Technologisches Gewerbemuseum

Systemtechnik-Maturaausarbeitung

Systemintegration und Infrastruktur

Autor:
Burkhard Hampl

30. Mai 2016

Inhaltsverzeichnis

1 Abstract

In den letzten Jahren ist eine neuer Trend aufgekommen, das Cloud-Computing. Es gibt viele Plattformen, welche auf dem Prinzip des Cloud-Computings aufbauen. Viele Unternehmen setzten heutzutage schon auf Cloud-Computing und wahrscheinlich verwendet jeder Täglich die “Cloud” ob direkt oder indirekt. Auch die Technologie, welche hinter solchen Plattformen steckt, entwickelt sich sehr schnell weiter und es gibt immer neuerer Ansätze. Aber auch die Clients, also die Geräte die solche Plattformen verwenden, entwickeln sich andauernd weiter und es gibt stetig Erneuerungen. Es gibt aber viele Sachen die man bei Cloud-Computing-Plattformen beachten sollte und verbessern kann, vor allem was die Hochverfügbarkeit und Sicherheit einer solchen Plattform angeht.

2 Cloud Computing und Internet of Things

Cloud-Computing ist heutzutage sehr verbreitet und erfreut sich über sehr große Beliebtheit. Es gibt viele Ansätze, die umgesetzt werden können und es haben sich schon viele Anwendungsgebiete und Bereiche gefunden, in denen sich Cloud-Computing durchsetzten konnte.

2.1 Was ist eine Cloud?

Cloud-Computing kommt aus dem Englischen und bedeutet "Rechnerwolke". Das Prinzip von Cloud-Computing ist die Auslagerung von Diensten/Ressourcen ins Internet. [1]

Zur Zeit gibt es viele Anbieter von Cloud-Computing und viele Unternehmen setzen auf Cloud-Computing. Bei Cloud-Computing werden allgemein Anwendungen, Daten und Rechenvorgänge ins Web ausgelagert, was viele Vorteile mit sich bringt. Es ist einfach möglich gemeinsam an Dokumenten, da die Daten und Dienste Zentral an einer Stelle liegen und dadurch auch die Synchronisation dieser gegeben ist, zu arbeiten. Auch ist es möglich Daten an Externe frei zu geben und so einen externen Zugriff zu ermöglichen. Der Datenspeicher in der Cloud ist zwar auch begrenzt, aber so groß, dass es fast nicht möglich ist, diesen auszufüllen und für jeden Anwendungszweck gibt es eine entsprechende Speichergröße. Rechenvorgänge können durch eine große Anzahl an Virtuellen Maschinen unterstützt werden und die Abrechnung erfolgt auf Stunden, d.h. man bezahlt nur die Leistung, die mach auch Verbraucht, was Kosten spart. [1]
Eine mögliche Definition wäre: "Cloud-Computing ist die Auslagerung von Anwendungen, Daten und Rechenvorgängen in das Internet" [1]

Gemeinsame Benutzung von Ressourcen Mit Cloud-Computing ist sehr einfach für eine Anwendung auf gemeinsame Ressourcen zuzugreifen und zu verwenden, dadurch können verschiedene Benutzer gleichzeitig diese Ressourcen verwenden. Für die Sicherheit und Absicherung der Datensätze und Ressourcen ist die Cloud-Computing-Plattform und Anwendung verantwortlich. [1]

Einfache Verwaltung Durch die Auslagerung muss sich ein Unternehmen nicht um eventuelle Administrationen kümmern, welche z.B. für eine "traditionellen Systemumgebung" benötigt werden (z.B. Load Balancing oder Serverwartung). [1]

On demand Mit "On demand" ist gemeint, dass Cloud-Computing-Plattformen sofort verfügbar und einsetzbar sind, ohne lange auf die Bereitstellung und auf eventuelle (komplexe) Verträge zu warten, was im Gegensatz zum "konventionellen On-Promis-Hostings" steht. Auch wenn z.B. mehr Speicher für eine Anwendung benötigt wird, kann dieser in wenigen Minuten zur Verfügung gestellt werden. [1]

2.2 Offenheit der Plattformen

Es gibt verschiedene Offenheiten der Plattformen, unter denen unterschieden wird. [1]

2.2.1 Public Cloud

Der Begriff "Public Cloud" umfasst jene Cloud-Computing-Plattformen, welche für jeden von überall Zugreifbar ist. Dabei werden Ressourcen und Dienste zwischen vielen verschiedenen Kunden geteilt. [1]

2.2.2 Private Cloud

Die “Private Cloud” ist genau das Gegenteil der “Public Cloud” (2.2.1). Die “Private Cloud” ist Typischerweise nur innerhalb eines Unternehmens zugreifbar, dadurch kann eine größere Datensicherheit und Privatsphäre gewährleistet werden. Der große Unterschied dabei zu einem “konventionellen On-Promis-Hostings”, welcher in einem Unternehmen gehostet wird, sind die Schnittstellen, welche nach außen zur Verfügung gestellt werden. Mittels dieser, kann dann von externen Unternehmen auf interne Daten zugegriffen werden. Auch kann so eine Plattform besser auf das Unternehmen angepasst werden und so mehr auf die Bedürfnisse eingegangen werden. Im Prinzip ist “On-promise-Hosting” das gleiche, deswegen muss sich ein Unternehmen auch um die ganze Hardware und Software selbst kümmern und auch Personal dafür abstellen. [1]

2.2.3 Hybrid Cloud

Um die Vorteile von “Public” (2.2.1) und “Private Cloud” (2.2.2) zu vereinen gibt es die “Hybrid Cloud”. Dabei kann mittels einer Middleware ein “interner Dienst” für die Kunden von außen zugreifen. Dadurch ist es dem Kunden nicht möglich auf unternehmensinterne Daten zuzugreifen. [1]

