon S3

In Amazon S3 sind alle Objekte und Buckets standardmäßig privat. Durch die Verwendung von presigned URLs können Objekte durch Links geteilt werden ohne AWS Sicherheitscredentials.„All objects and buckets are private by default. However, you can use a presigned URL to [...]“, aws-signed-urls S3: Using presigned URLs, [16].

> Presigned URLs werden für die Generierung von Links verwendet, um auf S3 Buckets und Objekte zugreifen zu können. Bei der Erstellung eines solchen URLs können andere Nutzer von außerhalb AWS ohne Credentials auf die gewünschten Daten zugreifen. Jeder der diesen Link hat, kann darauf zugreifen und Aktionen ausführen. Der Link wird bei Konfiguration nach einer bestimmten Zeit die Gültigkeit verlieren. Amazon S3 überprüft das Verfallsdatum des Links während des HTTP Requests, vgl. aws-signed-urls ebd.

Durch die Nutzung des URL können Nutzer entweder ein Objekt lesen oder ein Objekt hochladen. In dem Fall von leoticket liegt das Lesen des Objekts im Fokus. Die URL beinhaltet spezielle Parameter, welches von einer Anwendung gesetzt werden. Es gibt drei Parameter um den Zugriff für den Nutzer einzuschränken. Diese sind einmal Einschränkung des Buckets. Das bedeutet, das ein Nutzer nicht auf jeden Bucket zugreifen kann, sondern nur auf den Bucket, indem sich das Objekt befindet. Der zweite Parameter ist der Name des Objekts und zuletzt die Zeitspanne in der die URL gültig ist. Sobald die Zeit abgelaufen ist, kann der Nutzer nicht mehr auf den Link zugreifen.

Für leoticket bedeutet das, Tickets und Rechnungen nicht mehr als Email-Anhänge bereitstellen zu müssen. Durch die Methode der presigned URLs können diese Links über die Email an die Clients versendet werden.

Google Cloud Storage

In Google Cloud Storage funktioniert die signed URL mit dem ähnlichen Prinzip wie in Amazon S3. Durch zeiteingeschränkte Links können Nutzer, die den Link besitzen, auf Objekte und Buckets zugreifen ohne Google Cloud Credentials. Dabei bietet Cloud Storage mehrere Methoden zur Generierung der Signed URL an. Die V4 signing with service account authentication, Signing with HMAC authentication und die V2 signing with service account authentication. Letztere Methode wird ausgeschlossen, da sie von Google Cloud selbst nicht empfohlen wird. Ein Beispiel wie eine signed URL aussehen kann:

2.3 Auswahl des Speichersystems

In diesem Kapitel werden anhand der Anforderungen an leoticket Konfigurationseinstellungen empfohlen. Um der Anforderung der sicheren Speicherung an leoticket zu entsprechen bieten beide Cloud Provider verschiedene Funktionen an, die im vorherigen Kapitel unter Eigenschaften untersucht worden sind. Beide Cloud Provider bieten ähnliche Funktionen für die Erstellung von Benutzern, Gruppen und Rollen um den Zugriff auf die Dienste einzuschränken. Die IAM Funktion bei beiden Cloud Providern ermöglicht es den Zugriff auf S3 und Cloud Storage zu beschränken. Bei der IAM-Funktion in AWS S3 werden einige bewährte Vorgehensweisen empfohlen, um die Sicherheit und den Zugriff auf S3-Ressourcen zu verbessern. Zum einen das Prinzip des geringsten Privilegs. Benutzern und Anwendungen sollten nur die Berechtigungen, die sie zum Durchführen Ihrer Aufgaben benötigen, gewähren. Übermäßige Berechtigungen, um potenzielle Sicherheitsrisiken zu minimieren, sollten vermieden werden. Für leoicket bedeutet dies, wenn eine Anwendung Dateien nach S3 hochladet, dann sollte diese Anwendung nur die benötigten Rechte haben. Dies wären beispielsweise Rechte für das Schreiben und Abrufen von Objekten. Andere Rechte wie beispielsweise das Hinzufügen von Policies können ebenfalls vergeben werden. Diese Empfehlung gilt auch für die IAM Funktion in Google Cloud. Für die Authentifizierung für Anwendungen gibt es Service Accounts, die spezielle Rechte haben, die für die Anwendung relevant sind. Für jeden Dienst oder Anwendung kann ein eigener Service Account erstellt werden.

Auch bei der Datenverschlüsselung bieten beide Cloud Provider ähnlichen Möglichkeiten an, mit Encryption Keys umzugehen. Da leoticket eine sichere Speicherung anfordert, kommt die SSE-KMS customer-managed Variante in Frage. Durch die selbstständige Generierung des Schlüssels und Verwaltung, hat der Nutzer mehr Kontrolle im Gegensatz zum S3-managed oder Google-managed Keys. Die generierten Schlüssel werden jedoch bei den Cloud Providern im Key Management Service gespeichert. Wenn für SSE KMS entschieden wurde, kann die Funktion "Bucket Keysïn S3 aktiviert werden. Dadurch hat man einen Master Key, der die Schlüssel der einzelnen Objekte verschlüsselt. Dies reduziert die Request Kosten bis zu 99 Prozent, indem die Request Traffic von S3 zu AWS KMS reduziert wird.

Um noch unabhängiger vom Cloud Provider zu sein, kann die Variante customer-supplied in Google Cloud oder die customer-managed Variante in S3 in Frage kommen. Hier hat der Nutzer die volle Verantwortung für die Schlüssel. Das bedeutet, die Schlüssel werden extern vom Nutzer generiert, verwaltet und gespeichert. Dabei besteht das Risiko, den Schlüssel zu verlieren und nicht mehr auf die Objekte in S3 zugreifen zu können. Außerdem ist der Aufwand für die selbstständige Verwaltung des Schlüssels höher als die Verwendung des KMS.

Beide Cloud Provider bieten die Möglichkeit, ACLs auf Objekte anzuwenden. Das bedeutet, das für jedes Objekt Zugriffskontrollen eingestellt werden. Jedoch wird bei beiden Cloud Providern empfohlen diese Einstellung deaktiviert zu lassen und nur bei besonderen Fällen zu verwenden.

Für die Sicherheit und Verfügbarkeit der Daten wird empfohlen Object Versioning zu aktivieren, um mehrere Versionen von Objekten bereitzustellen. Dies kann Objekte beim versehentlichen Löschen oder Überschreiben wiederhergestellt werden. Beide Cloud Provider bieten diese Funktion beim Erstellen eines Buckets an.

AWS und Google Cloud bieten eine Auswahl von Speicherklassen zur Verfügung. Beide Provider versprechen eine Verfügbarkeit von mindestens 99.9 Prozent.
Standard-IA und One Zone-IA bieten kostengünstigere Optionen für die Speicherung der Daten. Die Standard-IA ist für Daten geeignet, auf die weniger häufig zugegriffen wird, aber bei Bedarf schnell verfügbar sein müssen. Im Vergleich zur Standardklasse sind die Kosten für die Speicherung niedriger, aber es fallen zusätzliche Gebühren für den Datenabruf an. Wenn Daten von zwei Speicherklassen hergeschoben werden, dann fallen Kosten für den Speicherklassenwechsel an. Diese hängen von der Datenmenge ab, die verschoben werden.

Die One Zone-IA ist für Daten geeignet, auf die selten abgerufen werden, jedoch mit einer Einschränkung. Anders als bei den anderen Speicherklassen wird One Zone-IA nur in einem einzelnen Availability Zone in einer bestimmten Region gespeichert. Dies bedeutet, dass die Daten in einer einzigen AZ verfügbar sind und ein potenzieller Datenverlust auftreten kann, wenn diese AZ nicht verfügbar ist.

