Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen

Proseminar Speicher- und Dateisysteme
Betreuer: Dr. Michael Kuhn
Abgabedatum: 4. August 2015

Die Cloud

im Kontext von Speicher- und Dateisystemen

Vorgelegt von:
Sven Schmidt
4sschmid@informatik.uni-hamburg.de
Matrikelnummer: 6647018

Bachelor Informatik
2. Fachsemester

Inhalt

1 Einleitung

„Die Cloud“ ist in aller Munde. In Abbildung 1 sehen wir die Suchhäufigkeit des Begriffs bei Google im Verlauf von zehn Jahren. Offensichtlich (und das mag mit Sicherheit nicht nur mit schlechtem Wetter zusammenhängen) gab es einen signifikanten Anstieg ohne ein Anzeichen für eine baldige Abkühlung. Man könnte sagen: Das sind gute Aussichten, um sich eingehend mit dem Thema „Cloud“ zu befassen.

Der Grundstein für diese Entwicklung wurde wohl im Jahr 2006 gelegt, als – Zufall? – der damalige Google CEO Eric Schmidt den Begriff als erster im Kontext der Informatik benutzt hat. Auf der „Search Engine Strategies Conference“ sprach er über die Entwicklung der Branche und neue Modelle. Zum Thema Cloud-Computing eröffnete er mit der Feststellung, dass die Menschen noch nicht richtig verstünden, wie groß die Chancen seien, die sich böten [16]. Abbildung 1 belegt sicher, dass er Recht hatte, stand die Entwicklung doch damals noch ganz am Anfang. Cloud-Computing beginne mit der Prämisse, dass die Daten-Servivcs und -strukturen auf Server verlagert werden. „Wir nennen das Cloud-Computing - sie sollen irgendwo in einer Cloud sein“ [16].

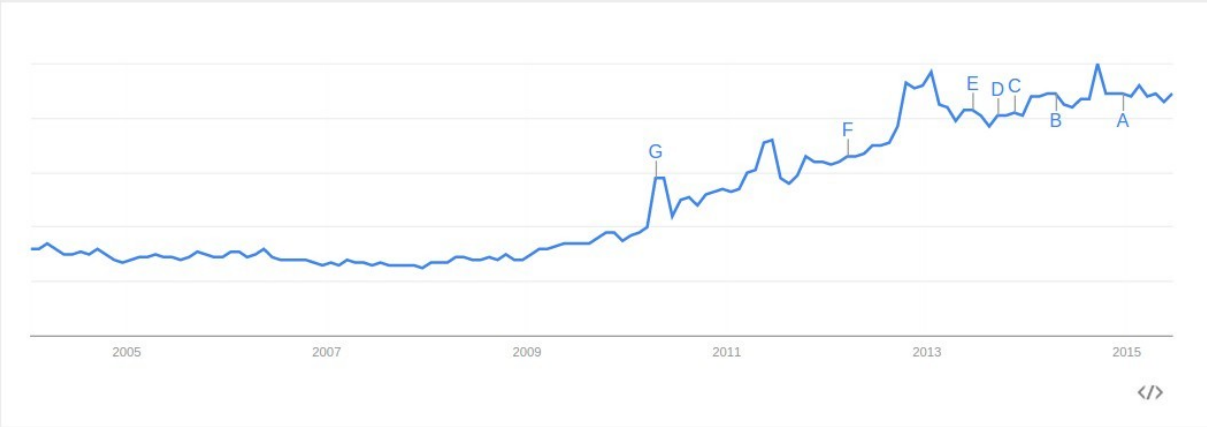

Abbildung 1: Google-Suchtrend zum Begriff „Cloud“ [17]

Heute – wir schreiben das Jahr 2015 – ist „Cloud“ zwar einer dieser Begriffe, von dem jeder eine vage Vorstellung darüber hat, was es ist und wie es funktioniert. Aber im Zuge einer Studie aus dem Jahr 2012 gaben trotzdem 56% der Befragten an, dass sie das Gefühl haben, Menschen würden über die Cloud sprechen ohne zu verstehen, was eigentlich gemeint ist. Das degradiert „Cloud“ zu einem dieser Buzzwords für Menschen, die sich in einem Themengebiet eher bedeckt halten und das verschleiern möchten. Dass 54% der Befragten angaben, die Cloud niemals zu nutzen, unterstützt diese These. Vor allem deshalb, weil von den Cloud-Skeptikern 95% sie eben doch nutzen, wenn auch ohne es zu merken. Umso wichtiger ist es, mit einem Maß an Neugierde zu fragen: Was steckt eigentlich dahinter? Wir wollen in diesem Papier einmal den Kopf in die Wolke stecken und uns mit dem Begriff auseinandersetzen: Was bedeutet „Cloud“ eigentlich? Anschließend geht es um die Entwicklung der Technologie (und vor allem auch darum, wieso die Entwicklung des Cloud-Computing ein logischer, weil notwendiger, Schritt war). Im Hauptteil betrachten wir das Innere der Cloud: Welche Speichertechnologien, welche Dateisysteme kommen infrage? Was für Charakteristika erfüllt die Cloud?