2.3 Ansätze des Cloud-Computings

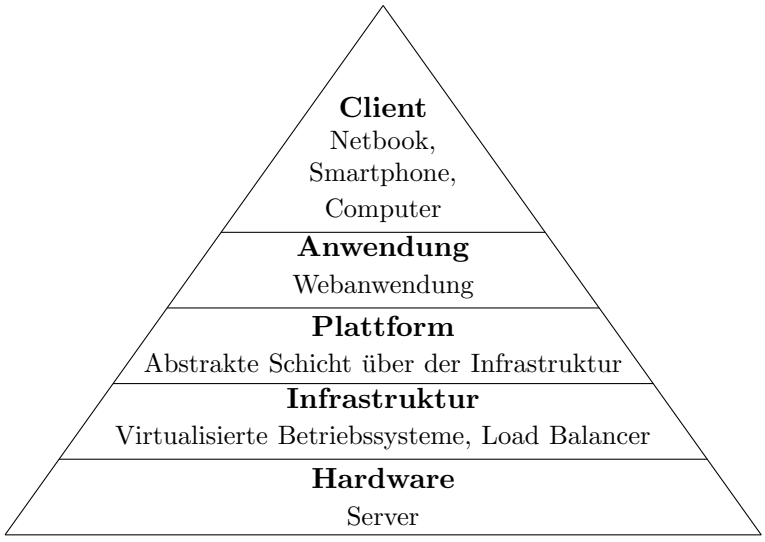

Abbildung 1: Ebenen des Cloud-Computings [1]

2.3.1 Infrastructure as a Service

“Infrastructure as a Service” (kurz IaaS) ist die unterste Schicht des Cloud-Computings, dabei handelt es sich um die Infrastruktur des Cloud-Computing-Anbieters. Der Benutzer kann dann auf dieser Infrastruktur seine Projekte verwirklichen und sie so verwenden wie er will. Dabei werden oft (virtuelle) Betriebssysteme und Sachen wie Lastverteilung oder Backup vom Anbieter zur Verfügung gestellt. Dadurch dann der Kunde sich seine Umgebung und Plattform ganz individuell Zusammenstellen, was aber natürlich mehr Administrationsaufwand mit sich bringt. Wenn eventuelle Anwendungen auf der Infrastruktur ausgeführt werden soll, dass muss sich der Kunde selber um die Laufzeitumgebungen kümmern. [1]

2.3.2 Platform as a Service

"Platform as a Service" (kurz PaaS) ist die Schicht über "Infrastructure as a Service" (2.3.1), bei der zwar nicht mehr auf das Betriebssystem zugegriffen werden kann, aber der Kunde sich nicht mehr um "Low-Level-Administration" kümmern muss. Auch bieten viele Anbieter "Sandboxen mit High-Level APIs" zur Anwendungsentwicklung und verschiedene andere APIs z.B. zur Verwaltung von Plattformen oder Datenspeicher. Der größte Einsatzzweck für so eine Plattform sind Webanwendungen, welche auch von rechenaufwendigen Aufgaben getrennt werden können. Auch wenn Aufgaben wie Wartung und Installation für solch eine Plattform wegfallen, verliert man als Kunde ein bisschen die Flexibilität gegenüber "Infrastructure as a Service" (2.3.1). [1]

2.3.3 Software as a Service

"Software as a Service" (kurz SaaS) ist die Schicht über "Platform as a Service" (2.3.2), die eher für den Endanwender gedacht ist. Dabei wird auf eine fertige Webanwendung zugegriffen, welche direkt verwendet werden kann. Dabei muss sich nur der Anbieter um Aktualisierungen und Wartung kümmern, dabei geht aber natürlich die Flexibilität der darunterliegenden Schichten verloren. Da sich so eine Anwendung für den Kunden "zentral" befindet, kann er von einem beliebigen Device drauf Zugriffen, was aber wiederum Abhängigkeiten mit dem Internet schafft. [1]

2.4 Projektumfeld

Projektumfeld: "UN Bojen-Netz"
Cloud-Computing macht in dem Sinn bei einem solchen Projekt Sinn, da die Auswertung hochverfügbaren und zentral zugreifbar sein soll. Am meisten Sinn würde Cloud-Computing-Plattformen mit "Platform as a Service" machen, da so ein einfaches Ausliefern und Verwalten eine Anwendung möglich wäre. Es gibt viele Anbieter einer solchen Plattform, sie haben aber alle viele Gemeinsamkeiten, was Lastverteilung und der gleichen angeht. Die wichtigsten Faktoren einer Evaluation einer solchen Plattform würde vor allem Preis und die auszuliefernde Anwendung sein, da sich da die Anbieter am meisten Unterscheiden.

3 Automatisierung, Regelung und Steuerung

3.1 Projektumfeld

Projektumfeld: “Loxone”

4 Security, Safety, Availability

Eine Cloud-Computing-Plattform kann nicht ohne richtiger Hochverfügbarkeit funktionieren. Man braucht sich nur vorstellen, dass ein Dienst wie "Dropbox" oder "Google Drive" nicht verfügbar ist oder gar Daten verloren gehen. Um so was zu verhindern, wird in vielen Bereichen Hochverfügbarkeit eingesetzt, wo es ein immer verfügbares System benötigt.