Die Speicherklasse Standard-Infrequent Access von S3 und Nearline von Cloud Storage können für leoticket empfohlen werden. Da in leoticket die Anforderungen an das Abrufen von Dateien innerhalb Millisekunden liegt, fallen die Speicherklassen von S3 Glacier weg, da das Abrufen von Dateien Tage oder Stunden dauern kann. Die Daten in leoticket müssen auf Abruf zur Verfügung stehen, jedoch wird davon ausgegangen, das Objekte im Durchschnitt zweimal abgerufen werden. Einmal von der Anwendung und einmal von den Kunden, die die Tickets gekauft haben. Durch die geringe Abrufanzahl eines einzelnen Objekts und die 10 Jahre Speicherung von Daten kommen die Speicherklassen Standard-IA und Nearline in Frage. Diese Speicherklassen sind für Daten geeignet, auf die weniger häufig zugegriffen werden, aber bei Bedarf schnell verfügbar sein müssen. Sie bieten gleichzeitig eine kostengünstigere Option für die Speicherung der Daten. Die One Zone-IA von S3 speichert die Daten in nur einer AZ und wird aufgrund des Risikos der Verringerung der Verfügbarkeit bei Ausfall der AZ nicht empfohlen. Dadurch kann das Abrufen der Daten länger dauern. Die Coldline Speicherklasse von Google Cloud kann in Frage kommen, wenn Daten für eine sehr lange Zeit nicht mehr abgerufen worden sind und auch in nächster Zeit nicht mehr zugegriffen werden wird. Denn die Coldline ist für Daten ausgelegt, auf die selten zugegriffen werden und bei denen eine längere Zugriffszeit akzeptabel ist. Sie eignet sich gut für die Archivierung.

Bei der API Integration unterscheiden sich die beiden Provider in wenigen Punkten. Für leoticket ist es wichtig, dass die API in die leoticket Anwendung integrierbar ist, ohne viel Aufwand. Dies ist bei beiden Providern der Fall, denn beide bieten SDKs, CLIs und REST APIs an, die leicht in die eigene Anwendung integrierbar sind. Frameworks wie Spring Boot bieten Dependencies zu der AWS SDK Version 1 und 2 und auch für Google Cloud an. AWS hat den Vorteil, dass On-Premise Anwendungen wie MinIO die AWS API verwendet und so eine S3-Kompatibilität verschafft. Dies hat den Vorteil, dass man sich bereits mit der API auskennt und über MinIO Dateien nach S3 hochladen und herunterladen kann. Dies ist zwar auch über Google Cloud möglich, aber wenn man sich noch nicht mit der AWS API auskennt, ist es nötig sich in die Technologie einzuarbeiten. Um sich mit der API authentifizieren zu können ist es empfehlenswert bei der Google SDK Service Accounts zu verwenden. Jedoch gibt es auch andere Methoden zur Authentifizerung. Über die ADC in Google Cloud kann entschieden werden mit welcher Methode authentifziert wird. AWS

bietet für die Authentifzierung das „AWS Toolkit “an, dass von IntelliJ, Eclipse und anderen IDEs unterstützt wird. Über das Toolkit ist die Authentifzierung empfohlen.

Um die beste Performance für leoticket zu gewährleisten ist es ratsam, Performance Analysen durchzuführen. Dafür gibt es Tools, die AWS und Google Cloud anbieten. Bei Google Cloud kann man mit dem "gsutil"CLI eine Performance Diagnose durchführen. Dies muss auch durchgeführt werden, wenn man den Support von Google Cloud kontaktiert, um eine Fehlerbehebung zu finden. AWS bietet in diesem Punkt Metrics direkt in dem S3 Bucket an. Es werden verschiedene Widgets für Request Metrics bereitgestellt, die man analysieren kann. Im Kapitel "Prototypische Umsetzung"werden Messungen für die Performance Analyse durchgeführt und anschließend analysiert, um die beiden Cloud Anbieter in diesem Punkt besser vergleichen zu können.

Die Entscheidung zwischen den Cloud Providern liegt sowohl an den Anforderungen an leoticket als auch an den Präferenzen der Entwickler. Wenn sich Entwickler bereits mit einem Cloud Provider auskennen und beschäftigt haben, dann ist es ratsam sich für diesen Provider zu entscheiden. Dies hat den Vorteil, dass man sich nicht auf eine neue Technologie einarbeiten muss und Zeit sparen kann bei der Implementierung und Integration der API. Es ist legitim zu sagen, das beide Anbieter viele Funktionen und ähnliche Produkte anbieten, die auf die Anforderungen von Leoticket angepasst werden können. Diese Arbeit dient lediglich dazu, eine Hilfestellung bei der Entscheidung zu sein, um sich am Ende für eines entscheiden zu können.

2.3.1 Kostenanalyse

Bei der Kostenanalyse werden die Gebühren aus dem Kosten Abschnitt übernommen und eine Kostenabschätzung für Amazon S3 und Cloud Storage durchgeführt. Es werden die Kosten für die Speicherung, Abrufen und Datenübertragung verglichen und dargestellt. Die Berechnungsdaten stammen aus leoticket und dienen der groben Kosteneinschätzung. Die Kosten können von verschiedenen Faktoren voneinander abweichen. Der aktuelle Stand der Kosten ist gleich dem Abgabedatum der Arbeit.

Für die grobe Einschätzung der Kosten wird von 800 000 Bestellungen im Durchschnitt über leoticket pro Jahr ausgegangen. Pro Monat sind das ca. 67 000 Bestellungen mit jeweils 4 Objekten pro Bestellung. Die Größe eines Objektes beträgt dabei im Durchschnitt 100kb. Pro Bestellung werden im Durchschnitt drei Tickets gekauft, wobei noch eine Rechnung hinzugefügt wird. Diese 4 Objekte machen eine Bestellung mit insgesamt 400kb aus. Es wird davon ausgegangen, dass es monatlich ungefähr 268 000 POST Requests nach Amazon S3 verschickt werden. Außerdem wird davon ausgegangen, dass Objekte jeweils zweimal im Monat abgerufen werden können. Einmal von Leomedia und von den Kunden aus. Insgesamt werden es 536 000 GET Requests von Amazon S3 aus geben. Jedes Objekt wird separat hochgeladen.

Die Speichergröße ergibt sich aus den 67 000 Bestellungen pro Monat mal die 400kb Größe pro Bestellung. Diese werden geteilt durch 1024 und das Ergebnis nochmal durch 1024 um die GB Einheit zu bekommmen. Das ergibt 25GB pro Monat an Speicher. Wenn man die Retention von 10 Jahren dabei bedenkt, dann sind das ungefähr 3TB Daten die gespeichert werden müssen, wenn Daten erst nach 10 Jahren gelöscht werden. Mit diesen gewonnen Daten werden die Preisrechner von AWS und Google Cloud verwendet und die entsprechenden Daten eingesetzt.

AWS Preisrechner
GC Preisrechner

Amazon S3

Im Folgenden wird die folgende Tabelle bereitgestellt, um die Kosten für die verschiedenen Speicherklassen darzustellen.

	Standard	Intelligent Tiering	Standard-IA	One Zone - IA
Monatliche Kosten der Speicherklasse	70.27 EUR	70.27 EUR	38.72 EUR	30.98 EUR
Vorauszahlung	162.42 EUR	162.42 EUR	300.76 EUR	300.76 EUR
PUT requests	1.35 EUR	1.35 EUR	2.50 EUR	2.50 EUR
GET requests	0.22 EUR	0.22 EUR	0.50 EUR	0.50 EUR
Data Retrieval	–	–	0.47 EUR	0.47 EUR
Data Transfer	4.20 EUR			
Monitoring und Automation objects	–	75.19 EUR	–	–

Abbildung 2.3.1: Übersicht der einzelnen Kosten der Datenspeicherung in Amazon S3

Zuerst werden die Einheitsumwandlungen durchgeführt. Dabei werden die angegebenen 3TB im Monat mal die 1024GB multipliziert, um die exakte Menge an GB auszurechnen. Dies ist bei 3072GB. Danach wird die Objektgröße von 100kb in GB umgerechnet, was 0.000095367432 GB ergibt. Anschließend werden die Preise kalkuliert.

$$\frac{3072 \text{ GB per Month}}{0.000095367432 \text{ GB average item size}} = 32.212.255 \text{ number of objects} \quad (3.1)$$

$$3072 \text{ GB} \times 0.013 \text{ EUR} = 41.472 \text{ USD Standard-IA costs} \quad (3.2)$$

$$268.000 \text{ PUT requests} \times 0.00001 \text{ USD per request} = 2.68 \text{ USD (PUT requests cost)} \quad (3.3)$$

$$536.000 \text{ GET requests} \times 0.000001 \text{ USD per request} = 0.536 \text{ USD (GET requests cost)} \quad (3.4)$$

$$50 \text{ GB} \times 0.01 \text{ USD} = 0.50 \text{ USD (data retrievals cost)} \quad (3.5)$$

$$41.472 \text{ USD} + 2.68 \text{ USD} + 0.536 \text{ USD} + 0.50 \text{ USD} = \underline{\textbf{45.19 USD (Total Standard-IA)}} \quad (3.6)$$

$$32.212.255 \text{ number of objects} \times 0.00001 \text{ USD} \quad (3.7)$$

$$= 322.12 \text{ USD (Cost for PUT, COPY, POST requests for initial data)} \quad (3.8)$$

Die obigen Formeln sind ähnlich wie bei der Standard und One Zone-IA, wenn man dabei die jeweiligen Gebühren aus dem Kosten Abschnitt übernimmt. Bei der Intelligent Tiering gibt es noch den Unterschied, dass die Kosten für Monitoring und Automation dazu kommt und die Einteilung des Speichers in Prozent auf drei Speicherklassen. In der letzten Zeile der abgebildeten Tabelle werden die Gesamtkosten gemeinsam mit den Kosten des Datentransfers, Vorauszahlung und die puren monatlichen Speicherkosten der jeweiligen Speicherklassen addiert.