2 Etymologie

Ich habe bereits herausgestellt, dass es Eric Schmidt im Jahr 2006 war, der den Begriff der Cloud als erster fallen ließ. Er nutzte „Cloud“ im Zusammenhang mit der sukzessiven Verlagerung von Infrastruktur vom eigenen Rechner ins Netz. Ein logischer Schritt, wenn man bedenkt, dass beispielsweise Software von CDs, die wir früher durch mühseliges Klicken durch Installations-Assistenten auf den Rechner gebracht haben („fette“ Programme) durch immer schnellere Verbindungen problemlos ins Netz verlagert werden

kann. Da viele Menschen, die nicht „digital native“ sind, bereits den eigenen Computer nicht richtig verstehen, bietet sich der Begriff „Cloud“ geradezu an, um diesen Vorgang zu beschreiben: Wolken sind eben irgendwie neblig, sind nicht greifbar, sind mit Bedeutung (bzw. Wasserdampf) geladen. Und: Sie sind verteilt; wir wissen nicht, wo genau sich welcher Teil der Wolke in welchem Moment befindet. Das gilt eben auch für die Daten, die wir in die Cloud verlagern. Wer weiß schon, wo diese sich befinden? Für viele gilt sicher, dass sie lieber den Kopf in den Sand, als in die Cloud stecken. In der Informatik wollen wir jedoch ganz allgemein sagen: Unter der „Cloud“ verstehen wir „Rechnernetze, deren Inneres unbekannt oder unbedeutend ist“ [6]. Unbedeutend in dem Sinne, als es den normalen End-Benutzer nicht interessieren muss, wie die Cloud aufgebaut ist. Viele von uns nutzen beispielsweise Dropbox, speichern Backups, teilen Inhalte. Das mag auf viele gerade wie Magie wirken, wenn Dateien in einen lokalen Ordner verschoben werden, um dann in die Cloud hochgeladen zu werden. Aber wir müssen nicht wissen, wie die Architektur im Hintergrund aussieht. Und unbekannt ist das Innere dieser Netze analog deswegen, weil wir es gar nicht bis ins Detail kennen können. Wir werden später sehen, dass wir bestimmte Regeln ableiten können, die eine Cloud sinnvollerweise erfüllen sollte. Aber es gibt sie eben nicht, „die eine Cloud“ .

3 Die Entwicklung der Cloud

Doch mit Sicherheit haben die meisten Cloud-Systeme bestimmte Einsatzgebiete gemein. Durch immer schneller werdende Verbindungen, durch immer mehr Speicher-Kapazität, durch immer leistungsstärker werdende Prozessoren war es – wie kurz angedeutet – ein konsequenter Schritt, Rechenleistung, Software und Speicher weg vom eigenen Computer auf dedizierte Server zu verlagern. Es erlaubt uns also, effiziente Service-Leistungen, Speicher, Rechenkapazität, Datenbank-Leistung und vieles mehr zu nutzen. Denn schließlich merken wir den Unterschied immer weniger zwischen zuhause und „da oben“ . Doch wie bei vielen Dingen, die heute machbar sind (z.B. Top-Level-Domains mit der Endung .bayern registrieren) stellt sich die Frage: Warum? Warum sollte jemand das wollen? Was für Vorteile hat es? Vielleicht auch: Wie kann es das Leben verbessern, wie einfacher machen?

3.1 Amazons Weihnachtsproblem

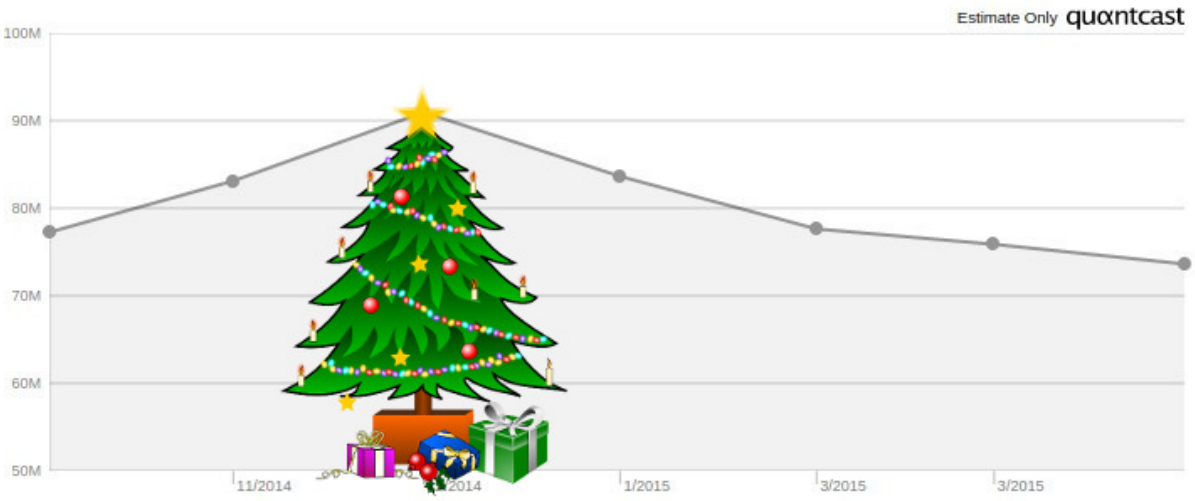

Abbildung 2: Amazon.com Zugriffszahlen 2014/15, Screenshot von [15], Weihnachtsbaum von [4]

Bereits sechs Jahre vor der ersten Verwendung des Wortes, also Anfang des Jahrtausends, gab es erste Anzeichen für die Entwicklung der Technologie. In Abbildung 2 sehen wir die Zugriffszahlen auf amazon.com in den Jahren 2014/15 mit einem deutlichen Maximum während der Weihnachtszeit. Das illustriert ein grundsätzliches Problem von Amazon, wie es auch schon im Jahr 2000 existierte: Die Spitzenlast, die auf die Server-Architektur drückt, ist zu Weihnachten ungleich höher als im Rest des Jahres. Wie trägt man dem Rechnung? Lässt man es zu, dass die Server unter der Last in die Knie gezwungen werden und potenzielle Kunden während der umsatzstärksten Zeit des Jahres die Seite nicht aufrufen können und – sehr wahrscheinlich – zur Konkurrenz wechseln? Kann man umgekehrt Server-Ressourcen bereithalten, die im Sommer nicht verwendet werden, nur um sie im „Fall der Fälle“ nutzen zu können (im Grunde wie dieses eine Hemd, das im Schrank hängt, „weil ja mal...“)? Die Antwort auf beide Fragen muss lauten: Natürlich nicht.