4.1 Grundlagen Hochverfügbarkeit

Hochverfügbarkeit (kurz HA) ist die Fähigkeit eines Systems, auch bei einem Fehler einer Komponente den Betrieb aufrecht zu halten. Ein Beispiel für so einen Fehler wäre ein Ausfall eines Servers oder einer Festplatte in einem Server. Die Wahrscheinlichkeit wird dabei in Prozent angeben, z.B. 99,99 %. Mittels dieser Prozentzahl kann dann die maximale Ausfallszeit in einer gewissen Zeitspanne berechnet werden. [2]

Verfügbarkeit	**Auszeit pro Monat** [a]	**Auszeit pro Jahr** [b]
99,0 %	7,2 h	87,66 h
99,5 %	3,6 h	43,83 h
99,9 %	0,72 h	8,77 h
99,99 %	4,32 min	52,59 min
99,999 %	0,43 min	5,26 min

[a] Ein Monat wird hier der Einfachheit halber mit 30 Tagen angesetzt.

[b] Ein Jahr wird hier mit 365,25 Tagen angesetzt.

Tabelle 1: Verfügbarkeit und Auszeit pro Monat bzw. Jahr [2]

4.2 Hochverfügbare Architekturen

Es gibt verschiedene hochverfügbare Architekturen, welche eingesetzt werden können. [2]

4.2.1 Cold Standby

Der "Cold Standby" ist ein redundantes System, das neben dem Produktivsystem ausgeschaltet ruht. Wenn ein Fehler des Produktivsystemes entstehen sollte, muss diese Ersatzsystem manuell hochgefahren werde. [2]

Vorteile

- Keine Komplexitätserhöhung
- Geringe Kosten

[2]

Nachteile

- Ersatzsystem muss ständig aktuell gehalten werden
- Ersatzsystem muss manuell aktiviert werden

- Daten müssen ggf. migriert werden

[2]

4.2.2 Hot Standby

Der "Hot Standby" ist ähnlich wie der "Cold Standby" (4.2.1), nur das das Ersatzsystem im hochgefahrenen Zustand betreiben wird. Bei einem Ausfall wird von dem Produktivsystem auf das Ersatzsystem gewechselt, dies kann auch durch z.B. automatische Fehlererkennung durchgeführt werden. [2]

Vorteile

- Ausfallzeiten geringer als bei Cold-Standby
- Relativ Kostengünstig
- Ersatzsystem in Betrieb, daher Datenreplikation möglich

[2]

Nachteile

- Ersatzsystem muss ständig aktuell gehalten werden
- Ggf. manuelle Aktivierung notwendig

[2]

4.2.3 Cluster

Ein "Cluster" ist ein Zusammenschluss mehrerer Rechner, dadurch kann eine höhere Verfügbarkeit und Performance gewährleistet werden. Eine Anwendung die auf so einem Cluster läuft, wird dann entweder auf einem oder parallel auf mehreren Rechnen gleichzeitig ausgeführt. [2]

4.2.3.1 Load balanced Cluster

Der "Load balanced Cluster" ist ein "Cluster", der mittels eines "Load-Balancing-Algorithmus" (siehe 7.1) die Client-Anfragen auf mehrere Server, die die Applikation bzw. Instanzen einer Applikation laufen haben, verteilt. Dadurch ist neben einer Ausfallsicherheit, die durch Redundanz geschaffen wird, auch eine Performance-Steigerung möglich. [2]

Vorteile

- Steigerung der Verfügbarkeit und Performance
- Alle Ressourcen werden dauerhaft genutzt
- Skalierbar
- Komplexität geringer als Failover Cluster

[2]

Nachteile

- Nicht für alle Applikation einsetzbar

[2]

4.2.3.2 Failover Cluster

Der "Failover Cluster" ist ein "Cluster", der bei einem Ausfall eins Rechners bzw. mehreren Rechnern den Betrieb automatisch auf funktionierende Rechner wechselt. Das automatische Wechseln wird Failover genannt und wird mittels einer sogenannten "Heartbeat"-Verbindung (siehe 7.2) zwischen den Rechnern umgesetzt. [2]

Vorteile

- Erhebliche Steigerung der Verfügbarkeit
- Voll Automatisiert

[2]

Nachteile

- Hoch komplex
- Nicht gut skalierbar
- Nur teile der Ressourcen werden genutzt
- Hohe Kosten

[2]

4.3 Projektumfeld

Projektumfeld: "UN Bojen-Netz"
Hochverfügbarkeit macht in dem Sinn bei einem solchen Projekt Sinn, da die Auswertung hochverfügbaren und zentral zugreifbar sein soll. Bei der Auswahl der Architektur kommt es sehr auf den Preis und die Applikation, die auf der Architektur laufen soll, an. Aber eine auf Cluster basierende Architektur wäre vorzuziehen, da dies sehr viele Vorteile hat (siehe 4.2.3).

5 Authentication, Authorization, Accounting

Authentication, Authorization und Accounting spielt eine wichtige Rolle bei sehr vielen Aktionen in der Informatik und vor allem im Internet. Auch in Bereichen wie Cloud-Computing ist eine richtige Authentication, Authorization und Accounting sehr wichtig und unabdingbar.

5.1 Authentication und Authorization in der Cloud

Je mehr Service von einem Provider ausgeht (je höher man sich in der Pyramide 2.3 befindet), desto mehr ist er auch für die Sicherheit und damit auch für die eventuellen Probleme verantwortlich. Durch die Integration von IT-Systemen von Organisationen und Unternehmen in Cloud-Computing-Umgebungen, entstehen neue Sicherheitsrisiken, vor allem was Sensible- und Private-Daten anbelangt. Die Lösung für solche Probleme ist "Security as a Service" (kurz SecaaS), welches ein Sicherheitskonzept ist, das Unternehmen übers Internet bereitgestellt wird. So ein Service bietet unter anderen Zugriffskontrollen und verschiedene Verschlüsselungs-Algorithmen und -Mechanismen. Dadurch soll sichergestellt werden, dass nur zugelassene User, Software oder andere Systeme auf die Ressourcen zugreifen dürfen und vor allem können, dabei müssen natürlich Sachen wie "Audit-Logging" oder Verifizierung durchgeführt werden. Doch sind die Hauptfunktonen das Authentifizieren und Autorisieren von Benutzern/Ressourcen/Services. Wie schon vorher erwähnt, ist für ein solches System Verschlüsselung einer der wichtigsten Bestandteile, da ohne dieser nicht sichergestellt werden kann, dass Verbindungen nicht Abgehört bzw. verändert werden können, auch kann ohne dieser keine Daten als sensibel oder private gehalten werden. So eine Service ist normalerweise Zentralisiert aufgebaut und wird durch standardisierte "Security-Frameworks" umgesetzt. [3]