Google Cloud Storage

In der folgenden Tabelle werden die Kosten von Google Cloud Storage für drei Speicherklassen zusammengefasst:

	Standard	Nearline	Coldline
Monatliche Speicherkosten	63.75 EUR	36.03 EUR	16.63 EUR
Data Retrieval	—	0.45 EUR	0.90 EUR
Class A Operationen	1.21 EUR	2.42 EUR	4.84 EUR
Class B Operationen	0.19 EUR	0.48 EUR	4.84 EUR
Internet Egress	3.83 EUR		

Abbildung 2.3.2: Übersicht der einzelnen Kosten der Datenspeicherung in Google Cloud Storage

Die teuerste Speicherklasse mit 64.10 Euro ist die Standard Klasse. Die billigste mit 16.63 Euro ist die Coldline Klasse. Die Nearline Speicherklasse ist mit 36.03 Euro teurer als die Coldline. Bei der Standard Klasse gibt es keine Gebühren für die Daten Retrieval.

3 Prototypische Umsetzung

Im folgenden Kapitel wird der Prototyp genauer betrachtet und die verschiedenen Aspekte seiner Entwicklung und Implementierung werden erläutert. Es werden dabei die verwendeten Technologien und die Art der Speicherung und Bereitstellung von Binärdaten beleuchtet. Um die Performance zu bewerten, werden Messungen auf generierte Testdaten durchgeführt.

3.1 Überblick und Vorgehensweise

Zunächst wird auf die eingesetzten Technologien eingegangen, die bei der Entwicklung des Prototyps verwendet werden. Dies umfasst das Framework Spring Boot, Programmiersprachen und andere Tools wie Terraform, die zur Umsetzung des Prototyps genutzt werden. Ein besonderes Augenmerk wird auf die Speicherung der Binärdaten gelegt. Hier werden verschiedene Ansätze betrachtet, wie beispielsweise die Verwendung von AWS SDK und Google Client Libraries. Des Weiteren wird die Bereitstellung der Binärdaten behandelt. Hier wird die Methode des Signed URLs betrachtet, um die Daten effizient an die Anwender zu übertragen. Um die Leistung des Prototyps zu bewerten, werden Testdaten mit zufälligem Inhalt generiert. Dies ermöglicht eine realistische Simulation der Tickets und Rechnungen in leoticket und erlaubt eine Bewertung der Performance der Cloud Provider. Die Messungen werden auf einem virtuellen Server durchgeführt, um eine präzisere Analyse zu gewährleisten.

Insgesamt bietet das Kapitel einen umfassenden Überblick über den Prototypen, seine Technologien, die Speicherung und Bereitstellung von Binärdaten sowie die Performance-Messungen. Es dient als Grundlage für weitere Untersuchungen und die Optimierung des Prototyps.

3.2 Eingesetzte Technologien

Für die Umsetzung des Prototyps werden die folgenden Technologien eingesetzt:

- Spring Boot v3
- Terraform v1.4.4
- AWS SDK 2.0 Version
- Google Cloud Storage client library
- Java SDK 17 Temurin Version
- Maven v4.0.0
- AWS Toolkit
- aws cli
- gcloud cli

Für die Implementierung wurde als Entwicklungsumgebung Intellij Idea Ultimate verwendet. Intellij bietet das AWS Toolkit als Plugin an, das installiert werden kann. Als Framework wird die zum Zeitpunkt des erstellten Prototyps aktuellste Spring Boot 3 Version verwendet. Spring Boot stellt als Maven Dependency SDKs von beiden Cloud Anbietern zur Verfügung.

> Am 17. März 2021 wurde die neue Version des Spring Cloud AWS 2.3 veröffentlicht. Spring Cloud GCP und Spring Cloud AWS sind nicht mehr Teil des Spring Cloud Releases. Nicht Teil des Releases zu sein bedeutet auch, dass sie aus der Spring Cloud Organisation auf Github herausgenommen worden sind und dadurch neue Maven Package Namen haben. Das neue Package für Spring Cloud AWS heißt nun „io.awspring.cloud“, vgl. Gibb: Spring Cloud AWS 2.3 is now available, [6].

`spring-cloud-announce`

Die unten aufgeführten Maven Packages werden für AWS S3 und Cloud Storage verwendet:

```
<dependency>
        <groupId>com.google.cloud</groupId>
        <artifactId>spring-cloud-gcp-starter-storage</artifactId>
    </dependency>
```

```
<dependency>
        <groupId>io.awspring.cloud</groupId>
        <artifactId>spring-cloud-aws-s3</artifactId>
        <version>3.0.0</version>
    </dependency>
```

Als Spring Cloud GCP Version wird die 4.2.0 verwendet. Beide Spring Cloud Packages werden von der Community auf Github verwaltet und aktualisiert. Für die Erstellung des Spring Boot Projekts wurde der Spring Initializier von Spring selbst verwendet unter https://start.spring.io/. Terraform wird für die Erstellung der Buckets in S3 und Cloud Storage verwendet. Die verwendete

Terraform Version zum Zeitpunkt der Erstellung des Prototyp ist die 1.4.4. Zudem wird die Java SDK 17 Temurin Version verwendet. Für Maven wird die 4.0 Version verwendet.

Für die Authentifierung werden die aws cli und gloud cli verwendet. Diese werden über die offiziellen Dokumentationen installiert. Siehe https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html für AWS CLI und https://cloud.google.com/sdk/docs/install?hl=de für die gcloud CLI. Das AWS Toolkit für die Authentifizierung mit AWS angewendet. Um sich mit Google Cloud zu verbinden wird eine Methode des ADC verwendet, welches im nächsten Abschnitt genauer erläutert wird.

3.3 Speicherung von Binärdaten

Um Daten in S3 oder Cloud Storage speichern zu können, wurde der Prototyp so implementiert, dass der Nutzer sich zwischen S3 oder Cloud Storage entscheiden kann. Dies geschieht über die Klasse **CloudStorageServiceFactory**. Hier kann der Nutzer über die Environment Variable "cloud_provider"den gewünschten Provider mit „aws“oder "google cloudängeben. Dabei wird die Groß-, und Kleinschreibung nicht berücksichtigt. Die Environment Variablen können im System durch „export ¡EnvironmentVariable¿=¡value¿“exportiert werden. Wenn eine IDE wie Intellij verwendet wird, kann dies unter den Run-Einstellungen als „Environment Variables“eingefügt werden. Nach der Eingabe des Cloud Providers wird das Programm die entsprechende Klasse aufrufen. Für AWS ist das die Klasse **AWSS3StorageService** und für Google Cloud die Klasse **GoogleCloudStorageService**. Beide Klassen implementieren von dem Interface **CloudStorageService**. Die **CloudStorageService** definiert zwei Methoden und eine davon ist für die Speicherung der Daten zuständig, siehe unteren Code:

```
void uploadObject(String bucketName, String key, String file, String encryptionKey)
throws IOException;
```