3.2 Die Cloud als logische Konsequenz

Die naheliegende Lösung ist die Cloud. Zunächst wurde die zweite Variante umgesetzt: Ein großer Pool aus Ressourcen wurde aufgebaut, das heißt Speicher, Rechenleistung, Datenbanken. Diesen Pool, und das ist der wesentliche Punkt, nutzt Amazon allerdings nicht alleine: Die Infrastruktur wird an End-Kunden weitervermietet. Es wurde also ein System aufgebaut, das später in die Amazon Web Services münden sollte. Im Gegensatz zu klassischen Hosting-Lösungen, wo ein Kunde sich zwischen einer überschaubaren Anzahl an Paketen entscheiden muss, erfüllte dieses System bereits damals eines der Haupt-Charakteristika der Cloud: Elastizität. In diesem Kapitel soll es uns reichen, Elastizität als ein „Ressourcen so viel man braucht wann man sie brauche“ zu definieren. Abgerechnet wird genau dieser Bedarf. Der Vorteil ist klar: Zu Spitzenlastzeiten (bei Amazon also z.B. Weihnachten) werden die Ressourcen so verteilt, dass die Server eben nicht einfach in die Knie gehen. Es geht sogar so weit, dass die Systeme innerhalb von wenigen Augenblicken die Skalierung verändern können. Dadurch sind Ausfälle der Server und Down-Zeiten der Webseiten nahezu ausgeschlossen.

3.3 Exkursion: Die alte und neue Welt im Vergleich

Dieses Modell ist evident anders, als es früher war. Der „klassische“ Weg war der, dass ein potenzieller Kunde einen der großen Hosting-Anbieter anruft (z.B. IBM) und sich einen Server mietet oder sogar ein eigenes Rechenzentrum aufbaut. Dabei weiß man also eindeutig, wo genau die Daten gespeichert sind und welche Architektur dahinter steckt. Es handelt sich also eben *nicht* um „Rechnernetze, deren Inneres unbekannt oder unbedeutend ist“ [6]. Die Cloud ist hingegen ein eher theoretisches Modell. Sie existiert (bzw. erscheint) zwar als ein großes, zusammenhängendes System, existiert in Wahrheit jedoch nur virtuell.

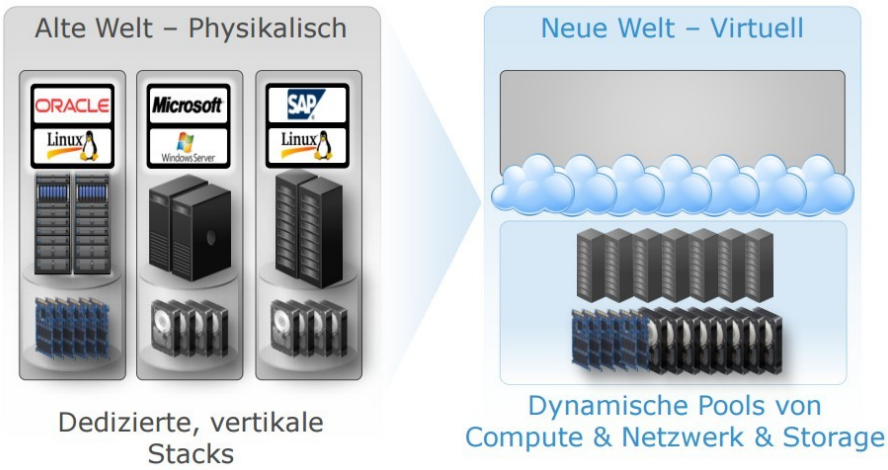

Abbildung 3: Vergleichsbild von [9]

4 Cloud-Architektur

Wir wissen nun, was der Begriff „Cloud“ bedeutet. Ich habe auch herausgestellt, warum die Cloud eine logische Entwicklung ist. In diesem Kapitel soll es nun endlich darum gehen, was eigentlich dahinter steckt. Im Kontext dieses Proseminars bedeutet das insbesondere, welche Speicher-Technologie sich am besten dazu eignet, Daten in der Wolke abzulegen und welchem Dateisystem die Aufgabe zukommen soll, diesen Speicher zu verwalten. Doch vorher wollen wir uns noch klar machen, welche Charakteristika die Cloud (also auch: Speicher- und Dateisystem) eigentlich erfüllen muss.

4.1 Die 5 Charakteristika der Cloud

Es existieren insbesondere fünf Charakteristika, die als anerkannte Kriterien dafür gelten, was genau die Cloud ausmacht. Die Definitionen wurden vom *National Institute of Standards and Technology* (NIST)[1] der Vereinigten Staaten verfasst und im September 2011 veröffentlicht[2].

4.1.1 On-demand self-service

Wir haben bereits im Kontext von Amazon über das Kriterium der Elastizität gesprochen. Dabei wurde impliziert, dass der Benutzer nichts dafür tun muss, dass die Provisionierung der Ressourcen erfolgt. Genau diese Tatsache nennen wir *on-demand self-service*. Im Kontext von Speicher- und Dateisystemen bedeutet dies insbesondere, dass die Kapazität des Systems die Illusion von Unbegrenztheit erweckt. Das System teilt also jedem User automatisch und ohne Zutun eines Menschen bedarfsgerecht Ressourcen (Speicher, Rechenleistung) zur Verfügung.

4.1.2 Broad network access

Broad network access (etwa *ausgedehnter Netzwerk-Zugriff*) definiert die Zugriffs-Wege, im weiteren Sinne also die Frage, wie auf die Cloud zuzugreifen ist. Wir müssen uns die Cloud wie ein System vorstellen, das im Inneren Ressourcen bereithält und deren Hülle Schnittstellen zur Verfügung stellt, um mit diesen Inhalten kommunizieren zu können. Wichtig sind dabei insbesondere die Standard-Technologien, d.h. beispielsweise Zugriffsmöglichkeiten über das Hypertext Transfer Protokoll. Gemeint ist aber auch eine Streuung der Geräte: Wir wollen Zugriffsmöglichkeiten über Computer, Laptops, Tablets, Phablets, Smartphones – und möglicherweise auch intelligente Kühlschränke.

4.1.3 Resource pooling

Ressourcen werden zusammengeschlossen und als ein gemeinsamer Pool betrachtet, aus dem heraus die Provisionierung erfolgt. Dabei ist es nicht wichtig, um was für Systeme es sich handelt, wo diese stehen, was für Dateisysteme darauf laufen. Cloud-Dateisysteme übernehmen die Virtualisierung (aus vielen Ressourcen eine große abstrahieren). Und vor allem übernehmen Cloud-Dateisysteme auch die Bereitstellung einer generischen Schnittstelle.

4.1.4 Rapid elasticity

Da der Begriff der Elastizität in diesem Papier bereits mehrfach in verschiedenen Kontexten gefallen ist, sehen wir, dass er wichtig sein muss. Insofern ist es nicht verwunderlich, dass sie als Charakteristikum der Cloud gilt. Das NIST geht hier noch einen Schritt weiter und nennt *Rapid elasticity* als Kriterium, im Sinne von kurzfristig. Kurzfristig ist hier vor allem so zu verstehen, dass wir auf die Umverteilung der Ressourcen (z.B. bei höherem Bedarf) nicht lange warten müssen (können, wollen). Das geschieht innerhalb weniger Sekunden.

[1] etwa wie das Deutsche Institut für Normung
[2] Siehe [13]

4.1.5 Measured service

Das letzte Kriterium definiert die Art und Weise der Abrechnung. Die Cloud sollte so aufgebaut sein, dass ein *„pay as you go“* -Modell möglich ist. Das heißt zum einen, dass Dateien eindeutig einem Besitzer zuzuordnen sind. Das heißt aber auch, dass bestimmte Messgrößen existieren müssen, um den Verbrauch zu „vermessen“ . Das sind z.B. die klassischen Einheiten wie Byte und ihre Steigerungen Kilobyte, Megabyte und Gigabyte, aber auch abstrakte, zusammengesetzte Größen wie *Database throughput units* im Fall von Microsoft Azure.

4.2 Object-based storage als Cloud-Speicher

Fünf Charakteristika – oder: Kriterien – sind eine hohe Messlatte für Systeme, denen es trotzdem Rechnung zu tragen gilt. Im Rahmen dieses Proseminars liegt der Fokus auf Speicher- und Dateisystemen, die aber ebenfalls so designt sein sollten, dass sie die fünf Kriterien der Cloud erfüllen. Die Frage lautet also: Wie sollten Dateien in der Cloud gespeichert werden? Ich möchte hier insbesondere auf den Objekt-Speicher eingehen, der in vielen Cloud-Systemen wie z.B. den Amazon Web Services Anwendung findet.

4.2.1 Überblick: Block-Speicher vs. Objekt-Speicher

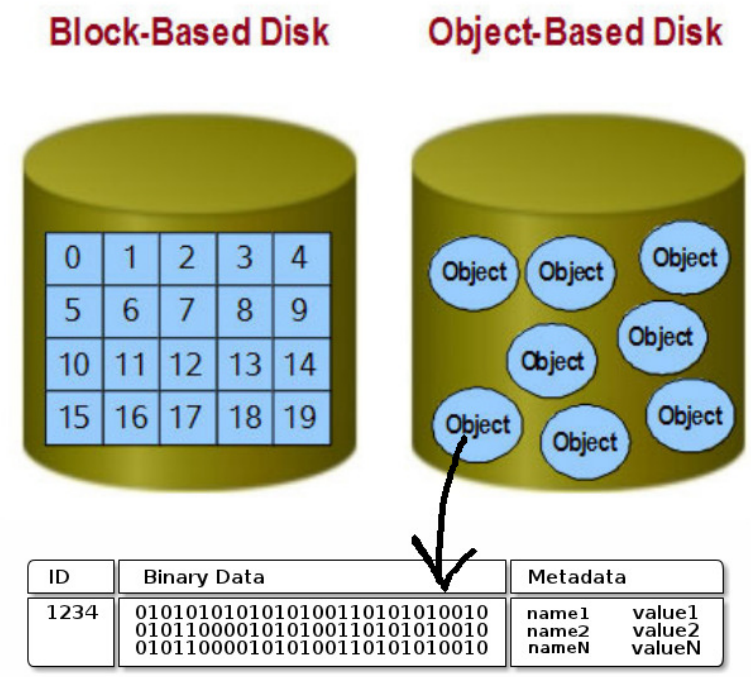

Abbildung 4: Vergleich zwischen Block- und Objekt-Speicher [1]

Abbildung 4 zeigt einen einfachen Vergleich zwischen Block-basiertem Speicher, wie wir ihn häufiger im Rahmen dieses Proseminars gesehen haben, und Objekt-Speicher. Objekte sind also eine Speicherform, die als Alternative neben Block- und auch Dateispeichern existiert. Sie bestehen aus drei Teilen: Den - binären - Daten selbst, Meta-Daten wie Dateityp und Besitzer, und einem globally unique identifier (GUID). Letzterer ist vor allem dafür wichtig, ein Objekt global, d.h. über das gesamte Dateisystem hinweg eindeutig zu identifizieren. Aber über die GUID sollen auch Konflikte bezüglich der Identifier vermieden werden: Verschiedene Dateisysteme können ja durchaus dieselbe ID unabhängig voneinander vergeben. Aufgrund dieser gemeinsamen, assoziativen Speicherung von Daten, Meta-Daten und GUID

sind Objekte in der Lage, für sich selbst zu existieren: Sie haben keine Abhängigkeiten z.B. zu Metadaten-Speichersystemen, wie es bei Block-Speichern der Fall ist, bei denen die Adressen bekannt sein müssen. Die Speicherung der Meta-Daten im Objekt verleiht diesen Kontext. Dabei ist die Menge und die Art der Meta-Daten variabel und abhängig vom Administrator des Systems. Ein weiterer Unterschied besteht in der Art, *wie* gespeichert wird. Objekte werden flach organisiert, d.h. sie liegen auf derselben Ebene und es existieren keine Hierarchien wie beispielsweise Verzeichnisse. Wir sehen also: Objekt-Speicher unterscheidet sich stark vom Block-Speicher. Während alle Speichermedien, die jemand sich privat zulegt, auf Blöcke setzen, eignet sich Objekt-Speicher vor allem für massive, ungeordnete Datenmengen, die verteilt gespeichert werden sollen und konkurrierende Zugriffe ermöglichen sollen. Damit eignet er sich perfekt für die Cloud.

4.2.2 Verteilbarkeit von Objekten

Aufgrund der Architektur der Block-Lösung ist es nicht möglich, Dateien zu zerschneiden und an beliebig vielen Orten zu speichern. Bei Objekten ist das anders. Objekte sind dafür designt, quasi unendlich oft zerschnitten werden zu können.

Abbildung 5: Symbolbild nach [18]

Es muss sich nur gemerkt werden, an welcher Stelle im System welcher Part des Objekts liegt. Setzen wir die Daten wieder in der richtigen Reihenfolge zusammen, so erhalten wir das ursprüngliche Objekt. Im Kontext der binären Daten bedeutet das ja nur, dass wir eine sequenzielle Folge von 1en und 0en wieder zusammensetzen. Diese Teilbarkeit erlaubt es, Objekte auf mehrere Systeme zu verteilen. Beachten wir Redundanzen, so ist es sogar möglich, den ersten Part eines Objekts durch den äquivalenten, zweiten Teil einer Kopie zu ergänzen.

Abbildung 6: Zusammensetzen von Objekten, Symbolbild [11]

4.2.3 Zugriff über HTTP-Schnittstellen

Abbildung 7: Amazons Storage Interface, Darstellung verändert nach [3], [5], [19]

Typischerweise erfolgt der Zugriff auf Cloud-Resourcen über HTTP-Schnittstellen (REST, SOAP). In Abbildung 7 sehen wir ein vereinfachtes Beispiel mit der Syntax, wie sie von den Amazon Web Services genutzt wird: Eine einfache Anfrage wird gestellt (über die GET-Methode). Die Cloud wertet den Zugriff aus, prüft z.B. die Autorisierung und liefert am Ende das entsprechende Objekt zurück, im Beispiel ein Bild. Dabei werden die bekannten Methoden genutzt, die im Web bekannt sind: GET, um Inhalte zu holen, POST, um sie anzulegen, PUT zum aktualisieren und OPTIONS um Informationen abzufragen.

4.3 Cloud-Dateisysteme

Um diese Aufgaben - Teilung von Objekten, Tracking der Position der Daten-Parts, Herstellen von Redundanzen, Zugriffe über APIs - zu ermöglichen, benötigen Cloud-Systeme mächtige Dateisysteme, die viel mehr können müssen, als ein lokales Dateisystem.

4.3.1 Was Cloud-Dateisysteme leisten müssen

Welche Eigenschaften Cloud-Dateisysteme noch leisten müssen, ist in der folgenden Aufzählung zusammengefasst:

- Daten-Verluste müssen kompensierbar sein. Das wird durch entsprechendes, redundantes Speichern der Daten ermöglicht. Es ist allerdings auch wichtig, dass diese Redundanzen bestimmten Eigenschaften Rechnung tragen. Da die Cloud global ist, sollten Kopien von Dateien nicht im selben Rechenzentrum, vielleicht nichtmal im selben Land gespeichert werden. Durch kluges Verteilen können Daten so gegen alle möglichen Risiken abgesichert werden, so z.B. auch politische Krisen und Katastrophen.

- Entsprechend müssen die Systeme selbst-heilend sein. Sie sollen dazu in der Lage sein, den Verlust einer Datei, eines Speicher-Systems, einer ganzen Reihe an physischen Speicher-Einheiten verkraften und die Daten wiederherstellen zu können.
- Der Standort der Daten darf keine Rolle spielen. Selbstverständlich ist ein Zugriff auf Daten auf einem USB-Stick in meiner eigenen Wohnung kein Problem. Für die Cloud darf ein Zugriff auf Dateien von Festplatten in China aber genauso wenig ein Problem sein, wie der Zugriff von einem Rechenzentrum in Hamburg.
- Latenzzeiten müssen stemmbar sein. Liegen Objekt-Parts an verschiedenen Orten, dürfen keine Fehler durch evtl. Wartezeiten auftreten.
- Verschiedene Dateisysteme müssen sinnvoll virtualisiert werden.

Zwei Beispiele für Cloud-Dateisysteme sind Lustre (Linux Cluster) und CephFS.

4.3.2 Beispiel: Lustre

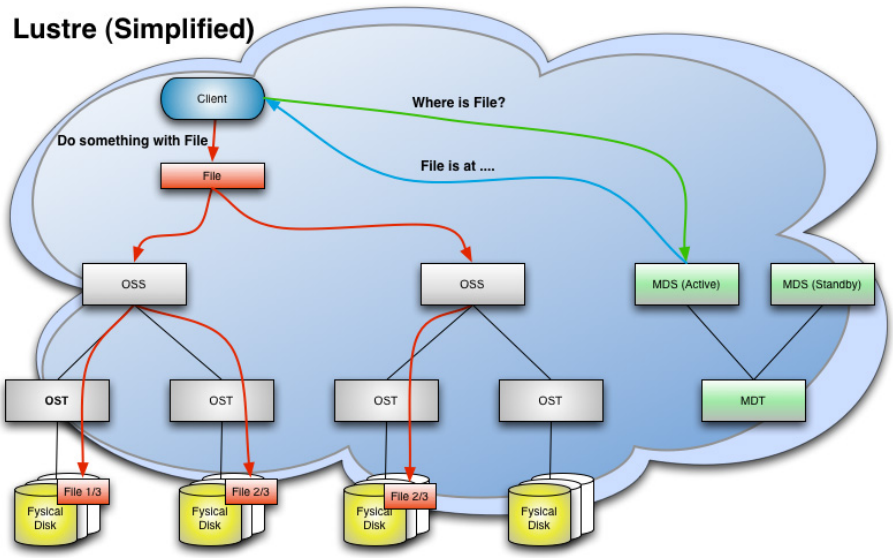

Abbildung 8: Lustre Übersicht [12], Logo [2]

Lustre ist ein Cluster-Dateisystem, das von 15 der 30 Top-Computer weltweit eingesetzt wird. Darunter auch die Plätze 2 und 3 (Titan und Sequoia). Es besteht aus drei Haupt-Komponenten: dem Client, den Metadata-Servern (MDS) und den Objekt-storage-servern (OSS). Der Client ist die einfachste Komponente: Damit ist der Benutzer gemeint, der auf eine Datei im Dateisystem zugreifen möchte. Der Zugriff wird an einen Metadata-Server weitergereicht. Abbildung 8 zeigt eine ältere Version von Lustre, wo es einen aktiven und einen passiven MDS gibt, der bei Bedarf den aktiven MDS ersetzen kann. In neueren Lustre-Versionen können unbegrenzt MDS eingesetzt werden. Für jedes Dateisystem, das am unteren Ende angeschlossen ist, existiert ein Metadata-Target (MDT), auf dem die Meta-Daten gespeichert sind, also wo die Datei liegt, welche Teile zu ihr gehören, wer sie zugreifen darf usw. Anders als bei block-basierten Systemen sind MDS nicht aktiv am Prozess der Daten-Verteilung involviert. Sie stellen lediglich Hilfsmechanismen bereit, um Dateien im System aufzufinden. Diese Informationen zeigen dann auf einen oder mehrere Object-storage server (OSS). OSS sind höhere Sinneinheiten, die für die Speicherung der Daten zuständig sind und denen bis zu acht Object-storage targets (OST) zugeordnet sind, die letztendlich den richtigen, physischen Speicher repräsentieren.

4.3.3 Beispiel: CephFS

Ceph ist im Allgemeinen ein fehler-tolerantes, verteiltes Cluster-Dateisystem, das – im Gegensatz zu Lustre – aktiv im Bereich Cloud eingesetzt wird. Vereinfacht gesagt gibt es eine Vielzahl an Servern, die jeweils Speicher zum System beitragen. Darüber ist ein Dateisystem gespannt, das den Servern adäquate Schnittstellen zur Verfügung stellt. Wenn Fehler passieren – also z.B. eine Speichereinheit ihren Dienst einstellt –, dann ist das in Ordnung und kann aufgrund der Architektur vom System kompensiert werden. Allerdings ist Ceph mehr als das (vgl. Abb. 9). Der wirklich interessante Teil befindet sich unter der Oberfläche des Dateisystems, der **R**eliable **A**utonomic **D**istributed **O**bject **S**tore (RADOS). Im Grunde lässt sich das übersetzen als die performante Speicherung von Objekten in einem verteilten System.

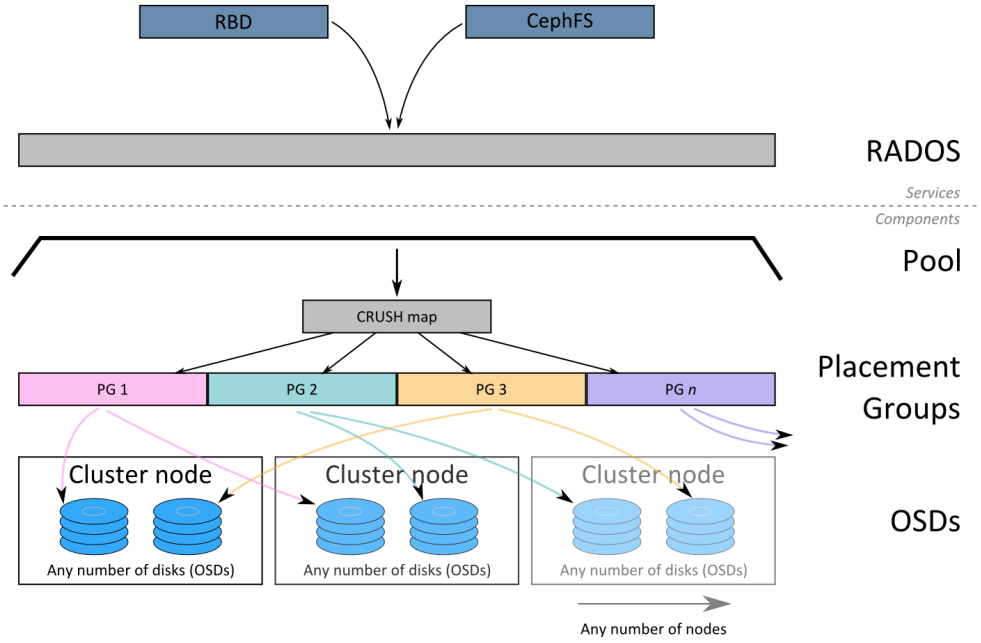

Abbildung 9: Ceph File System Übersicht [8]

Das führt uns zu den Komponenten des Systems. Am unteren Ende haben wir die **Object storage devices** (OSDs). Damit sind im Grunde die „richtigen“ , physischen Speicher-Einheiten gemeint, die in Form eines Verzeichnisses in Ceph repräsentiert werden: */var/lib/ceph/osd-1*. Wir haben bereits darüber gesprochen, dass eine wichtige Eigenschaft von Cloud-Dateisystemen die redundante Speicherung von Daten ist. Deshalb können die Dateien nicht direkt vom Dateisystem auf die OSDs verteilt werden. Es muss ein System zwischengeschaltet werden, das die Verteilung übernimmt. Zunächst werden hier die OSDs in **Placement Groups** (PGs) organisiert. Gemäß offizieller Dokumentation [14] übernehmen die PGs vor allem das Tracking der Objekte und die Verteilung auf die OSDs, da es zu kostenintensiv wäre, jedes Objekt einzeln von den oberen Schichten des Systems organisieren zu lassen. Denn dann müssten die oberen Komponenten jedes Objekt „kennen“ . PGs sind in dem Sinne also keine eigenen, intelligenten Systeme, sondern lediglich Anhäufungen von OSDs. Wichtig zu erwähnen ist vor allem, dass ein OSD auch in mehreren Placement Groups organisiert sein kann. Die eigentliche Verteilung erledigen dann **CRUSH-Maps**. Gemäß Dokumentation[7] ist es der CRUSH Algorithmus, der „weiß“ , wo ein Objekt zu speichern ist bzw. umgekehrt, wo es sich befindet. CRUSH-Maps übernehmen also die Verteilung, das Tracking und auch die Vervielfältigung. Dabei handelt es sich um Regeln, die vom System-Administrator festgelegt werden. Das sind z.B. bestimmte Eigenschaften, die ein OSD erfüllt und von denen der CRUSH-Algorithmus feststellt, dass ein bestimmtes Objekt auf einem OSD mit dieser Eigenschaft zu speichern ist. Die CRUSH-Maps und die untergeordneten Ebenen werden in einzelnen Pools organisiert. Ein Pool ist die Ebene, deren Oberfläche die Interaktion mit dem Benutzer ermöglicht und beispielsweise Schnitt-

stellen (z.B. REST) zur Kommunikation anbietet. Mit all diesen zusammenarbeitenden Komponenten ist CephFS ein sehr gutes Beispiel für ein Cloud-Dateisystem, da es offensichtlich die fünf genannten Charakteristika erfüllt.

4.4 Nachteile

Zum Abschluss möchte ich noch zwei Nachteile von Objekt-based Storage bzw. Cloud-Dateisystemen diskutieren: Objekt-Speicher lässt keine In-Place-Changes zu, und aufgrund der Architektur der Cloud entsteht ggf. ein HTTP Overhead.

4.4.1 Keine In-Place-Changes

Während es bei Block-Speichern kein Problem ist, einzelne Blöcke zu verändern, sieht es bei Objekt-Speicher anders aus: Es sind keine In-Place-Changes möglich. Typischerweise werden nur die Operationen *anlegen*, *löschen* und *lesen* angeboten, nicht jedoch *ändern*. Jede Veränderung an einem Objekt erzeugt eine neue Version desselben, an der gearbeitet wird. Somit werden bei konkurrierenden Veränderungen verschiedene Kopien des Objekts erzeugt, sodass am Ende des Prozesses verschiedene Versionen existieren [10]. Das ist bei häufigen Änderungen natürlich sehr kostenintensiv.

Allerdings muss relativiert werden: Bei Cloud-Systemen wird davon ausgegangen, dass Dateien selten geschrieben, dafür aber umso öfter gelesen werden (*write once, read often*). Damit eignet sich Cloud-Speicher besonders gut für statische, sich selten verändernde Daten. Frequenziell ändernde Daten sind woanders besser aufgehoben.

4.4.2 HTTP-Overhead

Abbildung 10: HTTP-Request in den Google Chrome Dev Tools

Abbildung 10 zeigt einen Screenshot aus den Developer Tools von Google Chrome. Ich habe dort zur Verdeutlichung des Problems einen einfachen HTTP-Request über die GET-Methode an einen lokalen Web-Server gesendet. Die Datei beinhaltet ein Objekt in JavaScript-Objekt-Notation (*JSON*), das unter dem Schlüssel „data“ den String „hallo welt!“ beinhaltet. HTTP hat jedoch die Eigenschaft, dass neben dem eigentlichen Content (der hier 27 Byte umfasst) diverse Header mitgeschickt werden: Das sind z.B. generelle Informationen über den Request wie der Status Code, aber auch tiefergehende Mitteilungen über die Server-Architektur (was wird an Sprachen akzeptiert? Sind Inhalte komprimiert?). Es entsteht

also ein *Overhead* (etwa *Überhang*) an Daten. Bekannt ist das Problem in einfacherer Form z.B. bei der Auslieferung von statischen Inhalten (JavaScript, CSS, Bilder) auf Seiten, die Cookies speichern. Die Cookies werden beim Aufruf dieser Dateien trotzdem mitgesendet, obgleich sie in derartigen Kontexten keine Bedeutung haben. Bei vielen millionen Zugriffen ist das eine beträchtliche Datenmenge.

5 Zusammenfassung

Ziel dieses Papiers war es, einen Überblick über Cloud-Speicher im Kontext von Speicher- und Dateisystemen zu vermitteln und vielleicht sogar den Horizont all jener zu erweitern, die Wolken bisher für ein Wetter-Phänomen halten. Zusammengefasst ist mir folgendes besonders wichtig:

- **Die Cloud ist bereits da**
  Server-Architekturen verlagern sich sukzessive in die Cloud. Amazon war einer der ersten Vorreiter in diesem Prozess und hat ein Cloud-System aufgebaut, um skalierbare Systeme zu erhalten. Cloud ist vor allem kein einfaches Buzzword der Technologie-Branche, sondern nützlich für jeden von uns.
- **Der Begriff suggeriert Einfachheit**
  Im Hintergrund stehen jedoch komplexe Architekturen.
- **Die Cloud muss 5 Charakteristika abdecken**
  Diese sind: on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service
- **Die Cloud braucht mächtige Dateisysteme**
  Cloud-Dateisysteme müssen skalierbar und elastisch sein, Latenzzeiten ausgleichen können, Ressourcen sinnvoll virtualisieren, sie müssen ausfallsicher sein... Insgesamt müssen Dateisysteme in der Cloud viel mehr leisten, als lokale Dateisysteme.
- **Objekt-basierter Speicher eignet sich perfekt für Cloud-Speicher**
  Daten sind mit Objekten partiell speicherbar, flach organisiert und eindeutig identifizierbar.

6 Quellenangaben

[1] URL: http://ceph.com/docs/master/_images/ditaa-ae8b394e1d31afd181408bab946ca4a216ca44b7.png.

[2] Lustre Logo. URL: https://en.wikipedia.org/wiki/File:Lustre_file_system_logo.gif.

[3] Amazon.com. *Amazon REST API*. Screenshot. URL: http://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectGET.html.

[4] *Christmas tree*. URL: http://images.clipartpanda.com/christmas-tree-clipart-christmas-tree10.png.

[5] *Cloud*. URL: http://www.udldigital.de/wp-content/uploads/2013/03/cloud.gif.

[6] *Cloud Computing*. URL: https://de.wikipedia.org/wiki/Cloud_Computing (abgerufen am 21. 06. 2015).

[7] *Crush maps*. URL: http://ceph.com/docs/master/rados/operations/crush-map/.

[8] Barney Desmond. 14. Sep. 2012. URL: http://www.anchor.com.au/blog/2012/09/a-crash-course-in-ceph/ (abgerufen am 21. 06. 2015).

[9] *George Greenleaf with EMC - IT Transformation – Stalwart Executive Briefing 2012*. 8. Mai 2012. URL: http://de.slideshare.net/StalwartAcademy/emc-it-transformation-stalwart-executive-briefing-2012.

[10] Jacob Gsoedl. *Advantages of using an object storage system*. URL: http://searchcloudstorage.techtarget.com/tip/Advantages-of-using-an-object-storage-system (abgerufen am 01. 08. 2015).

[11] *Link holding triforce*. URL: http://orig12.deviantart.net/fc2e/f/2012/220/8/1/alttp_link_holding_triforce_sprite_by_eri_tchi-d4lcrgy.png.

[12] *Lustre schema*. 3. Juli 2010. URL: http://louwrentius.com/static/images/lustre-schema.jpg.

[13] Peter Mell und Timothy Grance. *The NIST Definition of Cloud Computing*. URL: http://www.seu.ac.lk/careerguidanceunit/freedownload/0000%20The%20NIST%20Definition%20of%20Cloud%20Computing.pdf.

[14] *Placement groups*. URL: http://ceph.com/docs/master/rados/operations/placement-groups/.

[15] Quantcast. *Amazon.com Traffic and Demographic Statistics by Quantcast*. URL: https://www.quantcast.com/amazon.com (abgerufen am 21. 06. 2015).

[16] Eric Schmidt. *Search Engine Strategies Conference*. 9. Aug. 2006. URL: http://www.google.com/press/podium/ses2006.html.

[17] *Trend für den Begriff Cloud*. URL: https://www.google.de/trends/explore#q=cloud (abgerufen am 30. 06. 2015).

[18] *Triforce*. URL: http://stickerish.com/wp-content/uploads/2011/09/TriForceYellowSS.png.

[19] *Zelda cat*. URL: https://wrathofnino.files.wordpress.com/2009/07/zelda.jpg.