5.1.1 Schüssel Paare

Ein wichtiger Teil damit ein solches System auch funktioniert sind Private- und Öffentliche-Schlüssel (englisch "public-private key pairs"). Mittels dieser ist es, für alle Komponenten des Systems, möglich eine Verifizierung durchzuführen und zu überprüfen, ob der Kommunikationspartner auch wirklich derjenige ist, für den er sich ausgibt. Auch sind solche Schlüssel für eine verschlüsselte Kommunikation sehr wichtig und verschlüsselte Kommunikation selber ist einer der essenziellen Bestandteile des Systems, ohne der das ganze System nicht funktionieren kann. Weil beider Teilenehmer einer Kommunikation "public-private key pairs" (seinen Privaten-Schlüssel und den Öffentlichen-Schlüssel des Kommunikationspartners) brauchen, muss es eine Infrastruktur zum abrufen der Öffentlichen-Schlüssel geben. Eine Möglichkeit Öffentlichen-Schlüssel zur Verfügung zu stellen ist ein "Public-Key-Infrastruktur" (kurz PKI) System, welches für die Cloud und den Einratszweck angepasst ist. [4][3]

5.1.2 Identitäten Management

Damit ein User auf Ressourcen/Services zugreifen kann, muss dieser identifizierbar und bekannt sein. Dafür braucht es ein (zentrales) System, welches die Identitäten speichert und einem Administrator die Möglichkeit bietet diese auch zu verwalten. Ein solches System kann dann auch mit "Single Sign-on" (5.2) verbunden werden. Dieser Bereich des "Security as a Service" (5.1) wird auch "Identity as a Service" (kurz IDaaS) genannt. So ein System kann auch mit LADP oder Active Directory verbunden werden, damit sich der Administrationsaufwand und Migrationsaufwand in grenzen hält. So ein System kann auch wie ein "Password-Save" agieren, bei dem ein User mehrere Benutzernamen und Passwörter für eventuell mehrere Services gespeichert hat. Damit die Authentifizierung mit verschiedenen Services

funktioniert, müssen sich natürlich die beteiligten Services/Ressourcen an gewisse Standards halten. [3]

5.2 Single Sign-on

Ein Unternehmen hat oft nicht nur eine Anwendung, es gibt z.B E-Mail-Server und mehrere Web-Servers mit Services, und oft wächst das Unternehmen ständig und bekommt neue Mitarbeiter und Services dazu. Da sich ein User dieser Services aber immer Authentifizieren muss, wird das bei einer großen Anzahl an Services schnell sehr unbequem. Die Lösung für ein solches Problem ist "Singe Sing-on" (kurz SSO). So eine Service funktioniert in dem es eine "Ticket-Server" gibt, welcher "Tickets" bei einer erfolgreichen Authentifizierung ausgibt, wenn dann der Authentifizierte User auf eine Ressource zugreifen will, fragt der Verwaltende Server beim "Ticket-Server" nach, ob der zugreifende User ein gültiges Ticket hat, dadurch muss sich der User nur mehr einmal "Anmelden" und kann auf alle Services, die so ein System verwenden, zugreifen. [3]

5.3 Authentisierungs-Methoden

Es gibt verschieden Methoden wie man sich bei einem System authentisieren kann. Manche davon sind einfacher zu verwenden andere sind schwerer zu umgehen bzw. zu knacken. [3]

5.3.1 Wissen

Bei der Authentisierung durch Wissen, muss der Benutzer etwas vorher wissen, damit er nachweisen kann, dass er auch wirklich er ist. Beispiele für so ein Wissen ist ein Passwort oder eine persönliche Identifikationsnummer (kurz PIN). [4]

Nachteile

- kann vergessen werden
- kann ausspioniert werden

[4]

5.3.2 Besitz

Bei der Authentisierung durch Besitz, muss der Benutzer etwas fälschungssicheres besitzen. Dieser Besitz beinhalten die Informationen in mechanischer, magnetischer oder elektrischer Form. Beispiele für so einen Besitz ist ein Personalausweis, eine EC-Karte oder die Transaktionsnummern beim Internet-Banking (TAN-Listen). Bei Computern ist es oft eine Token-Karte, ein SSL-Zertifikat, eine Smartcard oder eine Liste von Einmalpasswörtern. [4]

Nachteile

- kann verloren werden
- kann entwendet werden (gegensteuern mit kurzer Gültigkeitsdauer)

[4]

5.3.3 Biometrische Merkmale

Bei der Authentisierung durch ein biometrisches Merkmal, muss der Benutzer ein biometrisches Merkmal besitzen. Beispiele für so ein Merkmal sind Aussehen, Stimme, Fingerabdruck, Augenhintergrund oder Unterschrift. [4]

Nachteile

- kann der Computer nicht besonders gut erkennen
- ist fehleranfällig

[4]

5.4 Projektumfeld

Projektumfeld: "Loxone"
Authentication und Authorization ist einer der wichtigsten Dinge die ein System im allgemeine braucht und ist unabdingbar. Auch Sensoren oder mobile Robotik muss sich bei einem "Internet of Things" (kurz IoT) oder Cloud Computing Service Authentifizieren und Autorisieren, dabei reicht ein "einfaches" Password oder Zertifikat, die vorher eingespielt und sicher "aufbewahrt" werden. Die Benutzer eines solchen Systems müssen sich natürlich auch Authentifizieren und Autorisieren damit kein Missbrauch entstehen kann.