Dieser Methode wird der Bucket Name, der Name des Objekts, der Pfad des Objekts und der Encryption Key übergeben. Der Encryption Key kann dabei der Schlüssel sein, der in AWS KMS oder Google Cloud KMS generiert wurde. Die Implementierung dieser Methode ist für beide Cloud Provider ähnlich gestaltet.

```
28    @Override
29    public void uploadObject(String bucketName, String key, String file, String encryptionKey) {
30        try {
31
32            PutObjectRequest putObjectRequest = PutObjectRequest.builder()
33                    .bucket(bucketName)
34                    .key(key)
35                    .serverSideEncryption(ServerSideEncryption.AWS_KMS)
36                    .ssekmsKeyId(encryptionKey)
37                    .storageClass(StorageClass.STANDARD_IA)
38                    .build();
39
40            Path filePath = Paths.get(file);
41            byte[] fileBytes = Files.readAllBytes(filePath);
42            RequestBody requestBody = RequestBody.fromBytes(fileBytes);
43
44            this.s3Client.putObject(putObjectRequest, requestBody);
45
46            System.out.println("File " + file + " uploaded to bucket " + bucketName + " as " + key);
47
48        } catch (S3Exception | IOException e) {
49            System.out.println(e.getMessage());
50        }
51    }
```

Abbildung 3.3.1: Prototyp - Hochladen eines Objekts nach S3

Der obere Code beschreibt den Speichervorgang eines Objekts nach AWS S3. Zuerst wird ein PUT Request Objekt erzeugt, indem die Parameter wie der Bucket Name, Objektname als Key und die Authentifizierungsmethoden angegeben werden. Es wird die AWS KMS Methode verwendet und den entsprechenden Schlüssel bereitgestellt. Zuletzt wird die Speicherklasse angegeben, indem das Objekt gespeichert wird. In diesem Beispiel wird die Standard - IA Klasse verwendet. Anschließend wird der Pfad der angegeben Datei gelesen, als Bytes Array umgewandelt und dem RequestBody übergeben. Dieser RequestBody gemeinsam mit dem PutObjectRequest wird an den S3Client übergeben und mit der AWS .putObject() Methode nach S3 hochgeladen. AWS S3 verschlüsselt das Objekt mit dem angegeben Encryption Key und ladet es nach S3 hoch.

Die folgenden Abbildung zeigt den Code Ausschnitt von der Klasse GoogleCloudStorageService. Ähnlich wie bei der Methode für AWS S3 wird auch hier ein Objekt nach Cloud Storage hochgeladen.

```
29  @Override
30  public void uploadObject(String bucketName, String key, String file, String encryptionKey) throws IOException {
31
32      Map<String, String> kmsKeyName = new HashMap<>();
33      kmsKeyName.put("kmsKeyName", encryptionKey);
34
35      // Get a reference to the bucket
36      BlobId blobId = BlobId.of(bucketName, key);
37      BlobInfo blobInfo = BlobInfo.newBuilder(blobId)
38              .setStorageClass(StorageClass.NEARLINE)
39              .setMetadata(kmsKeyName)
40              .build();
41
42      // set a generation-match precondition to avoid potential race
43      // conditions and data corruptions. The request returns a 412 error if the
44      // preconditions are not met.
45      Storage.BlobWriteOption precondition;
46      if (this.storageClient.get(bucketName, key) == null) {
47          // For a target object that does not yet exist, set the DoesNotExist precondition.
48          // This will cause the request to fail if the object is created before the request runs.
49          precondition = Storage.BlobWriteOption.doesNotExist();
50      } else {
51          // If the destination already exists in your bucket, instead set a generation-match
52          // precondition. This will cause the request to fail if the existing object's generation
53          // changes before the request runs.
54          precondition =
55                  Storage.BlobWriteOption.generationMatch(
56                          this.storageClient.get(bucketName, key).getGeneration());
57      }
58      this.storageClient.createFrom(blobInfo, Paths.get(file), precondition);
59
60      System.out.println("File " + file + " uploaded to bucket " + bucketName + " as " + key);
61  }
```

Abbildung 3.3.2: Prototyp - Hochladen des Objekt nach Cloud Storage

Dabei werden ähnliche Parameter der Methode wie in AWS S3 übergeben. Um ein Objekt in ein Bucket hochladen zu können, wird eine Referenz zum Bucket erstellt. Dies geschieht durch die BlobId, der den Bucket namen und den Namen des Objekts beinhaltet. Anschließend wird diese b†lobId dem BlobInfo Objekt übergeben und die Speicherklasse NEARLINE definiert. Anschließend wird über die Metadaten die KMS Encryption Key angegeben. Nach dem überprüft worden ist, ob das Objekt bereits im Bucket existiert oder nicht, wird das Objekt in der Zeile 58 hochgeladen.

Um das Programm zum Laufen zu bringen, wird die Hauptklasse **HandsonAwsGcApplication** ausgeführt. Damit das Programm erfolgreich läuft, müssen die Environment Variablen im System exportiert werden. Alle Environment Variablen sind in der „application.properties“hinterlegt. Diese werden beim Start des Programms gelesen und angewendet. Außerdem müssen die AWS und Google Cloud Credentials hinterlegt werden. Dies kann entweder über die AWS Toolkit Plugin gesteuert werden oder mit dem Befehl „aws configure“in der Kommandozeile. Für Google Cloud Credentials kann mit dem Befehl „export GOOGLE_APPLICATION_CREDENTIALS=¡service-account-json-file¿ der Service Account hinterlegt werden oder durch Ausführen des Befehls ”gcloud auth application-default login“in der Kommandozeile, was die Credentials lokal speichert und für ADC verwendet wird.

3.4 Bereitstellung der Binärdaten

Bei der Bereitstellung der Binärdaten geht es dabei um die signed URL Methode, um in leoticket Dateien über Links an den Kunden bereitzustellen. Es werden Code Implementierungen von beiden Cloud Providern vorgestellt und erläutert.

Die untere Abbildung zeigt die Methode der Klasse **AWSS3StorageService**:

```
53  @Override
54  public void getPresignedUrl(String bucket, String key, Integer minutes, String encryptionKey) {
55      try {
56
57          GetObjectRequest getObjectRequest = GetObjectRequest.builder()
58                  .bucket(bucket)
59                  .key(key)
60                  .build();
61
62          GetObjectPresignRequest getObjectPresignRequest = GetObjectPresignRequest.builder()
63                  .signatureDuration(Duration.ofMinutes(minutes))
64                  .getObjectRequest(getObjectRequest)
65                  .build();
66
67          PresignedGetObjectRequest presignedGetObjectRequest = presigner.presignGetObject(getObjectPresignRequest);
68
69          String url = presignedGetObjectRequest.url().toString();
70
71          System.out.println("Presigned URL: " + url);
72
73      } catch (S3Exception e) {
74          System.out.println(e.getMessage());
75      }
76  }
```

Abbildung 3.4.1: Prototyp - AWS getPresignedUrl() Methode

Die Methode bekommt ähnlich wie beim Upload des Objekts erstmal die beiden Parameter Named des Bucket und Name des Objekts. Anschließend wird eine Zahl als Minuten übergeben, die dafür sorgt, dass der Link nur für diese bestimmte Zeit valide ist. Auch hier muss man den gleichen Encryption Key wie beim Upload für das Entschlüsseln der Daten mitgeben. In Zeile 57 wir das GetObjectRequest Objekt erstellt und den Bucket Namen und Objektnamen mitgegeben. Anschließend wird dieses Objekt dem GetObjectPresignRequest Objekt übergeben und dabei die Signaturdauer angegeben. Die Signaturdauer stellt die signierte URL solange wie die angegeben Minute zur Verfügung. Um nun die signierte URL zu erzeugen, wird die GetObjectPresignRequest dem S3Presigner übergeben, die presignGetObject() Methode darauf aufgerufen und dem PresignedGetObjectReuqest übergeben. In Zeile 69 wird die generierte URL als String gespeichert und anschließend ausgegeben.

Bei der Methode von Cloud Storage ist der Vorgang zur Erstellung des signierten URLs kürzer. Folgende Abbildung zeigt die Methode zur Generierung des signierten URLs für Cloud Storage:

```
63  @Override
64  public void getPresignedUrl(String bucketName, String key, Integer minutes, String encryptionKey) {
65
66      Map<String, String> kmsKeyName = new HashMap<>();
67      kmsKeyName.put("kmsKeyName", encryptionKey);
68
69      // Define resource
70      BlobInfo blobInfo = BlobInfo.newBuilder(BlobId.of(bucketName, key))
71              .setMetadata(kmsKeyName)
72              .build();
73
74      URL url =
75              this.storageClient.signUrl(blobInfo, minutes, TimeUnit.MINUTES, Storage.SignUrlOption.withV4Signature());
76
77      System.out.println("Generated GET signed URL:");
78      System.out.println(url);
79  }
```

Abbildung 3.4.2: Prototyp - GC getPresignedUrl() Methode

Ähnlich wie bei S3 wird eine Referenz zum Bucket erstellt, indem man den Bucket Namen und den Namen des Objekts dem BlobInfo Objekt mitgibt. Anschließend kann die URL durch Aufrufen der Methode in Zeile 63 erstellt werden. Hier wird das BlobInfo Objekt, die Minuten in und die Signatur Methode übergeben. Zuletzt wird die URL in der Kommandozeile ausgegeben.

Mit signierten URLs können die Anforderungen von leoticket an die sichere Speicherung und Bereitstellung der Daten über signierte URLs berücksichtigt werden. Dies bedeutet, Kunden können über diese Links die Dateien für Tickets und Rechnungen herunterladen ohne das Problem von zu großen Email-Anhängen zu haben.

3.5 Messung der Performance

In diesem Abschnitt erfolgt die Messung der Performance für die Dienste von AWS und Google Cloud. Dabei werden bis zu 1000 generierte Dateien sowohl für den Upload als auch für den Download betrachtet.Die Performance Analyse wird auf einer virtuellen Maschine ausgeführt, um eine realistische Messung zu gewährleisten. Die Performancemessung beim Hoch- und Herunterladen kann jedoch von verschiedenen Faktoren wie Netzwerklatenz, verfügbarer Bandbreite, Serverkapazität, Datenmenge und der Auslastung der Server beeinflusst werden. Die Messungen dienen lediglich des groben Vergleichs beider Cloud Provider. Die Performance-Messung wird auf verschiedene Speicherklassen durchgeführt, um einen Vergleich zwischen dieser zu ermöglichen und eine Grundlage für die Bewertung ihrer Leistungsfähigkeit zu schaffen.

AWS

AWS bietet Dienste für die Performance Analyse. Unter anderem die Amazon CloudWatch, S3 Storage Lens und die S3 Transfer Acceleration. Die AWS CLI stellt einfache Methoden zum S3 Upload und Download Tests vor. Beispielsweise kann man mit dem Befehl:

```
aws s3 cp <lokaler_pfad> s3://<Bucket_Name>/<Ziel_Dateipfad>
```

Dateien hoch,-und herunterladen und die Zeit für die benötigte Request verwenden. Auch mit der AWS SDK können Performance Test Skripte geschrieben werden. Diese Methode wird für den Prototypen angewendet. Dabei werden Tests bereitgestellt, die mehrere Dateien automatisch hoch, und herunterlädt und dabei die Zeit misst, die vergangen ist.

Google Cloud Storage

GC bietet einen eigenen "gsutil"Tool für die Performance Analyse. Im Abschnitt Performance bereits erwähnt können durch die perfdiag Funktion Performance Diagnosen erstellt werden.

> Mehrere Testdateien werden aus einem angegebenen Bucket hoch-und heruntergeladen. Nach der Analyse werden alle Testdateien wieder gelöscht nach erfolgreicher Diagnose. Die gsutil Performance kann von einigen Fakoren beeinflusst werden wie vom Client, Server oder Netzwerk. Möglich sind die CPU Geschwindigkeit, der verfügbare Speicher, die Netzwerk Bandbreite, Firewalls und Fehlerraten zwischen gsutil und den Google Servern. Die perfiag Funktion wurde dafür bereitgestellt, damit Nutzer Messungen durchführen können, die beim Troubleshooting von Performance Problemen helfen, vgl. Storage: perfdiag - Run performance diagnostic, [18].

gc-perfdiag

Um die Performance Diagnose auszuführen, kann der folgende Befehl verwendet werden:

```
gsutil perfdiag -o test.json -n 67000 -s 400kb gs://leoticket-bucket
```

Die -o Option schreibt den Output des Ergebnisses in eine Datei. Die Output Datei ist eine JSON Datei mit System Informationen und enthält die Performance Diagnose Ergebnisse. Die -n Option setzt die Anzahl der Objekte, die heruntergeladen und hochgeladen werden sollen während dem Test. Mit der -s Option kann man die Objektgröße in bytes angeben. Zum Schluss wird der Name des Buckets angegeben. Damit der Befehl erfolgreich ausgeführt werden kann, braucht man die entsprechenden Rechte und muss sich authentifizieren können.

Um die Performance der SDKs zu analysieren, werden Objekte mit jeweils 100kb Objektgröße in verschiedenen Speicherklassen hoch-, und heruntergeladen. Anschließend wird die Performance über die SDK von Cloud Storage ähnlich wie bei AWS getestet. Der Test wird auf einer virtuellen Maschine ausgeführt.

Die Performance wird schrittweise gemessen. Angefangen von einer Datei bis hin zu 1000 Dateien in 10-fach Schritten. Das bedeutet, es werden die Messsungen für eine Datei, für zehn Dateien, für 100 Dateien und für 1000 Dateien untersucht. Um die Messung durchzuführen, wird eine Java Jar Datei des Prototyps erstellt. Dabei werden die Testmethoden ausgeführt. Die Testmethoden generieren zuerst die Testdaten gefüllt mit zufälligen String Werten in der Größe von 100kb. Anschließend werden die Testmethoden in der Kommandozeile ausgeführt.

3.5.1 Messungsergebnisse

In diesem Abschnitt werden die Messungsergebnisse der Performance Messung in genauen Zahlen in Millisekunden bereitgestellt. Die Speicherklassen Standard, Standard-IA und One Zone-IA von AWS wurden jeweils mit Standard, Nearline und Coldline von GC gemessen und verglichen. Im folgenden werden die Ergebnisse der Upload und Download Geschwindigkeit aller Speicherklassen präsentiert: Test

```
STANDARD-IA vs. NEARLINE

1 File-------------------------------------------------------------------------

Elapsed Time for 1  Object Upload in S3:-------------------------------705ms.
Elapsed Time for 1 Object Upload in Cloud Storage:---------------------841ms.

Elapsed Time for 1 Object Download in S3:------------------------------18ms.
Elapsed Time for 1 Object Download in Cloud Storage:-------------------26ms.

10 Files-----------------------------------------------------------------------

Elapsed Time for 10  Object Uploads in S3:-----------------------------1862ms.
Elapsed Time for 10 Object Uploads in Cloud Storage:-------------------3217ms.

Elapsed Time for 10 Object Downloads in S3:----------------------------35ms.
Elapsed Time for 10 Object Downloads in Cloud Storage:-----------------94ms.

100 Files----------------------------------------------------------------------

Elapsed Time for 100  Object Uploads in S3:----------------------------16681ms.
Elapsed Time for 100 Object Uploads in Cloud Storage:------------------22683ms.

Elapsed Time for 100 Object Downloads in S3:---------------------------129ms.
Elapsed Time for 100 Object Downloads in Cloud Storage:----------------269ms.

1000 Files---------------------------------------------------------------------

Elapsed Time for 1000  Object Uploads in S3:---------------------------71175ms.
Elapsed Time for 1000 Object Uploads in Cloud Storage:-----------------1851348ms.

Elapsed Time for 1000 Object Downloads in S3:--------------------------663ms.
Elapsed Time for 1000 Object Downloads in Cloud Storage:---------------1489ms.
```

```
STANDARD vs. STANDARD

1 File--------------------------------------------------------------------------

Elapsed Time for 1  Object Upload in S3:--------------------------------585ms.
Elapsed Time for 1 Object Upload in Cloud Storage:----------------------660ms.

Elapsed Time for 1 Object Download in S3:-------------------------------28ms.
Elapsed Time for 1 Object Download in Cloud Storage:--------------------19ms.

10 Files------------------------------------------------------------------------

Elapsed Time for 10  Object Uploads in S3:------------------------------1369ms.
Elapsed Time for 10 Object Uploads in Cloud Storage:--------------------3002ms.

Elapsed Time for 10 Object Downloads in S3:-----------------------------70ms.
Elapsed Time for 10 Object Downloads in Cloud Storage:------------------108ms.

100 Files-----------------------------------------------------------------------

Elapsed Time for 100  Object Uploads in S3:-----------------------------8110ms.
Elapsed Time for 100 Object Uploads in Cloud Storage:------------------20750ms.

Elapsed Time for 100 Object Downloads in S3:----------------------------180ms.
Elapsed Time for 100 Object Downloads in Cloud Storage:-----------------396ms.

1000 Files----------------------------------------------------------------------

Elapsed Time for 1000  Object Uploads in S3:---------------------------73992ms.
Elapsed Time for 1000 Object Uploads in Cloud Storage:-----------------176004ms.

Elapsed Time for 1000 Object Downloads in S3:---------------------------650ms.
Elapsed Time for 1000 Object Downloads in Cloud Storage:----------------1619ms.
```

```
ONEZONE-IA vs. COLDLINE

1 File--------------------------------------------------------------------------

Elapsed Time for 1  Object Upload in S3:--------------------------------505ms.
Elapsed Time for 1 Object Upload in Cloud Storage:----------------------636ms.

Elapsed Time for 1 Object Download in S3:-------------------------------28ms.
Elapsed Time for 1 Object Download in Cloud Storage:--------------------18ms.

10 Files------------------------------------------------------------------------

Elapsed Time for 10  Object Uploads in S3:-----------------------------1105ms.
Elapsed Time for 10 Object Uploads in Cloud Storage:--------------------2855ms.

Elapsed Time for 10 Object Downloads in S3:-----------------------------65ms.
Elapsed Time for 10 Object Downloads in Cloud Storage:------------------89ms.

100 Files-----------------------------------------------------------------------

Elapsed Time for 100  Object Uploads in S3:-----------------------------7539ms.
Elapsed Time for 100 Object Uploads in Cloud Storage:--------------------20391ms.

Elapsed Time for 100 Object Downloads in S3:----------------------------187ms.
Elapsed Time for 100 Object Downloads in Cloud Storage:-----------------250ms.

1000 Files----------------------------------------------------------------------

Elapsed Time for 1000  Object Uploads in S3:----------------------------73086ms.
Elapsed Time for 1000 Object Uploads in Cloud Storage:------------------163661ms.

Elapsed Time for 1000 Object Downloads in S3:---------------------------701ms.
Elapsed Time for 1000 Object Downloads in Cloud Storage:----------------1559ms.
```

Die Ergebnisse werden im Kapitel Zusammenfassung der Ergebnisse zusammengefasst und als Liniendiagramm dargestellt.

3.6 Zusammenfassung der Implementierung

Der Prototyp soll einen Vergleich zwischen den beiden Cloud Providern bieten. Das Ziel dabei ist es, ähnliche Technologien von beiden Providern zu verwenden und die Performance mit Testdaten dabei messen zu können. Es dient für zwei Funktionen. Das Hochladen und Herunterladen von Dateien. Dabei werden die Anforderungen an leoticket mitberücksichtigt. Das bedeutet, das signierte URLs für die Bereitstellung der Dateien verwendet werden. Außerdem wird für die Datenverschlüsselung die SSE KMS von beiden Providern angewendet. Außerdem soll der Prototyp im weiteren Verlauf als externe Library für eigene Anwendungen integriert werden. Für die Implementierung wurden die offiziellen Dokumentationen von AWS und GC verwendet, um die aktuellsten Versionen zum Zeitpunkt der Arbeit bereitzustellen. Des weiteren wurde auf Wunsch von Leomedia Spring Boot als Framework verwendet, um eine Enterprise Anwendung bereitstellen zu können. Terraform wird dabei für die Erstellung von Buckets erstellt und stellt auch ein Vergleich für die Integration der beiden Providern nach Terraform dar. Aus Preferenzgründen wird Java als Programmiersprache gewählt, jedoch stellen beide Provider viele weitere Sprachen, die bereits im Kapitel API Anbindung erwähnt wurden, zur Verfügung.

Der Prototyp wurde so aufgebaut, dass man zwischen AWS und GC durch eine Environment Variable wählen kann. Die Klasse **CloudStorageServiceFactory** stellt die Auswahl zwischen AWS und GC zur Verfügung. Je nachdem welchen Cloud Provider der Nutzer angegeben hat, wird die entsprechende Klasse instanziiert und aufgerufen. Die Klassen **AWSS3StorageService** und **GoogleCloudStorageService** implementieren die Methoden des Interfaces **CloudStorageService**.

Die bereitgestellten Tests dienen der Performance Messung. Dabei werden Dateien automatisch als 400kb große Objekte erstellt und durch die Testmethoden verwendet, um Objekte hoch-, und herunterzuladen. Sie unterstützen dabei, die Geschwindigkeit der verschiedenen Funktionen zu messen.

Da die Maven Dependencies kein Teil der Spring Cloud Release Train sind, hängt es von der Commmunity ab, die Versionen aktuell zu halten. Für AWS bedeutet dies, dass die AWS SDK 2. Version nicht komplett abgedeckt ist. Jedoch ist sie so weit, dass S3 bereits unterstützt wird.

4 Ergebnisse dieser Arbeit

In diesem Kapitel werden die Funktionalitäten des Prototyps beschrieben. Zum Schluss werden die Kalkulationsergebnisse der Kostenanalyse als Tabelle und die Messungsergebnisse der Performance Messung als Liniendiagramm bereitgestellt und auf einzelne Ergebnisse eingegangen. Außerdem dient dieses Kapitel zur schnellen Information über die Ergebnisse der Performance und Kosten.

4.1 Beschreibung und Funktionalität des Prototyps

Der Prototyp hat die Funktionalität des Uploads und Downloads von Objekten nach S3 oder Cloud Storage. Der Nutzer kann sich zwischen AWS und GC entscheiden, indem die cloud_provider Environment Variable vor dem Ausführen des Programms gesetzt wird. Damit das Programm erfolgreich durchlaufen kann, müssen alle Environment Variablen in der application.properties Datei gesetzt werden. Außerdem wird eine Authentifizierung für GC und AWS verlangt, sonst schlägt das Programm fehl. Die Authentifizierung für GC kann über die ADC erfolgen. In der offiziellen Dokumentation werden die verschiedenen Methoden aufgelistet. Für den Prototypen wurde die lokale Entwicklungsumgebung für die Authentifizierung durch

```
export GOOGLE\_APPLICATION\_CREDENTIALS=jsonKeyPath
```

oder durch das gcloud cli Befehl

```
gcloud auth application-default login
```

verwendet. Dabei ist der jsonKeyPath der Pfad zum erstellten Service Account, die Rechte für das Uploaden und Downloaden besitzt. Siehe auch: Set Up ADC Credentials.

Für die Authentifizierung mit AWS benötigt es lediglich das AWS Toolkit Plugin. Diese Toolkit stellt AWS für einige IDEs zur Verfügung. Unter anderem für Visual Studio, Eclipse und JetBrains. Falls das AWS Toolkit nicht zur Verfügung steht, dann kann die Authentifizierung auch über die aws cli erfolgen. Durch aws configure können die Credentials gesetzt werden. Siehe auch: Set Up AWS Credentials. Nachdem die Authentifizierungseinstellungen und die Environment Variablen gesetzt wurden, kann das Programm gestartet werden. Damit die Dateien in die richtige Speicherklasse hochgeladen werden können, muss für GC der richtige Bucket Name angegeben werden. In AWS muss jedoch die gewünschte Speicherklasse angegeben werden, damit das Objekt in die richtige Speicherklasse geschrieben wird. Dieser Unterschied besteht, da in GC die Speicherklasse für ein Bucket ausgewählt werden kann. In AWS ist dies jedoch nicht möglich, deshalb haben alle Buckets in AWS bei Erstellung die Standard Speicherklasse eingestellt. Das Objekt wird durch die SSE KMS verschlüsselt hochgeladen. Die Verschlüsselung wird automatisch von beiden Cloud Providern durchgeführt. Dabei muss der entsprechende KMS Schlüssel des ausgewählten Cloud Providers der Environment Variable sse_kms_key_id_arn übergeben werden. Beim Herunterladen des Objekts muss dieser Schlüssel erneut mitgegeben werden, damit das Objekt entschlüsselt werden kann. Die Hauptklasse **HandsonAwsGcApplication** ruft die entsprechende Klasse für AWS oder

GC auf und initialisiert diese. Vorher müssen die Parameter für die .uploadObject(bucketName, key, filePath, encryptionKey, storageClass) und .getPresignedUrl(bucketName, key, minutes, encryptionKey) übergeben werden. Die Minuten stellen die Zeit ein, in der die generierte signierte URL gültig ist. Der storageClass Parameter gilt nur für die AWS Funktion und kann die Werte STANDARD, STANDARD_IA und ONEZONE_IA annehmen.

Über Terraform werden die Buckets erstellt und die Konfigurationen eingestellt. Diese Konfigurationen sorgen für die Einhaltung der Anforderungen an leoticket. Die Buckets müssen vor dem Programmstart bereits existieren. Die Terraform Dateien werden im entsprechenden Ordner bereitgestellt. Diese werden vor dem Programmstart als erstes ausgeführt, um die benötigten Ressourcen zu erstellen.

Der Prototyp dient außerdem dazu die APIs miteinander zu vergleichen und jeweils die ähnlichen Funktionalitäten für beide Provider bereitzustellen.

4.2 Zusammenfassung der Ergebnisse

In diesem Abschnitt werden die Kosten der verschiedenen Speicherklassen nochmal aufgegriffen und die Gesamtkosten als Tabelle dargestellt. Auch werden die Messungsergebnisse der Performance Analyse als Liniendiagramm bereitgestellt und die Zahlen der Kosten und Messungen untersucht.

4.2.1 Kalkulationsergebnisse

Im folgenden werden die Gesamtkosten der Speicherklasse von beiden Cloud Providern dargestellt und untersucht. Zuerst werden in der unteren Abbildung die Kosten von AWS untersucht:

	Standard	Intelligent Tiering	Standard-IA	One Zone - IA
Monatliche Kosten der Speicherklasse	70.27 EUR	70.27 EUR	38.72 EUR	30.98 EUR
PUT requests	1.35 EUR	1.35 EUR	2.50 EUR	2.50 EUR
GET requests	0.22 EUR	0.22 EUR	0.50 EUR	0.50 EUR
Data Retrieval	—	—	0.47 EUR	0.47 EUR
Data Transfer	4.20 EUR			
Monitoring und Automation objects	—	75.19 EUR	—	—
Gesamtkosten pro Monat	**76.04 EUR**	**151.23 EUR**	**46.39 EUR**	**38.65 EUR**
Vorauszahlung	162.42 EUR	162.42 EUR	300.76 EUR	300.76 EUR
Gesamtkosten im ersten Monat	**238.46 EUR**	**313.65 EUR**	**347.15 EUR**	**339.41 EUR**

Abbildung 4.2.1: Zusammenfassung der Gesamtkosten für AWS S3 pro Speicherklasse

In dieser Abbildung werden in der letzten Zeile der Tabelle die Gesamtkosten der Speicherklassen pro Monat veranschaulicht. Die Gesamtkosten im ersten Monat der Speicherklasse Standard beträgt 123.67 USD. In diesen Kosten stecken die Vorauszahlungskosten, die PUT und GET Request Gebühren und die Datenübertragungsgebühren. Im Intelligent Tiering und Standard sind die Datenabrufgebühren nicht enthalten, da es dafür keine Gebühren abgezogen werden. Die Gesamtkosten des Intelligent Tiering pro Monat beträgt 120.15 USD. Die Kosten des Intelligent Tiering hängen jedoch von der Speicherverteilung ab. Die Kosten sind grobe Werte und können von der Verteilung des Speichers in verschiedene Speicherklassen in Prozent abhängen. Bei der Standard-IA Speicherklasse sind es mit 127.81 USD Kosten die teuersten Kosten im Vergleich zu den anderen Speicherklassen. Die One Zone-IA ist mit 119.51 USD die billigste Speicherklasse.

In Google Cloud gibt es keine Vorauszahlungen im Vergleich zu AWS. Hier unterscheiden sich die Gesamtkosten der verschiedenen Speicherklassen zu AWS. In der folgenden Abbildung werden die Gesamtkosten zusammengefasst:

	Standard	Nearline	Coldline
Monatliche Speicherkosten	63.75 EUR	36.03 EUR	16.63 EUR
Data Retrieval	—	0.45 EUR	0.90 EUR
Class A Operationen	1.21 EUR	2.42 EUR	4.84 EUR
Class B Operationen	0.19 EUR	0.48 EUR	4.84 EUR
Internet Egress	3.83 EUR		
Gesamtkosten	68.98 EUR	43.21 EUR	31.04 EUR

Abbildung 4.2.2: Zusammenfassung der Gesamtkosten für GC Storage pro Speicherklasse

Die Gesamtkosten bilden sich hauptsächlich aus den Datenabrufen, Class A und Class B Operationen, was die PUT und GET beinhalten und die Internet Egress, ausgehend von Google Cloud zum Internet, beispielsweise beim Abrufen von Dateien. Wobei keine Gebühren für die Datenabrufe der Standard Speicherklasse anfallen. Die Gesamtkosten des Standards beträgt 69.86 Euro, der Nearline 42.61 Euro und des Coldline 25.36 Euro.

4.2.2 Messungsergebnisse

In diesem Abschnitt werden die Messungsergebnisse der Performance Analyse bereitgestellt. Die Speicherklassen Standard, Standard-IA und One Zone-IA von AWS wurden jeweils mit Standard, Nearline und Coldline von GC verglichen. Im folgenden werden die Ergebnisse der Upload Geschwindigkeit aller Speicherklassen als Liniendiagramm dargestellt:

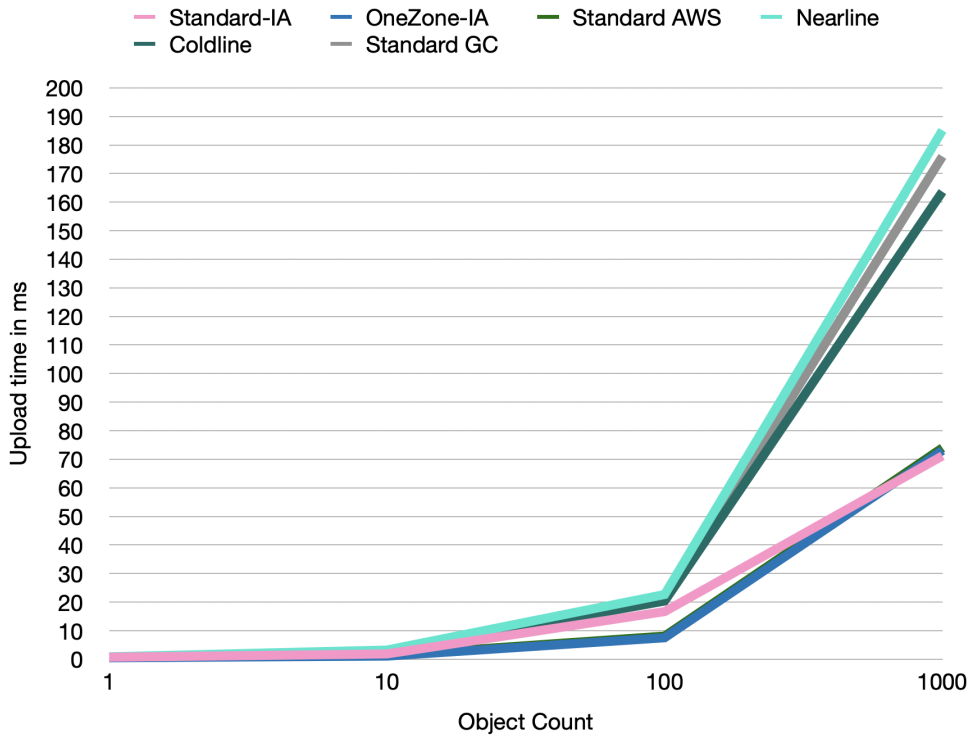

Abbildung 4.2.3: Liniendiagramm - Upload Zeit der verschiedenen Speicherklassen

In der obigen Abbildung wird die Upload Geschwindigkeit in Millisekunden auf der Y-Achse beschrieben. Die X-Achse beschreibt die Objektanzahl, die für jede Speicherklasse hochgeladen wurde. Die Farben unterscheiden die verschiedenen Speicherklassen voneinander, wobei die Speicherklassen von AWS jeweils nebeneinander platziert sind, genauso wie für die Speicherklassen von GC. Die genauen Zahlen vom Abschnitt 3.5.1 wurden in diesen Diagramm eingefügt, um die Unterschiede zwischen den Speicherklassen beider Provider zu visualisieren. So liegen die Speicherklassen von AWS im Graph nah beieinander, ähnlich wie mit den Speicherklassen von GC. Beim Hochladen von einer Datei bis hin zu zehn Dateien gibt es minimale Unterschiede und sind sehr nah beieinander. Diese Werte liegen unter 3033 Millisekunden, was ungefähr 3 Sekunden entspricht. Die Linien gehen ab dem Hochladen von 11 bis 1000 Dateien stärker auseinander. Die drei Speicherklassen von GC entfernen sich von den drei Speicherklassen von AWS. Ab 100 Dateien gibt es erneut einen Knick und die Speicherklassen von GC entfernen sich weiter von den AWS Speicherklassen. Die GC Speicherklassen wachsen steiler nach oben als bei AWS und bewegen sich in höheren Millisekunden Bereichen von ungefähr 20 000 Millisekunden bis hin zu 190 000 Millisekunden, umgerechnet bei ungefähr 20 bis 190 Sekunden. AWS bewegt sich dabei maximal bis 140 000 Millisekunden, umgerechnet bis 140 Sekunden.

Bei der Download Geschwindigkeit unterscheiden sich die Werte vom Upload stärker, denn die Zahlen für den Download bewegen sich in Bereichen von 16 bis hin zu 1556 Millisekunden. Hier steigen die Werte nicht höher als 2 Sekunden. In der folgenden Abbildung wird dies veranschaulicht:

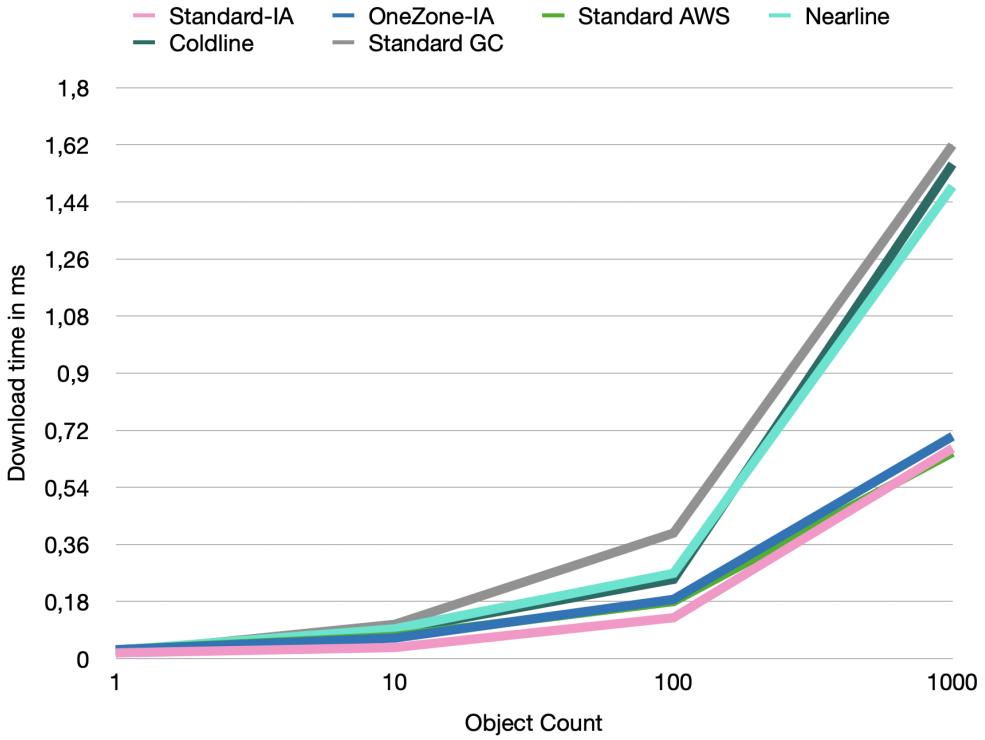

Abbildung 4.2.4: Lininediagramm - Download Zeit der verschiedenen Speicherklassen

Dieser Graph ähnelt dem Graph für den Upload in Abbildung 4.2.3. Denn auch hier entfernen sich die Linien für die GC Speicherklassen von den Linien der AWS Speicherklassen. Ab Downloads von zehn Dateien trennen sich die Cloud Provider voneinander. Die Coldline Speicherklasse von GC hat dabei am längsten mit 1556 Millisekunden für den Download von 1000 Dateien gebraucht. Die schnellste Speicherklasse von AWS ist die OneZone-IA mit 569 Millisekunden beim Download von 1000 Dateien. Wobei die Speicherklassen von AWS nah beieinander liegen und weniger als 1 Sekunde für den Download gebraucht haben. Bei GC dauern die Downloads von 1000 Dateien mindestens 1.5 Sekunden. Im Bereich von einer bis zehn Dateien bewegen sich die Speicherklassen in unter 160 Millisekunden Bereichen. Jedoch gibt es bei zehn Dateien bereits erste minimale Unterschiede die langsam wachsen.

5 Diskussion

In diesem Kapitel werden die Eigenschaften, der Prototyp, Kalkulations-, und die Messungsergebnisse der Performance analysiert und interpretiert. Dabei werden zuerst die Kalkulationsergebnisse und Messungsergebnisse untersucht. Anschließend werden die Eigenschaften und der Prototyp bewertet und dabei auf die Grenzen des Prototyps eingegangen.

5.1 Analyse und Interpretation der Ergebnisse

Aus der Kostenanalyse und den Kalkulationsergebnissen hat sich ergeben, dass Google Cloud insgesamt niedrigere Kosten als AWS bietet.

5.2 Bewertung des Prototyps

Stärken und Schwächen

6 Fazit

6.1 Beantwortung der Forschungsfrage

6.2 Potenzielle Anwendung des Prototyps

7 Danksagung

8 Literaturverzeichnis

Internetquellen

ws-availability [1] AWS: Data Protection in Amazon S3. Abgerufen am 14.05.2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/DataDurability.html.

aws-iam-s3 [2] AWS: Security: Identity and Access Management. Abgerufen am 11.05.2023. URL: https://docs.aws.amazon.com/de_de/AmazonS3/latest/userguide//access-control-overview.html.

aws-sse-c [3] AWS: Using server-side encryption with customer-provided keys (SSE-C). Abgerufen am 12.05.2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/ServerSideEncryptionCustomerKey
html.

gcp-sla [4] Google Cloud: Cloud Storage Service Level Agreement (SLA). Abgerufen am 17.05.2023. URL: https://cloud.google.com/storage/sla.

gcp-autoscale [5] Google Cloud: Request rate and access distribution guidelines: Auto Scaling. Abgerufen am 17.05.2023. URL: https://cloud.google.com/storage/docs/request-rate.

-cloud-announce [6] Spencer Gibb: Spring Cloud AWS 2.3 is now available. Abgerufen am 24.05.2023. URL: https://spring.io/blog/2021/03/17/spring-cloud-aws-2-3-is-now-available#why-has-the-package-name-changed.

ibm-topics [7] IBM: What is block storage? Abgerufen am 07.05.2023. URL: https://www.ibm.com/topics/block-storage.

ibm-storage [8] IBM: What is object storage? Abgerufen am 07.05.2023. URL: https://www.ibm.com/topics/object-storage.

dataCore-OS [9] 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 [10] GmbH Leomedia: Bachelor- und Master-Themen. URL: https://www.leomedia.de/unternehmen/abschlussarbeiten/.

gcp-blog [11] Colt McAnlis: Optimizing your Cloud Storage performance: Google Cloud Performance Atlas. Abgerufen am 18.05.2023. URL: https://cloud.google.com/blog/products/gcp/optimizing-your-cloud-storage-performance-google-cloud-performance-atlas/.

redHat-storage [12] 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.

aws-api [13] Amazon S3: Amazon S3 REST API Introduction. Abgerufen am 18.05.2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html.

aws-sdk [14] Amazon S3: Developing with Amazon S3 using the AWS SDKs, and explorers: Using this service with an AWS SDK. Abgerufen am 18.05.2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html.

rformance-guide [15] Amazon S3: Performance Guidelines for Amazon S3. Abgerufen am 18.05.2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/optimizing-performance-guidelines.html.

aws-signed-urls [16] Amazon S3: Using presigned URLs. Abgerufen am 19.05.2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-presigned-url.html.

gcp-encrypt [17] Google Cloud Storage: Google-managed encryption keys. Abgerufen am 13.05.2023. URL: https://cloud.google.com/storage/docs/encryption/default-keys.

gc-perfdiag [18] Google Cloud Storage: perfdiag - Run performance diagnostic. Abgerufen am 21.05.2023. URL: https://cloud.google.com/storage/docs/gsutil/commands/perfdiag.

nx-fileScala [19] 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

Github Link: https://github.com/gmzbae/handson-cloud-storage.git

9.1.2 Dokumentation

9.1.3 Code Snippets