

Aufbau einer Datenspeicher-Cloud

Motivation

Microsoft Azure Storage

■ Weltumspannendes System zur Speicherung von Daten

■ Heterogenes Nutzungsverhalten

- Eigene Dienste des Cloud-Betreibers vs. Anwendungen unabhängiger Nutzer
- Nutzung als Zwischenspeicher vs. Langzeitspeicherung von Daten
- Verwaltung strukturierter vs. unstrukturierter Daten

■ Ort der Datenspeicherung

- Global: Latenzüberlegungen, rechtliche Bestimmungen,...
- Lokal: Art der Anbindung an die Rechen-Cloud desselben Anbieters

■ Großes Spektrum an möglichen Fehlersituationen

- Defekte einzelner Rechnerkomponenten (z. B. Festplatten)
- ⋮
- Ausfall ganzer Datenzentren

■ Herausforderungen

- Wie feingranular bestimmt ein Nutzer den Speicherort seiner Daten?
- Wie tiefgreifend sollen die Maßnahmen zum **Schutz vor Datenverlust** sein?

Microsoft Azure Storage

■ Anforderungen [Werden sie erfüllt?]

- Starke Konsistenz
- Globaler Namensraum
- Kein Datenverlust bei Katastrophen
- Niedrige Kosten

■ Microsoft Azure Storage (früher: Windows Azure Storage)

- Einheitliches Speichersystem für unterschiedliche Nutzdaten
- Trennung des Datenspeichers vom Rest der Cloud
- Rückgriff auf das Domain Name System (DNS)
- Georeplikation über mehrere Datenzentren

■ Literatur

 Brad Calder, Ju Wang, Aaron Ogus, Niranjana Nilakantan, Arild Skjolsvold, Sam McKelvie et al. **Windows Azure Storage: A highly available cloud storage service with strong consistency** *Proceedings of the 23rd Symposium on Operating Systems Principles (SOSP '11)*, S. 143–157, 2011.

Adressierung von Datenobjekten

■ Verfügbare Datenobjekte

- *Blobs* [Binary large objects]
- Tabellen
- Warteschlangen

■ Typischer Einsatz von Objekten

- Eingabedaten: Blobs
- Zwischenergebnisse und Ausgabedaten: Blobs oder Tabellen
- Koordinierung: Warteschlangen

■ Globaler partitionierter Namensraum

[Protokoll]://[Konto].[Dienst].core.windows.net/[Partition]/[Objekt]

- Protokoll: http bzw. https
- Kontoname des Nutzers (→ Speicherort) als **Teil des DNS-Host-Namens**
- Dienst: blob, table oder queue
- Identifikation eines Objekts mittels Partitions- und Objektname

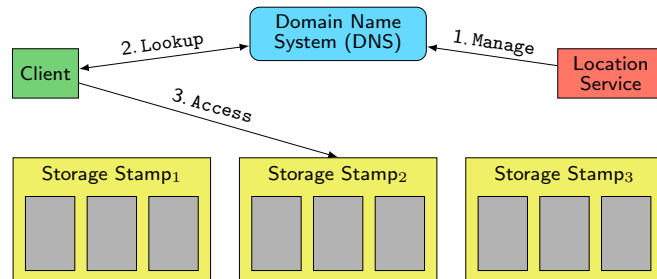
Architektur

■ Storage-Stamp

- **Gruppe aus mehreren Racks**
- Racks besitzen eigene Netzwerk- und Stromanbindungen → **Fehlerdomänen**
- Stamp von außen über eine eigene IP-Adresse erreichbar

■ Ortsdienst

- **Zuordnung von Nutzerkonten zu Stamps**
- Stamp-Auswahl für neue Konten
- Aktualisierung der Stamp-Adressen im DNS



Storage-Stamp

■ Front-End-Layer

- Authentifizierung eintreffender Anfragen
- Weiterleitung von Anfragen an den Partition-Layer

■ Partition-Layer

- Verwaltung von Blobs, Tabellen und Warteschlangen
 - Zusammenfassung kleiner Objekte
 - Aufteilung großer Objekte in Partitionen
- Verwaltung von Partitionen
 - Einteilung und Zuordnung zu Servern
 - Lastverteilung zwischen Servern
- Replikation über mehrere Stamps

■ Stream-Layer

- Direkter Zugriff auf Festplatten
- Bereitstellung von Datenströmen (*Streams*)
- Stamp-interne Replikation

Replikation

■ Replikation innerhalb eines Stamp (*Intra-Stamp Replication*)

- Aufgabe des Stream-Layer
- **Synchrone Replikation während des Schreibvorgangs**
 - Speicherung der Replikate in unterschiedlichen Fehlerdomänen
 - Replikation auf Binärdaten-Ebene
 - Typischer Replikationsfaktor: 3
- Im Fehlerfall: Rekonfiguration bzw. **Wechsel der Replikatgruppe**

■ Replikation zwischen Stamps (*Inter-Stamp Replication*)

- Aufgabenverteilung
 - Ortsdienst: Nutzerkonto-spezifische Konfiguration
 - Partition-Layer: Durchführung
- **Asynchrone Replikation im Hintergrund**
 - Replikation auf Objektebene
 - Durchschnittlich ca. 30s nach dem Schreibvorgang
 - Typischer Replikationsfaktor: 2
- Im Fehlerfall: Failover durch **Anpassung des DNS-Eintrags eines Kontos**

[Vergleiche: Einsatzszenario von Google's B4.]

Stream-Layer

Interne Datenstrukturen

■ Block

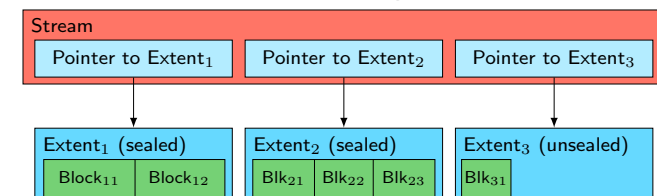
- **Kleinste Dateneinheit** für Lese- und Schreibaufrufe (variable Größe)
- Periodische Überprüfung der Datenintegrität mittels Checksummen

■ Extent

- NTFS-Datei mit aufeinander folgenden Blöcken
- Zustände
 - Unversiegelt (*unsealed*): **Anhängen** weiterer Blöcke möglich
 - Versiegelt (*sealed*): Nur noch lesender Zugriff erlaubt

■ Stream

- Liste von Referenzen auf Extents
- Nur der **letzte Extent eines Stream ist unversiegelt**



■ Extent-Nodes

- Datenspeicherknoten
- Aufgaben
 - **Speicherung von Extents**
 - Abbildung von Extent-Offsets zu Blöcken
- Mehrere Festplatten pro Rechner

■ Stream-Manager

- Verwaltungsknoten
- Aufgaben
 - **Erzeugung von Extents und Zuordnung zu Extent-Nodes**
 - Überwachung der Extent-Nodes
 - Extent-Replikation zur Kompensation nach Hardware-Ausfällen
 - Garbage-Collection für nicht mehr referenzierte Extents
- Verwaltung von Stream- und Extent-Informationen im Hauptspeicher
- Replikation des Stream-Manager-Zustands

[Vergleiche: Aufgabenverteilung zwischen Komponenten im Google File System]

■ Anlegen eines neuen Extent

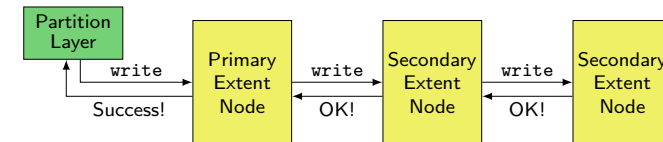
- Partition-Layer weist Stream-Manager an, einen neuen Extent zu erstellen
- Stream-Manager wählt drei Extent-Nodes (einen *Primary*- und zwei *Secondary*-Knoten) aus verschiedenen Fehlerdomänen aus

■ Hinzufügen eines Blocks zu einem Extent

[Vergleiche: Schreiben im Google File System]

- Partition-Layer sendet Block an Primary
- Primary zuständig für Koordinierung des Schreibaufrufs
 - Auswahl des Offset im Extent
 - Weiterleitung der Anfrage an die Secondaries
- Primary sendet Erfolgsbestätigung an Partition-Layer

→ Schreiben eines Blocks erfolgt **ohne Einbeziehung des Stream-Managers**



■ Fehlersituationen (Beispiele)

- Fehlermeldung, dass ein Extent-Node nicht erreichbar war
 - Fehlende Erfolgsbestätigung innerhalb einer vordefinierten Zeitspanne
- Partition-Layer kontaktiert Stream-Manager

■ Ausnahmebedingtes Versiegeln des aktuellen Extent

- Stream-Manager befragt Extent-Nodes nach aktuellem Extent-Offset
- **Versiegelung des Extent am kleinsten genannten Offset**

■ Anlegen eines (Ersatz-)Extent

- Auswahl einer neuen Gruppe von Extent-Nodes
- **Wiederholung der Anhängeoperation**

■ Anmerkungen

- Alle als „erfolgreich hinzugefügt“ bestätigten Daten bleiben erhalten
 - Ein einmal geschriebener Block wird **eventuell mehrmals gespeichert**
- Partition-Layer muss mit solchen Konsistenzgarantien umgehen können

■ Optimierung von Schreibzugriffen

- Problem
 - Intra-Stamp-Replikation erfolgt synchron → direkter Einfluss auf Antwortzeit
 - Primary muss auf Bestätigungen von Secondaries warten
 - Bestätigung kann erst erfolgen, wenn der Block persistent gesichert wurde
- **Instabile Antwortzeiten** in Überlastsituationen („hiccups“)
- Lösung
 - Einsatz einer zusätzlichen Festplatte (*Journal-Drive*)
 - **Doppelte Ausführung jeder Schreiboperation:** Journal-Drive + Daten-Disk
 - Senden der Bestätigung, sobald einer der beiden Aufrufe erfolgreich war

■ Lastbalancierung für Leseanfragen

- Festlegung einer **zeitlichen Schranke für die Bearbeitung einer Anfrage**
- Senden der Anfrage an einen für den Block zuständigen Extent-Node
- Extent-Node schätzt ab, ob sich die zeitliche Schranke einhalten lässt
 - Falls ja: Bearbeitung der Anfrage
 - Falls nein: Sofortige Ablehnung der Anfrage
- **Bei Ablehnung: Neuer Versuch** bei anderem Extent-Node

■ Zentrale Datenstruktur: **Objekttabelle**

[Vergleiche: Google's Bigtable]

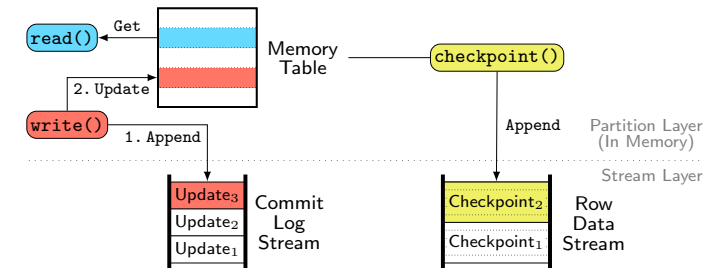
- Speicherung sehr großer Datenmengen [→ Petabytes]
- Aufteilung in **disjunkte Range-Partitions**
- Beispiele
 - Blob-Table: Tabelle mit allen Blobs eines Stamp
 - Partition-Map-Table: Zuordnung von Range-Partitions zu Objekttabellen

■ Komponenten

- Lock-Service
 - Vergleiche: Koordinierungsdienste [Siehe spätere Vorlesung.]
 - Vergabe von Leases für Range-Partitions an Partition-Server
- Partition-Server
 - **Verwaltung der ihm zugeteilten Range-Partitions**
 - Persistente Speicherung von Daten mittels Stream-Layer
- Partition-Manager
 - **Zuweisung von Range-Partitions zu Partition-Servern**
 - Mehrere Instanzen pro Stamp: Auswahl eines Anführers per Lock-Service

■ Kombination aus **flüchtigen und persistenten Datenstrukturen**

- Memory-Table für effizienten Lesezugriff
- Commit-Log-Strom zum Schutz vor Datenverlust



■ Erstellen von **Sicherungspunkten**

- Auslöser: Commit-Log erreicht eine bestimmte Größe
- Erzeugen eines Sicherungspunkts aus dem Inhalt der Memory-Table
- Aufräumen des Commit-Logs