6 Desaster Recovery

Desaster Recovery ist eine Möglichkeit, bei richtig großen Desastern wie z.B. bei Naturkatastrophen wie Erdbeben oder Tsunamis, vor allem Daten zu sichern.

6.1 Disaster Recovery as a Service

"Disaster Recovery as a Service" (kurz DRaaS) ist eine, wie man am Namen schon verratet, Art des Cloud-Computings. Der Unterschied zu einem "Cloud-Backup" ist, dass der ein solcher Service auch Rechnerleistung anbietet und als "Failover" für einen eventuellen Ausfall eine Produktivsystems dient. Auch ist ein solcher Service billiger, effizienter und oft schneller als der "traditionelle" Aufbau. [5]

6.1.1 Architekturen

To-Cloud Disaster Recovery as a Service Die Anwendung befindet sich nicht in der Cloud aber sie wird als Backup und Disaster Recovery verwendet. [5]

In-Cloud Disaster Recovery as a Service Sowohl die Anwendung selbst als auch das Backup sind in der Cloud. [5]

From-Cloud Disaster Recovery as a Service Genau das Gegenteil zu "To-Cloud Disaster Recovery as a Service" (siehe 6.1.1), die Anwendung selbst befindet sich in der Cloud, das Backup jedoch nicht. [5]

6.1.2 Sandbox

Eine weiter Funktion von "Disaster Recovery as a Service" ist die möglichkeit Sandboxen zu erstellen, in denen mit Produktivdaten getestet werden kann. Diese Sandbox ist ein abgetrennter Teil, auf welchen nur Administratoren Zugriff haben, auf dem eine Kopie der Produktivdaten liegt. [5]

6.2 Recovery Plan für Ausfall einer geografischen Zone der Cloud

In einer Cloud-Computing-Umgebung kann es auch nötig sein, gewisse Routinen vorzubereiten, um sehr seltene Ausnahmezustände abzusichern. [5]

6.2.1 Eine aktive Zone

Um so einen Fall zu verhindern müssen zuerst einmal die Grundlagen geklärt werden. Die erste Frage die man stellen muss, ist wie schnell ein solches System wiederhergestellt werden soll. Wenn man vom Extremfall ausgeht, dann soll ein solches System wenige Sekunden "downtime" haben. Dazu muss erstmal (mindestens) ein Zweitsystem parallel in einer anderen geografischen Zone betrieben/aufrecht gehalten werden. Diese Systeme sollte dem eigentlichen Produktivsystem gleichen und vor allem die gleichen Eigenschaften/Technologien verwenden. Das Wichtigste, dass bei einem Desaster verloren gehen kann sind die Daten, deshalb müssen (vor allem) diese gebackupt werden. Eine Möglichkeit wäre es, da die Daten sehr Wahrscheinlich auf einer Datenbank liegen, die Datenbank mit den Daten zu replizieren. Das heißt, es werden laufend die Daten des Produktivsystems auf das Backup-System "kopiert" und so ein ständiges Backup erzeugt. Dabei muss man aber davon absehen, dass dies ein Kontinuierliches-Backup ist und es keine Snapshots/Volle Backups erstellt werden, d.h. es gibt keine "Recovery-Point", wenn einmal ein Inkonsistenter oder nicht gewollter Zustand zustande kommt,

muss dieser mittels anderen Technologien/Methoden wieder ausgeglichen werden. Deshalb rät es sich, dennoch "normale" Backups zu erstellen. Da sich das Backup-System jedoch auf jeden Fall in einem "Standby-Mode" befindet, sollte die Konsistenz, vor allem beim Replizieren einfach gewahrt werden können, da nur in eine Richtung repliziert werden muss. Der Rest der Infrastruktur, muss nur soweit auf dem aktuellen Stand gehalten werden, so dass jederzeit bei einem Ausfall, darauf gewechselt werden kann. Doch braucht die restliche Infrastruktur "normalerweise" nicht Verfügbar zu sein, was Kosten und Ressourcen spart. [5]

Was passiert bei einem Ausfall einer geografischen Zone? Nachdem nun eine geografische Zone des Produktivsystems ausgefallen ist, muss die/eine andere geografische Zone einspringen. Dazu muss man erstens mal davon ausgehen, dass die andere/anderen Zonen den aktuellen Stand der Daten hat/haben. Nun muss es automatische Mechanismen geben, welche erkennen, dass die eine Zone nicht mehr erreichbar ist. Dazu muss eine Dienst laufen, der immer den Status der Zone/Zonen abprüft. Dieser Dienst muss einen gewissen Spielraum haben, damit er bei Schwankungen oder kurzen Aussetzern nicht gleich "anschlägt". Nachdem ein gewisser "Threshold" überschritten wurde, muss dieser Dienst reagieren. Als erste muss er die Schreibvorgänge der Replikation auf der Datenbank abschließen und sicherstellen, dass die Datenbank konsistent ist. Danach kann er die Verbindung zur ausgefallenen Datenbank trennen und seine Datenbank für den Produktivbetrieb bereitstellen. Währenddessen kann die Restliche Infrastruktur angefordert und "hochgefahren" werden. Nachdem alles bereit für den Produktivbetrieb ist, können die DNS-Einträge so geändert werden, dass nun diese Zone für alle verfügbar ist und vor allem die ausgefallene Zone ersetzt. So ein "Failover" sollte für den Kunden nicht wirklich bemerkbar sein, vielleicht bricht die Verbindung zum Server einmal ab und der Benutzer muss die Seite neu laden, mehr sollte aber nicht passieren. [5]

6.2.2 Mehrere aktive Zonen

Im vorherigen Fall, gab es nur eine aktive Zone, also nur eine Zone, welche im Produktivbetrieb gelaufen ist. Sollte es mehrere geografische Zonen geben, welche aktiv sind, muss vorher überlegt werden, wie das ausfallen eine Zone gehandhabt wird. Da es andere aktive Zonen gibt, die die Aufgaben übernehmen können, ist nicht so ein großer Aufwand nötig wie bei einer aktiven Zone (siehe 6.2.1). Es muss nur definiert werden, wer in die Rolle dieser Zone springt, oder ob die "Requests" auf die übrigen Zonen aufgeteilt werden. So eine geografische Zone hat normalerweise den Zweck, bestimmte "Regionen" an Clients zu bedienen. Diese müssen nun auf eine andere Zone geleitet werden, dazu kann die ausgefallene Zone aus dem Pool der Zonen genommen werden (oft DNS oder Loadbalancer basierend) und die Clients werden ab nun auf eine andere Zone geleitet. Da die Zonen sowieso ihr Daten untereinander synchronisieren müssen (meistens mit Replikation), sind die Daten sowieso synchron. Was bei einer Planung beachtet werden muss ist, dass es nicht möglich ist, das sich mehrere Zonen in die Quere kommen, d.h. es muss verschiedene Werte bezüglich "Timeouts" geben, damit nicht mehrere Zonen anfangen, Maßnahmen zu setzten und sich in die Quere kommen. [5]

6.3 Projektumfeld

Projektumfeld: "UN Bojen-Netz"
Eine Cloud-Computing-Plattform, welche die gemessenen Daten auswerten soll, sollte eine Desaster Recovery Plan haben. Denn bei einem Erdbeben welches einen Tsunami hervorrufen kann, kann es passieren, dass eine ganze geografische Zone der Cloud-Computing-Plattform ausfällt und keine Daten mehr empfangen werden können.

7 Algorithmen und Protokolle

In einer Cloud-Computing-Umgebung und vor allem bei Hochverfügbarkeit gibt es ein paar Algorithmischem und Protokolle, welche umgesetzt werden.

7.1 Load Balancing Algorithmen

Es gibt verschiedene Algorithmen um die Last eines Server auf mehrere zu verteilen. Diese werden bei Load Balancern oder Load balanced Clustern (siehe 4.2.3.1) eingesetzt. [6]

7.1.1 Round-Robin

"Round-Robin" ist ein sehr einfacher Algorithmus und profitiert von Hardware gleichen Servern. Mit "Round-Robin" wird jedem Server der Reihe nach eine Anfrage/Verbindung zugewiesen. Da aber meist nicht jede Anfrage gleich lange braucht um abgearbeitet zu werden, resultiert das in einer nicht gleichmäßigen Verteilung der Last. Vor allem "schwächere" Server können dadurch überladen werden. Durch die Einfachheit, fällt aber weniger Rechenaufwand auf den Load Balancer. [6]

7.1.2 Least Connections

Wie der Name schon verraten lässt, überprüft der Load Balancer bei "Least Connections" die Anzahl an offenen Verbindungen der Server und teilt die Anfragen den Server mit den wenigsten Anfragen zu. Dafür muss der Load Balancer aber immer die aktuelle Anzahl der Verbindungen auf den Servern kennen. Dieser Algorithmus wird sehr häufig eingesetzt und ist sehr effektive. [6]

7.1.3 Weighted Distribution

Bei "Weighted Distribution" wird jedem Server eine gewisse Gewichtung zugeteilte, dadurch können hardwareschwache Server kompensiert werden, in dem "stärkere" Server mehr last zugewiesen wird. Oft wird dieser Algorithmus mit anderen kombiniert, daraus resultiert "Weighted Round-Robin" (Verteilung nach Gewichtung, nach der Reihe) und "Weighted Least Connections" (Anzahl Verbindungen durch Gewichtung). [6]

7.1.4 Response Time

Hierbei wird nach der Reaktionszeit des Server bestimmt, wie gut die Performance der Server ist. Dafür müssen aber mehrere Messungen auf einer längere Zeitspanne durchgeführt werden und der Algorithmus ist sehr komplex. [6]

7.1.5 Server-Probe

Dabei werden verschiedene Werte, wie z.B. CPU- oder RAM-Auslastung, auf den einzelnen Server gemessen und dadurch bestimmt, welcher die wenigste Auslastung hat. Dafür muss aber extra Software entwickelt und auf dem Server installiert werden und diese nehmen dadurch der eigentlichen Anwendung Ressourcen weg. [6]

7.1.6 Kombiniert

Durch die Kombination von zwei oder mehreren Methoden kann eine bessere Erfassung der Performance und Serverlast erzielt werden. [6]

7.1.7 Server Load Thresholds

Damit Server nicht überlastet (Voller Arbeitsspeicher oder volle CPU-Auslastung) werden, kann eine maximale Auslastung der Server beim Load Balancer einstellt werden. Diese maximale Auslastung sollte knapp unter dem eigentlichen Limit angesetzt werden und so vor Überlastung schützen. [6]

7.1.8 Zufällig

Mittels eines Zufallszahlengenerators (auf englisch RNG (Random number generation)) kann auch die Last auf verschiedene Server "verteilt" werden. Das kann zu einer gleichmäßigen Verteilung führen und mach vor allem bei gleicher Hardwarekonkguration Sinn. [6]

7.2 Heartbeat

Heartbeat ist ein Daemon (UNIX Hintergrundprozess), welcher Cluster-Infrastruktur-Services für seine Clients anbietet. Es bietet die Möglichkeit den Zustandes des Peer-Prozessen auf einem anderen Nodes zu überprüfen und Nachtrichten mit diesem Auszutauschen. [7]
So ein Dienst wird bei Clustern benötigt, damit die einzelnen Nodes über den Status der anderen bescheint wissen. Dies ist essentiell, damit Hochverfügbarkeit funktionieren kann. Heartbeat läuft am besten auf einem eigenen Netzwerkinterface und verwendet damit eine eigene Verbindung zu den anderen Nodes. Es ist auch möglich mehrere Instanzen von Heartbeat laufen zu lassen und über mehrere Verbindungen mit den anderen Nodes zu kommunizieren, das bietet den Vorteil, mehr Ausfallsicherheit zu haben, was eine höher Verfügbarkeit mit sich bringt, denn sollte die einzige "Leitung" ausfallen, glauben alle Nodes, die jeweils anderen sind ausgefallen und sie müssen jetzt die Aufgabe übernehmen. [7]
Damit dieser sinnvoll verwendet werden kann, wird der Heartbeat-Deamon mit einem "Cluster Ressource Manager" (kurz CRM) verbunden, welcher Aufgaben wie Starten oder Stoppen eines Services (IP-Adressen, Webserver, etc.) eines Cluster übernimmt, was ihn Hochverfügbar macht. Pacemaker ist der bevorzugte CRM für Heartbeat. [7]

7.3 Simple Network Management Protocol

Das "Simple Network Management Protocol" (kurz SNMP) ist ein Set an "einfachen" Netzwerk-Operatoren, welche das Entfernte-Verwalten von Geräten ermöglicht, damit ist es möglich den Zustand eines SNMP-basierenden Devices zu ändern. Ein Beispiel für so eine Operation, welche mit SNMP durchgeführt werden kann, wäre das Abschalten eins Netzwerk-Interfaces auf meinem Router oder das checken der Netzwerkgeschwindigkeit meines Ethernet-Interfaces. Mit SNMP ist es sogar möglich die Temperatur meines Switches zu Monitoren und bei zu hohen Temperaturen zu warnen. Sogar Unix-Systeme, Windows-Systeme, Drucker, Netzteile oder anderes, kann mittels SNMP angesprochen werden. Jedes Gerät, welches Software laufen hat, welche SNMP-Befehle annehmen kann, kann auch mittels SNMP verwaltet werden. Wobei es nicht mal ein physisches Geräte sein muss, sogar Software wie Webserver oder Datenbanken können SNMP-Befehle annehmen. Es ist mittels SNMP möglich sein ganzes Netzwerk, statt nur eines Devices, zu Monitoren. Die aktuelle Version des Protokolls ist SNMPv3 und wird von der "Internet Engineering Task Force" (kurz IETF) entwickelt. Es basiert auf UDP und hat seit der Version 3 Sicherheitsmechanismen eingebaut. [8]

7.3.1 Managers und Agents

Der Manager ist eine Server, welcher mittels Software zum Annehmen von Management-Tasks für das Netzwerk eingesetzt wird. Dieser befindet sich oft auf einer sogenannten "Network Management Station" (kurz NMS) und ist verantwortlich für Polls und zum empfangen von Traps, welche von Agents gesendet worden sind. Ein Poll ist die Möglichkeit Informationen von Agents abzurufen, welche dann im nächsten Schritt interpretiert und darauf reagiert werden kann. Traps werden asynchron versendet und sind nicht für das Abrufen von Informationen verantwortlich. Der NMS ist mehr dafür verantwortlich, eine Aktion als Reaktion auf einen empfangenen Trap auszuführen. [8]
Der Agent ist ein Stück Software, welches auf einem Netzwerk-Gerät läuft. Dieser ist dazu da, Information für den NMS bereitzustellen, also das Gegenstück zum NMS. Er ist der, der die Traps abschickt und auf Polls antwortet, welche vom NMS gesendet werden. [8]

7.4 Projektumfeld

Projektumfeld: "UN Bojen-Netz"
Bei einem solchen Projekt, wo ein ausfallsicheres System benötigt wird, macht es Sinn gewisse Hochverfügbare Technologien einzusetzen. Wenn man sich dazu entscheidet einen Cluster umzusetzen, dann gibt es, wie in "Hochverfügbare Architekturen" (4.2) schon erwähnt, zwei Typen von Cluster. Diese beiden verwenden verschiedene Methoden/Technologien um miteinander zu Kommunizieren und interagieren. Abgesehen davon gibt es die Möglichkeit, z.B. den Status einzelner Netzwerkkomponenten zu überwachen und zu kontrollieren.

8 Konsistenz und Datenhaltung

Fencing ist ein sehr wichtiges Konzept in HA-Clustern. Fencing ist eine Methode Cluster auf einen bekannten Zustand zu bringen. Ein unbekannter Zustand ist z.B. Split Brain und eine Art des Fencing ist STONITH.

8.1 Split Brain

Split Brain ist ein Zustand, der das Ergebnis einer "Claster Partiton" ist. Bei so einem Zustand glauben beide Seiten eins Cluster, dass der andere "tot" ist und dann starten Prozesse, welche Ressourcen übernehmen, da sie glauben, die andere Seite greift nicht mehr darauf zu. So ein Zustand kann sehr leicht zu Inkonsistenzen führen und ist daher sehr ungewollt und zu vermeiden. Das passiert, da die Teilnehmer nach unvollständiger Information handeln. Wenn ein Node als "tot" erklärt wird, dann ist sein Status per Definition unbekannt. Vielleicht ist er "tot" oder er kann nur nicht Kommunizieren. Das einzige was bekannt ist, ist das der Status unbekannt ist. Das beste Mittel um gegen Split Brain vorzugehen ist Fencing (siehe 8.2). Eine Gute Möglichkeit, Split-Brain-Zustände zu vermeiden ohne Fencing einzusetzen, ist meistens redundante und unabhängige Claster-Kommunikationswege zu konfigurieren, sodass beim Verlust eines Interfaces oder Weges, die Kommunikation zwischen den Nodes nicht zum erliegen kommt, d.h. die Kommunikation sollte nicht einen "Single Point of Failure" (kurz SPOF) sein. Es wird stark empfohlen sowohl redundante Kommunikation als auch Fencing einzusetzen. [9]

8.2 Fencing

Es gibt zwei Arten des Fencing: Ressource-Level und Node-Level [9]

Ressource-Level Ressource-Level-Fencing hat die Aufgabe sicherzustellen, dass ein Node auf eine oder mehr Ressourcen nicht zugreifen kann. Ein typisches Beispiel wäre SAN, wo eine Fencing-Operation Zugriff auf einem SAN-Switch mit verändern von Regeln verhindert werden kann. Eine Möglichkeit Ressource-Level-Fencing umzusetzen ist die zu schützende Ressource von einer "normale" Ressource abhängig zu machen. Die normale Ressource starteten dann ganz einfach auf dem einen Node nicht und so kann auch die zu schützenden Ressource auf dem Node nicht starten. [10]

Node-Level Node-Level-Fencing stellt Sicher, das auf einem Node keine Ressource läuft. Das ist normalerweise in einer recht einfach aber brutal weise Umgesetzt, in dem der Node mittels dem Powerswitch zurückgesetzt wird. Das ist vielleicht sogar nötig, da der Note nicht mehr Ansprechbar ist. Dafür wird spezielle (virtuelle) Hardware benötigt.[10]

Die großen Vorteile von Fencing sind, das man kein Wissen über das Timing oder Verhalten des "verwirrten" Nodes braucht, noch braucht es keine Kooperation mit den verwirrten Nodes. Zusätzlich erhalten "Fenzing-Operations" eine Positive Bestätigung. Daher ist Fencing sehr zuverlässig und wird sehr oft eingesetzt. [9]

8.3 STONITH

STONITH steht für "Shoot The Other Node In The Head" und ist eine Umsetzung des Node-Level-Fencing. Es gibt fünf Kategorien von Geräten, die STONITH unterstützen:

- UPS (Uninterruptible Power Supply)

- PDU (Power Distribution Unit)
- Blade-Power-Control-Devices
- Lights-Out-Devices
- Testing-Devices

[10]

Die Auswahl hängt in erster Linie vom Budget und der Art der Hardware ab. Lights-Out-Devices erfreuen sich über eine steigende Beliebtheit und werden in der Zukunft vielleicht zum Standard. Das STONITH-Subsystem, besteht aus zwei Komonenten, dem stonithd und den STONITH plugins. [10]

8.3.1 stonithd

stonithd ist ein Daemon, welcher über lokale Prozesse oder über das Netzwerk zugreifbar ist. Dieser akzeptiert Kommandos, welche den Fencing-Operatoren gleichen: reset, power-off, und power-on. Es kann auch der Status des Fencing-Devices überprüft werden. stonithd läuft auf jedem Node des CRM-HA-Clusters und bekommt die Fencing-Anfragen vom CRM. [10]

8.3.2 STONITH plugins

Für jedes Unterstütztes Fencing-Gerät gibt es ein STONITH-Plugin, welches das Gerät steuern kann. Das STONITH-Plugin ist das Interface zum Fencing-Gerät, welches für den stonithd immer gleich ausschaut. Es gibt Plugins, welche mehr können als andere. Es gibt ein paar Dinge, die ein Plugin haben muss:

1. Es dürfen keine “False-Positive” für eine Reset gemeldet werden. Wenn ein STONITH-Plugin meldet, das eine Node down ist, dann ist er besser down.
2. Müssen das RESET-Kommando können (ON und OFF sind Optional).
3. Wenn ein RESET- oder OFF-Befehlt abgesetzt wurde, dann darf nicht returnt werden, bis der Node ganz heruntergefahren ist. Auf das Hochfahren bei einem RESET zu warten ist optional.
4. Alle Kommandos sollten unter allen Umständen funktionieren:
   a) RESET wenn der Node ON oder OFF ist, sollte erfolgreich sein und den Node hochfahren (oder es wenigstens probieren - vielleicht bootet der Node wegen andren Gründen nicht).
   b) OFF wenn ein Node OFF ist, sollte gelingend
   c) ON wenn ein Node ON ist, sollte gelingend

[10][11]

8.4 Projektumfeld

Projektumfeld: “UN Bojen-Netz”
Bei einem hochverfügbaren System wird Fencing benötige, welches eventuellen Konsistenz-Problemen zuvorkommt, daher ganz wichtig, dass z.B. STONITH umgesetzt wird. Ebenfalls ist es nötig, mehrere Verbindungswege zwischen den verschiedenen Cluster-Nodes zu haben, um ein mögliches Split-Brain-Problem zu lösen.

9 Conclusio

Cloud-Computing mach in viele Bereichen Sinn, es ist also kein Wunder, warum es so viel eingesetzt wird. Die Technologien, die hinter einer solche Plattform stecken, sind sehr ausgereift und gut durchdacht. Viele Probleme, welche bei einer Plattform dieser Art auftreten können, werden durch intelligente Technologien, Algorithmen, Abläufe oder Pläne verhindert. Ohne Hochverfügbarkeit würde keine Cloud-Computing-Plattform laufen, ohne Desaster Recovery könnte es passieren, dass der Kunde seine Daten verliert, ohne Load Balancing würde er ewig auf seine Anfragen warten und ohne richtiger Sicherheit würde jeder seine Daten stehlen können. Es benötigt viel damit die “Cloud” so funktioniert wie man sie kennt und liebt.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] Burkhard Hampl, Simon Wortha, "Cloud-Computing." Februar 2016.

[2] Hackenberger Christoph, