Cloud Storage: Wie viel Cloud Computing steckt dahinter?

Michael C. Jaeger und Uwe Hohenstein

Corporate Research and Technologies, Siemens AG
System Architecture & Platforms (CT T DE IT 1)
Otto-Hahn-Ring 6
D-81730 München
{uwe.hohenstein | michael.c.jaeger}@siemens.com

Kurzfassung: Durch die hohe Dynamik im Bereich des Cloud Computings entstehen neues Potenzial, aber auch Risiken bezüglich existierender und neuer Anwendungen. Eine wichtige Ressource bei der Nutzung von Cloud Computing ist sicherlich Cloud Storage. Unter Cloud Storage werden Datenspeicher subsumiert, die in unterschiedlichen Varianten von Cloud Computing Betreibern angeboten werden. Die wichtige Frage für die Nutzung von Cloud Storage ist hierbei, inwiefern auf technischer Ebene die wesentlichen Eigenschaften des Cloud Computings wie Elastizität, Skalierbarkeit und Kostenreduktion umgesetzt werden. Anhand von Beispielen wird deutlich, dass viele dieser Eigenschaften nach derzeitigem Stand nicht vollständig erfüllt werden.

1 Einleitung

Im Bereich Cloud Computing können wir eine große Veränderung im Markt feststellen, da etablierte und neue Anbieter mit neuen Produkten auftreten. Gleichzeitig sind große Investitionen zu beobachten, die in den Aufbau von Datenzentren und Plattformtechnologien fließen. Diese Veränderung schafft Potenzial in Form von neuartigen Anwendungen oder Kosteneinsparungen bei existierenden Anwendungen. Gleichzeitig entsteht die Gefahr, dass bisherige Geschäftsmodelle nicht mehr aufrecht erhalten werden können. Daher ist Cloud Computing für Unternehmen von großer Bedeutung.

Bausteine des Cloud Computings, wie zum Beispiel Virtualisierung oder Service-Orientierung, sind bereits zuvor in verschiedenen Formen auf technologischer Ebene realisiert worden. Jedoch waren bisher diese Bausteine nicht in dieser Breite und Kombination am Markt verfügbar und haben nicht zu den konkreten Vorteilen geführt, die einem das Cloud Computing nun bietet. Als Hauptvorteile zählen Armbrust et al. auf [AFG+09]:

1. Im Vergleich zu der Ressourcen-Kapazität, die durch eigene Mittel aufgebaut werden kann, scheinen die Ressourcen, die über Cloud Computing eingekauft werden können, praktisch unbegrenzt.

2. Die Investitionen sind sehr gering. Man geht praktisch keine relevanten Verpflichtungen ein, und Anschaffungskosten entfallen größtenteils. Zudem wird nach tatsächlichem oder reserviertem Verbrauch bezahlt. Dies ist gemeinhin als Pay-as-you-go-Prinzip bekannt.
3. Es ist möglich, punktuell Ressourcen je nach Bedarf mal mehr oder weniger zu beziehen und auch nur den aktuellen Bedarf zu bezahlen. Man spricht hierbei von der *Elastizität* der Ressourcen.

Betrachten wir die Entwicklungen in den vergangenen Jahren, sehen wir noch weitere Vorteile hinzukommen:

4. Bei einigen Angeboten ist der administrative Aufwand verglichen mit dem Betrieb einer eigenen Infrastruktur geringer.
5. Durch den Betrieb in Datenzentren wird ein gewisses Maß an Ausfallsicherheit und Hochverfügbarkeit erreicht, welches bei eigener Infrastruktur sonst nur durch erhöhte Investitionen erreicht würde. Z.B. ist das Aufsetzen und Administrieren von Datenbank-Clustern mit einem erheblichen Aufwand verbunden, der durch einen Cloud-Anbieter abgenommen werden kann.

Cloud Computing umfasst im Wesentlichen die Ressourcen Rechenkapazität, Applikationen und Speicher. In diesem Beitrag soll Letzteres im Vordergrund stehen. Bei Speicher beziehen wir uns dabei nicht auf Arbeitsspeicher, sondern auf persistenten Speicher, dem sogenannten *Cloud Storage*, den Anwendungen nutzen können.

Cloud Storage kann in zwei Formen genutzt werden: Zum einen ist es für Cloud Anwendungen sinnvoll, einen Speicher in der Cloud zu nutzen, anstatt einen eigenen Datenbankserver in die Cloud einzuspielen, aufzusetzen und zu administrieren. Zum anderen kann es auch für Anwendungen außerhalb der Cloud interessant sein, um z.B. Daten in einer Cloud auszulagern und dadurch Administrationskosten zu sparen. Da ein Cloud Storage direkt vom Internet aus erreichbar ist, wird zudem eine große Breite von Zugriffswegen ermöglicht.

Wenn Cloud Storage für neue Projekte in Betracht kommt, stellen sich unmittelbar folgende Fragen: Wie sind die Vorteile des Cloud Computings für Cloud Storage genau ausgestaltet? Was sind die technischen Begebenheiten? Ausgehend von existierenden Angeboten haben wir den Eindruck gewonnen, dass die gemeinen Vorteile derzeit nicht in allen Aspekten erfüllt werden. Im Folgenden diskutieren wir daher die Punkte Elastizität, Kosten und Skalierbarkeit exemplarisch an diversen Cloud Storage Lösungen. Zudem werden datenbankrelevante Punkte wie Performanz, Ausfallsicherheit und Administration beleuchtet. Damit soll eine Grundlage geschaffen werden, die Entscheidern dient, die Eigenschaften des Cloud Storage richtig einzuschätzen.

Der folgende Teil des Beitrags gliedert sich wie folgt: Im nächsten Abschnitt beschreiben wir die grundlegenden Konzepte der verschiedenen Cloud Storage Lösungen. Darauf folgend diskutieren und analysieren wir potenzielle Fallstricke und Probleme, illustriert durch Beispiele basierend auf dem momentanen Entwicklungsstand. Der Beitrag endet mit dem Fazit, das aus der Analyse gezogen wird.

2 Speicherkategorien in der Cloud

Cloud Storage ist eine bedeutende Basis-Ressource, die auf Infrastrukturebene im Rahmen des Cloud Computings angeboten wird. Mehrere Gründe sprechen dafür, eine Cloud Storage Lösung einzusetzen. Zum einen gelten die im vorangegangenen Abschnitt angeführten Vorteile und Motivationen für Cloud Computing. Zum anderen bieten Anbieter gleichzeitig nicht nur den Speicher, sondern führen auch Datensicherungen durch und bauen auf eine Mehr-Knoten-Architektur, der den klassischen Cluster Deployments nahekommt [LMM09]. Dies bedeutet, dass Produkte angeboten werden, die über einen einfachen Speicher deutlich hinausgehen. Bezüglich der Speicherung von Daten in der Cloud gibt es grundsätzlich zwei architekturelle Anordnungen:

- Speicher für Datenbankanwendungen in der Cloud: Eine Anwendung läuft in der Cloud und benötigt eine Datenbank zur Verwaltung persistenter Daten. Auch wenn prinzipiell die Cloud-Applikation ein On-Premise-Datenbankserver nutzen kann, so ist es aus Sicherheitsgründen (der Zugriff auf das DBS erfolgt von außerhalb der eigenen Organisation) und aus Latenzgründen wenig praktikabel. Sofern keine geschäftlichen Gründe gegen eine Speicherung der Daten in der Cloud sprechen, ist ein Aufsetzen der Anwendung und des Cloud Storages in der Cloud die technisch optimale Lösung.
- Auch wenn die Anwendung nicht in der Cloud aufgestellt wird, kann man dennoch von Cloud Storage profitieren. Mit der Nutzung von Cloud Storage kann auf die Anschaffung wie auch die Administration eines eigenen Datenbankservers verzichtet werden. Insbesondere das Aufstellen einer hochverfügbaren Cluster-Lösung benötigt einen erheblichen Aufwand. Zudem erhält man durch den Anbieter eine weltweite Erreichbarkeit, die gleichzeitig eine hohe Anzahl von Nutzern bedienen kann. Weiterhin kann man von neuartigen NoSQL-Angeboten [NoSQL] von einfacheren Datenspeichern profitieren, die nicht die Komplexität für die Planung und Betrieb eines relationalen Datenbankservers erfordern.

Es gibt im Wesentlichen drei Kategorien von Cloud Storage, die im Folgenden diskutiert werden sollen: Blob Storage, Table Storage und echte Datenbankserver. Weitere Kategorien wie der Plattenspeicher (z.B. Amazon EBS oder Windows Azure Drives) oder vorgefertigte Images z.B. von Oracle für Amazon sollen hier außer Acht gelassen werden.

2.1 Blob Storage

Blob Storages sind gedacht für die Speicherung von großen Binär- oder Textdaten („binary large objects“) wie z.B. Bilder, Software oder XML-Dokumente. Das Konzept Blob ist bereits aus der Welt der relationalen Datenbankserver bekannt. Blobs lassen sich beispielsweise gut für Plattformen zur Speicherung und Verteilung von Dokumenten und Software nutzen. Beispiele für Blob Storage Angebote sind Amazons Simple Storage Service (S3) aus dem Amazon Web Service (AWS) oder der Blob-Service des Microsoft Azure SDK.

Die Grundkonzepte eines Blob-Storage sind *Container*, die einen eindeutigen Namen besitzen müssen und die eigentlichen *Blobs* enthalten. Ein Benutzer kann mehrere Container anlegen. Ein Blob besitzt ebenfalls einen Namen und besteht aus den eigentlichen Objektdaten, im Prinzip einer Datei entsprechend, und zusätzlichen Metadaten. Zu den vordefinierten Metadaten zählen HTTP-Metadaten (ETag, Last-Modified, Content-Length, Content-Type, Content-Encoding, Content-Language etc.), es lassen sich aber auch benutzerdefinierte Metadaten aufnehmen. Da der Objektname das Trennsymbol „/“ enthalten kann, lassen sich Hierarchien im Sinne einer Verzeichnisstruktur ausdrücken.

Typische Operationen für einen Blob Storage sind:

- Create / Delete *Container*
- Write / Read / Delete *BLOB-Objekt*
- List *BLOB-Objekte*
- Get / Set *Metadata* / *Properties*

Die Adressierung eines Containers oder Blobs erfolgt über einen speziellen Uniform Resource Identificator (URI):

- `http://photos.s3.amazonaws.com/2009/Barbados/beach.jpg`: Bei diesem AWS-Beispiel ist `photos` der Container, der mit einem vordefinierten `s3.amazonaws.com` zu versehen ist; das Blob ist `2009/Barbados/beach.jpg`. `2009` und `Barbados` können hier als Unterverzeichnisse des Containers aufgefasst werden.
- `http://myaccount.blob.core.windows.net/?comp=list&maxresults=10&include=metadata`: Auch hier ist der Teil `blob.core.windows.net` für Microsoft Azure vorgegeben. `myaccount` ist die Benutzerkennung. Mit der URI werden alle Container inklusive (`include=`) ihrer Metadaten aufgelistet (`comp=list`), wobei das Ergebnis auf maximal 10 Sätze beschränkt wird (`maxresults=`).

Der Zugriff erfolgt in der Regel über zwei Protokolle, SOAP und REST (Representational State Transfer (RFC 2616) [Fi00]) über HTTP(S). REST ist ein HTTP-basiertes Protokoll, das HTTP-Operationen wie GET, DELETE oder PUT für Datenmanipulationen benutzt. Da der Zugriff aus einer Programmiersprache heraus recht aufwändig ist, werden entsprechende Programmierschnittstellen (wie JetS3t für Amazon S3) für die gängigen Programmiersprachen angeboten, mit denen sich die Benutzung der komplizierteren HTTP-Programmier-APIs vermeiden lassen.

Neben diesen grundsätzlichen Funktionalitäten bieten Blob Services weitere Funktionen, um die Performanz beim Zugriff zu erhöhen. So unterscheidet der Microsoft Azure Blob Storage zwischen Block und Page Blobs, deren Zugriffs-API einmal für sequenziellen Zugriff und ein anderes Mal für quasi zufällig platzierte Zugriffe optimiert sind. Hier muss der Anwender bzw. die Anwendung beim Anlegen entscheiden, welcher Blob-Typ angelegt werden soll.

2.2 Table Storage

Ein Table Storage gehört zur Gruppe der kürzlich aufgekommenen NoSQL-Datenbanken [NoSQL] und dient der Speicherung von strukturierten Daten - in einer relativ unstrukturierten, großen Tabelle, im Folgenden *BigTable* genannt. Die Tabellenstruktur muss dabei nicht im Sinne einer CREATE TABLE Anweisung vorgegeben werden, sondern ergibt sich dynamisch aus den zu speichernden Datensätzen. Jeder Datensatz besteht aus einem lokal eindeutigen Identifikator und einer Menge von Attributname/-wert-Paaren. Die Tabellenstruktur passt sich dann dynamisch den aktuellen Daten an. Beispiele sind Google's Implementierung einer BigTable [CDG+06], Amazon SimpleDB und der Azure Table Service. Abbildung 1 zeigt eine typische BigTable, die sowohl Kleidungsstücke als auch Auto- und Motoradteile enthält.

ID	Category	Subcat	Name	Color	Size	Make	Model
01	Clothes	Sweater	Cathair Sweater	Pink	S, M, L		
02	Clothes	Pants	Designer Jeans	Blue, Yellow, Pink	30x32,32x32		
03	Car Parts	Engine	Turbos			Audi	S4
04	Motorcycle Parts	Bodywork	Fender Eliminator	Blue			

Abildung 1: Beispiel eines BigTable

Für einen Table Storage ist charakteristisch, dass er keine vordefinierte Struktur besitzt. Zudem beziehen sich alle Operationen und Anfragen auf genau eine BigTable. Als unmittelbare Konsequenz sind keine Verbunde (Joins) zwischen Tabellen möglich. Mit Ausnahme von Bulk-Operationen stehen auch keine Transaktionen zur Verfügung; jede Einzeloperation ist atomar.

Der Zugriff erfolgt wie bei Blob Storages über SOAP und REST über HTTP(S) bzw. einfacher zu benutzende Programmierschnittstellen. Eine typische Anfrage als URI in Azure lautet

```
http://myaccount.table.core.windows.net/MyTable()?
$filter=(Model%20eq%"S4")%20and%20(Color%20eq%"Blue").
```

Hiermit werden aus MyTable alle Datensätze herausgefiltert, die der Bedingung `Model="S4" AND Color="Blue"` genügen. Azure bietet auch eine ADO.NET und eine LINQ-Schnittstelle (language-integrated queries), die aber nicht den vollen Funktionsumfang des Table Services bietet.

2.3 Datenbankserver

Die Rubrik der echten Datenbankserver stellt einen „virtuellen“ Datenbankserver für jeden Benutzer in der Cloud bereit. Der Begriff der Echtheit bezieht sich hierbei auf einen ähnlichen Funktionsumfang, wie man ihn von klassischen relationalen Datenbankservern erwarten würde. Der zugrunde liegende physische Datenbankserver kann prinzipiell mehrere Kunden bedienen, obwohl in der Regel jeder seinen eigenen Datenbankserver erhält, wobei aber auf einen physischen Rechner durchaus mehrere Datenbankserver laufen können. Beispiele sind Amazon RDS (Relational DB Service) mit einem MySQL-Server und Microsoft SQL Azure mit einem SQLServer.

Oracle bietet derzeit nur eine Lösung, ein Oracle-Image in der Amazon Cloud zu installieren[1]. Letzte Variante erfordert dann aber weiterhin eine Speicherplatz-Provisionierung und eine nunmehr Fern-Administration eines klassischen Datenbankservers. Insofern werden wir derartige Lösungen nicht in Betracht ziehen.

Während Blob und Table Storages ein SOAP-Interface bieten und mit dem REST-Protokoll neue Wege gehen, um den Zugriff der Daten zu ermöglichen, bieten die Datenbankserver die üblichen APIs wie JDBC, ADO.NET etc. an, die mit einer speziellen Datenbank-URL, die einen generierten Servernamen beinhaltet, zu versorgen sind. Beispiele für SQL Azure und Amazon RDS sind:

- `sqlcmd –S t17j2515ow.database.windows.net`
  `–U MyMasterUser@t17j2515ow –d MyDB`

  `t17j2515ow` ist ein vom Betreiber vergebener Name für den Datenbankserver, `MyDB` ist der Name Datenbank.

- `mysql –h myinstance.crwjauxgijdf.us-east-1.rds.amazonaws.com`
  `–P 3306 –u MyMasterUser –p`

  `myinstance` ist der Name der Datenbank, `crwjauxgijdf` der vom Betreiber vergebene Datenbankserver-Name.

Im Wesentlichen stellen die Angebote eine fremdverwaltete Instanz eines Datenbankservers dar.

Für Applikationen, die in der Cloud laufen und einen Datenbankserver benötigen, sind diese Lösungen wesentlich leichter als klassische Datenbankserver zu nutzen. Der Zugriff aus einer Cloud-Applikation heraus auf einen lokal betriebenen Datenbankserver ist zum einen aus Sicherheitsgründen (aufgrund einer quasi obligatorischen Firewall in großen Unternehmen) in der Regel nicht möglich. Zum anderen wird wegen der schwer nachzubildenden Symmetrie der Skalierungseigenschaften eine derartige Architektur problematisch sein: Die Anwendung in der Cloud wird viel elastischer skalieren als der lokal betriebene Datenbankserver, so dass Letzterer sich zum Flaschenhals entwickeln wird.

Ein weiterer Vorteil ist, dass der Zugriff auch aus Applikationen außerhalb der Cloud möglich ist, d.h., der Cloud-Datenbankserver ist prinzipiell von überall verfügbar. Hier

[1] http://aws.amazon.com/solutions/global-solution-providers/oracle/

kann wie beim Table- oder Blob-Store der Vorteil der ubiquitären Erreichbarkeit ausschlaggebend sein. Weitere Vorteile sind die mögliche Nutzung einer größeren Bandbreite der Internet Anbindung des Providers, die eventuell lokal nicht zur Verfügung steht, oder die teilweise besseren nicht-funktionalen Eigenschaften, die beim lokalen Setup unter Umständen mit Aufwand zu erzielen sind.

3 Einschätzung der Kategorien bzgl. Der Cloud Computing Vorteile

Im Folgenden beleuchten wir die Cloud Storage-Lösungen bezogen auf die eingangs diskutierten allgemeinen Vorteile von Cloud Computing.

3.1 Elastizität hinsichtlich Datenvolumen

Beim *Blob Storage* ist die Elastizität größtenteils gegeben. Obwohl diverse Limitationen wie die Anzahl der Container je Kunde oder eine maximale Blob-Größe bestehen, wirken sich diese nicht negativ auf die Erweiterbarkeit aus, da bei den gängigen Anbietern quasi beliebig viele Blobs je Container möglich sind.

Auch beim *Table Storage* gibt es Limitationen, die aber eine Datenbankobergrenze durch die Anzahl möglicher Tabellen und einer maximalen Tabellengröße (z.B. 100 x 10 GB = 1 TB bei Amazon) definieren. Insbesondere die maximale Tabellengröße ist für die Elastizität des Datenvolumens kritisch, da ihr Erreichen kontrolliert und in Applikationen entsprechend durch Anlegen einer neuen BigTable oder eine Umverteilung reagiert werden muss. Hierfür empfiehlt Amazon beispielsweise, an eine Vorabpartitionierung zu denken, wodurch zusätzlich eine bessere Skalierung der Zugriffe erzielt wird. Eine entsprechende Unterstützung ist z.B. beim Azure Table Service bereits gegeben.

Bei den *Datenbankservern* ist die Elastizität derzeit nicht im erwarteten Bereich, d.h. nicht so umgesetzt, wie man es bei den allgemein kommunizierten Vorteilen des Cloud Computings erwarten würde: So ist die Datenbankgröße beim Anlegen einer Datenbank festzulegen. Teilweise relativ kleine Obergrenzen, z.B. zurzeit von 50 GB bei SQL Azure als Business Edition, tun ihr Übriges, wobei sich diese Limitationen in naher Zukunft – wie in der Vergangenheit auch – noch ändern können. So ist die Grenze bei SQL Azure innerhalb von wenigen Monaten von 10 GB auf 50 GB in der Business Edition angewachsen, provisionierbar in 10 GB-Schritten.[2] Zu beachten ist, dass eine festgelegte Datenbankgröße provisioniert wird. Reicht die Datenbankgröße nicht aus, so kann eine neue Obergrenze über Administrationskommandos bis zur Maximalgröße vereinbart werden. Zurzeit sind unbegrenzt anforderbar die Anzahl die Datenbankserver und die Anzahl der Blobs; die Anzahl der Tabellen ist in der Regel limitiert.

[2] Angaben für Microsoft Azure abgerufen aus dem Internet unter http://www.microsoft.com/en-us/sqlazure/offers/default.aspx vom Sept. 2010

Eine generelle Einschränkung ist, dass besonders große Anzahl Instanzen (Datenbankserver, BigTables etc.) einer gesonderten Beantragung bedürfen und in der Regel nicht über eine Anbieter-Web-Anwendung automatisiert provisioniert werden. Letzten Endes tragen diese Mechanismen, ebenso wie einzelne voreinstellbare Provisionierungsfenster, zur Sicherheit des Kunden bei der Nutzung bei: Auf diese Weise soll verhindert werden, dass durch Irrtümer oder Programmierfehler ein zu großes Kontingent (automatisiert) angefordert wird, wodurch sonst hohe Kosten entstehen können.

3.2 Ausfallsicherheit und Hochverfügbarkeit

Die *Blob* und *Table Storages* replizieren die Daten in der Regel dreifach auf verschiedene Standorte. Zudem passiert ein Failover bei Ausfall eines der Speicher automatisch auf einen Anderen. Auch die *Datenbankserver* werden früher oder später automatisch replizieren, auch wenn es momentan noch nicht immer Standard ist, z.B. zurzeit bei Amazon RDS. Bei RDS kommt auch noch ein 4-stündiges Wartungsfenster je Woche hinzu, das festzulegen ist und in dem es z.B. bei Datenbank-Patchinstallationen durchaus zu einem Betriebsausfall kommen kann. Bedeutend ist, dass die Cloud-Datenbankserver „managed" Services sind, d.h., das Einspielen von Patches erfolgt automatisch, wodurch die Administration vereinfacht wird.

Ein kompletter Datenverlust ist nicht zu befürchten, da Backup/Restore-Möglichkeiten existieren. Zum Beispiel spezifiziert Amazon für den S3 Blob Storage eine Haltbarkeit („Durability") von „9-11"[3], was einen Verlust von einem aus 10.000 Blobs alle 10 Millionen Jahren gleichzusetzen ist. Gleichzeitig gibt es für die Kunden die Möglichkeit eine 9-4 Haltbarkeit für einen Datenbereich festzulegen (Reduced Redundancy Storage, RSS), welcher preiswerter angeboten wird.

Bezüglich der allgemeinen Verfügbarkeit haben Microsoft als auch Amazon gleiche Angebote: Grundsätzlich wird eine 99.9% Verfügbarkeit garantiert, die bei Verletzung in Form eines 10% Rabatts kompensiert wird. Sinkt die Verfügbarkeit unter 99%, wird durch beide Anbieter zurzeit ein Rabatt von 25% gewährt.[4] Hierbei ist zu beachten, dass nur auf gezahlte Beträge ein Rabatt eingeräumt wird. Es findet also kein Transfer von Beträgen vom Anbieter zum Kunden statt. Die Verfügbarkeit wird zudem durch Begrenzung der Dauer einer Abfrageausführung limitiert; wird ein Schwellwert überschritten, wird die Abfrageausführung abgebrochen. Zudem können erneute Anfragen verzögert werden, um Denial-of-Service-Angriffe entgegen zu wirken. Durch diesen Mechanismus kann schnell die durch einen (missglückten) Test erzeugte Last während der Entwicklung zu einer Sperrung der Datenbank führen, was während der Entwicklung zu beachten ist.

[3] 9-11 bedeutet eine Verfügbarkeit von 99.x % mit 2 Stellen vor und 9 Stellen hinter dem Komma.
[4] Angaben für Amazon abgerufen aus dem Internet unter http://aws.amazon.com/simpledb/ vom Sept. 2010

Eine Änderung der Instanzgröße ist jederzeit möglich, wobei eine Ressourcenerweiterung einen möglichen Betriebsausfall zur Folge hat. Ein Wechsel der Instanzgröße wird häufig den Umzug des Datenbankservers oder der Datenbank auf anderen Rechner bedeuten. In der Regel wird ein Wechsel der Instanzgrößen aber durch eine Hot-Switch/Staging Technik gelöst; es wird zunächst parallel eine neue Instanz gestartet, auf die umgeschaltet wird, um die Originale abzulösen.

3.3 Kosteneinsparung durch Elastizität

Die Kostenmodelle unterliegen derzeit noch starken Schwankungen und werden sich vermutlich auf einheitlichem Niveau einpegeln. Die Kosten, die einen erwarten, sind schwer überschaubar, da es neben den reinen Speicherkosten sehr viele Einflussfaktoren wie z.B. die CPU-Benutzung, Datentransferkosten von der (das Ergebnis) und in die Cloud (die Anfrage) gibt und mitunter XML-Daten transportiert werden, deren Größe nicht einfach abschätzbar ist. Zudem gibt es unterschiedliche Preise für die jeweils angebotenen Regionen (z.B. US East/West, EU)

Bei den *Blob Storages* treten bereits recht hohe Speicherkosten von 150$ für 1 TB pro Monat bei Amazon S3 auf, während eine eigene TB-Festplatte durchaus bereits für 100$ zu haben ist. Ein fairer Vergleich muss allerdings auch die Replikation und die eingesparte Administration (Restarts, Updates, Patches, etc.) berücksichtigen. Hinzu kommen noch die Transferkosten. Die Kosten skalieren mit dem Datenvolumen und den anderen Faktoren, wobei die relativen Kosten mit wachsendem Verbrauch abnehmen, z.B. von anfangs 15Ct für 1 GB auf 5,5Ct bei einer Belegung von über 5000 TB.

Bei den *Table Storages* sind der Speicherplatz und weiterer Ressourcenverbrauch häufig bis zu einer gewissen Größe frei, erst danach fallen Kosten an. Das Kostenmodell ist insofern für Privatpersonen und Start-ups interessant, als bei kleinen Datenmengen (häufig kleiner als 1GB) erst einmal keine Kosten anfallen, weder Fixkosten noch Verbrauchskosten. Die Nutzungskosten orientieren sich wieder an vielen Faktoren und wachsen mit dem Datenvolumen und den Zugriffen.

Andere Kostenmodelle gibt es für die *Datenbankserver*: Hier fallen Kosten bereits beim Anlegen einer Datenbank an, wobei in der Regel die Datenbankkonfiguration (Small, Large, bis hin zu Quadruple Extra Large bei Amazon RDS), die Speicherprovisionierung (anzugeben beim Anlegen der Datenbank), also *nicht* der aktuell belegte Speicher, sowie der Datentransfer einfließen. Der festgelegte Benutzungsrahmen ist zu bezahlen. Auch ist das Volumen von Backup-Daten (z.B. durch eine angegebene Backup-Periode gesteuert) zu berücksichtigen.

Bei SQL Azure ist das Preismodell recht einfach, da hier ein fester Preis für die aktuelle Datenbankgröße bis zum nächsten Provisionierungsschritt zu bezahlen ist. Dies bedeutet bei den zurzeit eingerichteten 10GB Schritten in der Business Edition, dass eine 13GB Datenbank zum Preis von 20GB abgerechnet wird, auch wenn die provisionierte Maximalgröße („cap“) 50GB beträgt. Bei der von [KKL10] durchgeführten Untersuchung führte dieser Punkt zu den günstigen Kosten. Auch bei den Cloud-Datenbankservern muss ein fairer Preisvergleich mit selbstadministrierten

Datenbankservern die bereitgestellte Ausfallsicherheit wie auch eingesparten Administrationskosten berücksichtigen, da gerade das Aufsetzen und Administrieren eines Datenbank-Clustern eine aufwändige Angelegenheit ist.

3.4 Administration

Beim *Blob Storage* fallen keine eigenen Administrationskosten an, sieht man einmal von der Registrierung und dem Anlegen einer Benutzerkennung ab.

Auch bei den *Table Storages* entstehen auf dem ersten Blick keine Administrationskosten. Typische DBA-Aufgaben werden vom Anbieter übernommen und können auch nicht selber wahrgenommen werden, z.B. übernimmt der Azure Table Service eine automatische Indexierung der BigTables, wobei sich sofort die Frage stellt, ob der Automatismus wirklich gut genug für eine optimale Performanz ist. Problematisch ist die Größenbeschränkung auf Tabellenebene, die eine Überwachung der Tabellengröße erfordert wie auch dann eine adäquate Reaktion beim Erreichen des Maximums.

Bei den *Datenbankservern* wird die Hardwareausstattung der Datenbankserver (Anzahl CPUs, Anzahl Core's, Platten etc.) gemäß der gewählten Instanzgröße bereitgestellt. Eine Administration entfällt also weitgehend bzw. ist gar nicht möglich. Eine manuelle Überwachung wird auch hier nötig, um bei Erreichen der Maximalgrößen reagieren zu können. Beispielsweise kann bei einer auftretenden Exception aufgrund einer erreichten Quotagrenze ein Administrationkommando abgesetzt werden, durch das die Maximalgrenze verschoben wird. Ebenso ist die Performanz zu kontrollieren, um ggf. in eine andere Instanzkategorie zu wechseln. Performanzkritische DBA-Aktionen wie das Erzeugen von Indexen haben weiterhin Bestand. Als Fazit lässt sich feststellen, dass die DBA-Kosten nicht entfallen. Insbesondere mangelt es an einer automatischen Kontrolle der Speicherbelegung bzw. automatischen Erweiterung bei Erreichen der Maximalgröße.

4. Allgemeine Punkte

4.1 Skalierbarkeit

Bei *Blob* und *Table Storages* gehen die Requests über das REST- oder SOAP-Protokoll an einen Web-Server, d.h., die Skalierung der Zugriffe liegt als Erstes in der Verantwortung des Web-Servers und wird von dort aus auf das Speichersystem verteilt. Eine implizite Replikation der Daten auf in der Regel drei physikalische Standorte trägt zudem zur Skalierbarkeit bei. Der Server selbst ist nicht zu beeinflussen.

Ein selbstverwalteter *Datenbankserver* wird ohne Umweg über einen Web-Server angesprochen. Insofern ist die Skalierbarkeit des Datenbankservers ausschlaggebend. Der Datenbankserver lässt sich in der Regel bzgl. des internen Caches, der Anzahl erlaubter Transaktionen, der benutzten Platten etc. konfigurieren bzw. mit parallelen Platten oder RAID-Systemen zu versorgen, um einen optimalen Betrieb und eine hohe

Skalierbarkeit zu gewährleisten. Amazon RDS und Microsoft SQL Azure sehen keine derartige Konfiguration vor, sondern bieten nur vorgefertigte Instanzkategorien wie Small, Large, XL etc. an, die beim Anlegen der Datenbank auszuwählen sind. Hinter jeder Kategorie verbirgt sich eine entsprechende Rechnerausstattung, z.B. 2 CPU-Kerne, 3.5 GB Hauptspeicher und 500 GB Plattenspeicher bei der SQL Azure Medium-Konfiguration. Die Skalierbarkeit erfolgt somit nur über die Datenbankserver-Konfiguration, insbesondere der maximale Anzahl erlaubter Verbindungen der jeweiligen Instanzgröße. Grundsätzlich ist keine weitergehende Einflussnahme auf die Hardware wie das Hinzuschalten weiterer Platten oder Disk Arrays möglich.

Eine nachträgliche Änderung ist als Reaktion auf beobachtete Skalierungs- oder Performanzprobleme möglich, aber auch hier stellt sich die Frage sowohl nach der Zeit zwischen Anforderung und Umsetzung bei Ressourcenerweiterung als auch einem möglichen Betriebsausfall.

4.2 Performanz

Ein signifikanter Unterschied zum klassischen Datenbankserver dürfte durch den Zugriff zum Cloud Storage an sich entstehen. Ein Cloud Storage ist im Internet verfügbar und ein darauf gerichteter Zugriff geht den bekannten Weg über Router und Internet-Anbieter. Hierbei sind deutliche Unterschiede in Latenz und Durchsatz im Vergleich zum lokal aufgestellten Datenbankserver zu erwarten. An dieser Stelle muss die Art der Anwendung entscheiden, ob Einschränkungen bzgl. der Latenz und der Bandbreite beim Zugriff in die Cloud einen Einsatz erlauben. Dieser Nachteil ist nicht gültig, wenn die Anwendung, die auf den Cloud Storage zugreift, selbst in der Cloud aufgestellt wird.

Üblicherweise hat der Datenbankadministrator mit dem Anlegen von Indexen einen direkten Einfluss auf die Performanz beim Datenzugriff auf ein Datenbanksystem. Die Implementierungen der Table Storages der unterschiedlichen Anbieter sehen im Gegensatz dazu vor, dass der Anwender nicht „mit der Erstellung von Indexen belastet wird“, wie es Amazon in der entsprechenden Dokumentation formuliert. Auch beim Azure Table Service wird nur das Vorhandensein fest eingeplanter Indexe dokumentiert, aber keine Möglichkeit geboten, eigene Indexe anzulegen. Dies bedeutet nicht, dass die Table Storages im Zugriff per se langsamer sind; es wird lediglich dem Benutzer die Möglichkeit genommen, direkt Einfluss auf die Datenorganisation zu nehmen. Abhilfe kann hier eine frühzeitig festgelegte Partitionierung schaffen, die bei Azure recht einfach über einen Partition Key steuerbar ist.

Weiterhin ist zu beachten, dass die Anbieter von Cloud Storages ein sogenanntes Throttling einsetzen um eine unausgewogene Nutzung der Ressourcen zu vermeiden. Im Extremfall dient Throttling dazu, eine Denial-of-Service-Attacke zu verhindern, im Normalfall wird durch Throttling eine Abgrenzung zwischen verschiedenen Produktklassen bzw. Service Level Agreements (SLA) hergestellt. Beispielsweise ist bei einer kostenlosen Nutzung der Google App Engine der Datendurchsatz auf knapp ein 1MB/sec beschränkt, wohingegen im Bezahlmodus der maximale Datendurchsatz über 150MB/sec betragen kann. Dies bedeutet, dass bei einer intensiven Nutzung von Cloud Storage die Throttling Einstellungen des Anbieters relevant sind.

Bei den Datenbankservern wird die Performanz, wie bereits erwähnt, indirekt über die Instanzgröße gesteuert. Bei Problemen mit der Verarbeitungsgeschwindigkeit kann nur durch einen Wechsel der Instanzkategorie die Anzahl der CPUs etc. erhöht werden. Zu beachten ist auch, dass man nur einen *virtuellen* Datenbankserver erhält. Es ist also nicht gewährleistet, dass einem ein *physischer* Datenbankserver allein zur Verfügung stehen wird; prinzipiell kann der Datenbankserver gleichzeitig mehreren Kunden bzw. virtuellen Instanzen zur Verfügung stehen. Häufig wird ein physikalischer Rechner mehrere Datenbankserver, bis zu vier bei SQL Azure, aufnehmen; eine gegenseitige Beeinflussung scheint vorprogrammiert und hat bei klassischen Datenbankservern massiv zu Performanzproblemen geführt. Aus diesem Grund gibt es vermutlich nur die Zusicherung der Verfügbarkeit per SLA, aber kein zugesagtes Antwortverhalten.

4.3 Standardisierung

Wie bereits erläutert, ist ein Table Storage nicht mit einem klassischen Datenbankserver zu vergleichen [Sh07], da sich alle Operationen, auch lesende Anfragen, auf genau eine BigTable beziehen, Beziehungen zwischen Tabellen sind folglich nicht als Join in der Abfragesprache herzustellen. Die Gründe hierfür sind zweierlei: Zum einen wird durch diese Vereinfachung eine optimale Skalierbarkeit und Partitionierung der Daten und damit eine optimale Performanz ermöglicht. Zum anderen orientieren sich die Table Storages an einem Speicher für Objekte, so dass Abfragesprachen unabhängig vom relationalen Modell gestaltet sind. Dies wird durch eine Übersicht der Abfragesprachen deutlich:

Google App Engine:	JDO Query Language (JDOQL)
Microsoft Azure Cloud:	Operatoren und Ausrücke via REST oder LINQ
Amazon SimpleDB:	SELECT Ausdruck mit Filter und Sortierung, keine JOIN Ausdrücke

Die Beschaffenheit der Abfragesprachen zeigt, dass bei der Benutzung eines Table Storages Beziehungen zwischen Datenobjekten auf der Ebene der Anwendungslogik aufgelöst werden müssen. Weiterhin wird deutlich, dass ein Äquivalent zum weit verbreiteten SQL bei Table Storages noch nicht vorhanden ist. Gleiches gilt auch, wenn man beispielsweise die Syntax einer via REST formulierten Abfrage an SimpleDB mit einer Anfrage an den Azure Table Storage vergleicht. Bei SimpleDB wird ein SELECT-Ausdruck als Parameter einer REST-Anfrage verpackt. Bei Microsoft Azure wird der Ausdruck auf die Schlüssel-Wert-Paare einer REST-Anfrage abgebildet. Hierbei muss allerdings angemerkt werden, dass in den meisten Fällen ein clientseitiger Proxy den Zusammenbau der REST-Anfrage erledigt. Dennoch wäre für die Vereinheitlichung von unterschiedlichen Client-Proxies eine Standardisierung für REST-Aufrufe für die unterschiedlichen Table Storage-Produkte von Vorteil.

Weiterhin ist eine Vereinheitlichung der Schnittstelle in einer Hochsprache für Table Storages nicht so zu finden, wie es mit klassischen relationalen Datenbankservern mit JDBC oder ODBC der Fall ist. Innerhalb der Java-Welt ließe sich JDO als objektorientiertes Mapping zum Table Storage vorstellen. Ein Standard wie JPA, also eine API, die sich nur auf Table Storages beschränkt, ist allerdings noch nicht

vorhanden. Gleichwohl gibt es erste Ansätze wie SimpleJPA [SJPA] oder SimpleORM [SORM] für einzelne Produkte, die in diese Richtung gehen.

Bei den Storage Produkten, die einen SQL Server zur Verfügung stellen, handelt es sich um Weiterentwicklungen auf Basis bereits bekannter Produkte (Microsoft SQL Server bzw. MySQL bei Amazon RDS). In diesem Fall sind die üblichen Standards vorhanden und ähnliche Feinheiten bezüglich Einrichten, Verhalten und Funktionalität zu erwarten, wie sie auch die bisherigen Produkte aufgezeigt haben.

4.4 Weitere Besonderheiten

Für die Migration bestehender Applikationen ist es bedeutend, dass es bei Amazon SimpleDB und bei Amazon S3 nur einen Datentyp gibt (nur Strings im UTF-8 encoding) und dass alle Attributwerte kleiner als 1KB sein müssen. Die Konsequenzen können dann bei einer Migration von einem klassischen Datenbankserver zu Amazon SimpleDB erheblich ausfallen, wenn Anwendungslogik, die auf Wertevergleich von Zahlen basiert oder Blobs verarbeitet, auf die Verarbeitung von diesen Strings geändert werden muss.

Google's BigTable erlaubt die Speicherung auf Basis von JSON-Definitionen (JavaScript Object Notation). Dies ist ebenso das Format, welches von den Softwareprodukten CouchOne oder MongoDB über deren APIs verarbeitet wird. Da JSON nicht nur Strings, sondern auch Zahlen oder boolesche Werte kennt, ist hier die Ähnlichkeit zu einem klassischen Datenbankserver viel größer als bei SimpleDB von Amazon. Der Table Service aus der Microsoft Welt verarbeitet die Datentypen der ADO.NET Data Services Spezifikation.

Bei SQL Azure sind wenige nennenswerte Unterschiede zu dem bisherigen SQL Server Produkt zu nennen, die nach und nach behoben werden. Erst kürzlich hat Microsoft die Unterstützung von hierarchischen Daten in SQL Azure als Datentyp nachgezogen und eine Unterstützung für Datenverschlüsselung angekündigt. Einige Datentypen früherer SQL Server Versionen werden wie von der aktuellen Version auch nicht von SQL Azure unterstützt (text, image, ntext). Da der Amazon RDS im Prinzip eine Version von MySQL darstellt (zurzeit MySQL 5.1.50), ist ein Transfer der Daten von einem MySQL Server zu Amazon RDS problemlos möglich.

Neben den genannten Punkten muss noch beachtet werden, dass die Testbarkeit durch den Einsatz von Cloud Storage leidet und im Normalfall Kosten verursacht. An dieser Stelle bieten die Anbieter teilweise Arbeitsumgebungen mit lokaler Datenbank, um diesen Nachteil zu kompensieren, zum Beispiel Microsoft mit der Developer AppFabric.

Schließlich können beim praktischen Einsatz der Cloud Storage Lösungen noch unerwartete Überraschungen auftreten: So kann es beim Zugang aus dem Firmennetz auf einen Cloud-Datenbankserver Probleme mit dem eigenen Proxy geben, der keine Requests über spezielle Ports hinauslässt. Diese können zum Beispiel von einem Administrationswerkzeug für SQL Azure oder Amazon RDS stammen. Der Zugriff auf Blob und Table Storages mit SOAP oder REST ist hingegen kein Problem.

5 Zusammenfassung und Ausblick

In dieser Übersicht wurde aufgezeigt, dass Eigenschaften, die man seitens klassischer Datenbankserver kennt, nicht per se im Bereich der Cloud Storages vorhanden sind. Beispiele hierzu lassen sich auch in [Sh07] nachlesen. Die Eigenschaften unterscheiden sich hinsichtlich der Produktabgrenzungen und hinsichtlich der Anbieter. Standardisierungen in der Benutzung und der Eigenschaften bleiben insbesondere für die neue Technologie der NoSQL-Datenbanken noch aus. Für den Anwender bedeutet dies, dass Kostenmodelle und Performanz der unterschiedlichen Produkte gegenübergestellt werden müssen, um eine Entscheidung zur Nutzung eines einzelnen Produktes treffen zu können.

Auffällig ist auch, dass bei technischer Betrachtung der Möglichkeiten die grundlegenden Vorteile von Cloud Computing, Elastizität, Skalierbarkeit und die Möglichkeit der Kostensenkung im Einzelfall unter Umständen nicht erreicht werden:

- Die Skalierbarkeit ist wie auch die Elastizität durch den Nutzungsrahmen (z.B. die maximale Datenbank- oder Tabellengröße, oder die Anzahl zur Verfügung stehender Datenbankverbindungen) der konkreten Produkte beschränkt. Bei einigen Nutzungsmodellen gibt es klare Obergrenzen, deren Überschreitung eines administrativen Aktes bedarf, sofern man sich nicht frühzeitig aus Vorsicht auf einen zu hohen und damit kostspieligeren Nutzungsrahmen festlegen möchte.
- Die Kosten können bei häufigen Datentransfer zur und von der Cloud u.U. größer ausfallen als bei Applikationen, die in der Cloud laufen. Hierbei können Server mit großen lokalen Festplatten wiederum eine preiswerte Alternative darstellen. Zwar bietet Amazon mit dem Reduced Redundancy Storage eine günstige Alternative an, die wiederum hinsichtlich der Zuverlässigkeit stark eingeschränkt ist. Letzten Endes entscheidet die Art Anwendung über den optimalen Einsatz von Storage Lösungen.
- Die Ausfallsicherheit ist nur in dem Umfang der Produktausgestaltung gegeben. Unter Umständen sind Wartungsfenster zu beachten, die die Verfügbarkeit im Vergleich zu lokal betriebenen Lösungen einschränken.
- Der administrative Aufwand kann in der Tat geringer ausfallen, allerdings auch unter dem Verlust der Kontroll- und Gestaltungsmöglichkeit. Hier müssen die Anforderungen der Anwendungen bzw. die erzielte Performanz entscheiden, ob die Einschränkungen nicht zu Nachteilen führen.

Dies Punkte zeigen deutlich, dass in vielen Bereichen noch Nachbesserungen erforderlich sind: Die in der Breite suggerierten Eigenschaften des Cloud Computing sind technisch bei den momentanen Cloud Storage Angeboten nicht konsequent umgesetzt.

5.1 Ausblick

Die genannten Arten von Cloud Storage und die verfügbare Produktvielfalt von Blob über Table Storages hin zu Datenbankservern machen insgesamt deutlich, dass viele

Faktoren die Auswahl eines Produktes beeinflussen. Betrachtet man dazu die Freiheitsgrade bezüglich der Aufstellung der eigenen Anwendung, wird klar, dass die Nutzung von Cloud Storage zu unterschiedlichen Formen der Anwendungsarchitektur führen kann. Insbesondere eine Partitionierung der Daten sollte aufgrund existierender Obergrenzen und Skalierungsgründen frühzeitig berücksichtigt werden. Mangelnde Erfahrungen mit Frameworks wie Hibernate oder das Java Persistence API (JPA) machen sich momentan noch negativ bemerkbar. Zudem wird insbesondere der Kostenfaktor die technologische Auswahl wie auch die Architektur beeinflussen. Auch gesetzliche Aspekte sind sicherlich relevant, wie z.B. wo die Daten letztendlich liegen und welchen lokalen Gesetzgebungen (z.B. der PATRIOT Act der U.S.A.) sie unterliegen.

Ein wichtiges Thema ist weiterhin die Migration bestehender Datenbanken und/oder Applikationen in die Cloud. Hier stellt sich heraus, dass Table Storages mit einer Portabilität von Anwendungen schlecht verträglich sind. Zum einen bedeutet die Migration einer bestehenden relationalen Datenbank in einen Table Storage einen erheblichen Aufwand, nicht nur bei der Datenkonvertierung sondern auch bei den Zugriffen. Zum anderen wird ein späterer Wechsel des Betreibers nicht einfach werden (Datentypen von Amazon SimpleDB gegenüber denen vom Azure Table Service), und insbesondere wird der Rückweg zu einem relationalen Datenbankserver erschwert. Die Migration zwischen klassischen Datenbankservern und Cloud-Datenbankservern gestaltet sich aufgrund ihrer relativen Verwandtschaft einfacher, auch wenn ihr Betrieb in der Cloud Einschränkungen unterliegt. Nichtsdestoweniger bleiben die APIs gleich.

Und schließlich blieben die administrativen Probleme in Unternehmen bisher unerwähnt: Als Privatperson ist es recht einfach die Kreditkarte anzugeben, die dann mit den Verbrauchskosten belastet wird. In vielen Firmen muss allerdings vor der Nutzung ein Bestellvorgang initiiert werden, der einen Preis benennt. Wird dieser Preis dann aufgrund des Pay-as-you-go überschritten, so können die zusätzlichen Kosten nicht mehr abgerechnet werden. Hier sind sicherlich noch Nachbesserungen in den Tarifoptionen notwendig, um den breiten Einsatz in Unternehmen zu erleichtern.

Literaturverzeichnis

[AFG+09] Armbrust, M., A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia: Above the Clouds: A Berkeley View of Cloud Computing. UC Berkeley Reliable Adaptive Distributed Systems Laboratory, Technical Report EECS-2009-28, February 2009 available online at http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[CDG+06] F. Chang, J. Dean, S. Ghemawat, et al.: Bigtable: A Distributed Storage System for Structured Data. In OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November, 2006

[CH09] D. Chappell: Introducing Windows Azure. White paper, Microsoft and Chappell Associates, December 2009. Available online (25 pages).

[Fi00] R. T. Fielding: REST: Architectural Styles and the Design of Networkbased Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.

[KKL10] D. Kossmann, T. Kraska, S. Loesing: An Evaluation of Alternative Architectures for Transaction Processing in the Cloud. ACM SIGMOD 2010

[LMM09] J. Lee, G. Malcolm, and A. Matthews: Overview of Microsoft SQL Azure Database, Microsoft Technical Whitepaper 2009.

[Mongo] Mongodb - scalable, high-performance, open source, document-oriented database. Available from http://www.mongodb.org/.

[MS1] Microsoft SQL Azure Database http://www.microsoft.com/azure/data.mspx

[NoSQL] *http://nosql-database.org*

[Sh07] N. Shalom: Amazon SimpleDB is not a database! http://natishalom.typepad.com/nati_shaloms_blog/2007/12/amazon-simpledb.html

[SJPA] SimpleJPA, version 1.0 project homepage, accessed Feb 2010 at http://code.google.com/p/simplejpa/

[SORM] SimpleORM, version 3.* project homepage, accessed Feb 2010 at http://www.simpleorm.org/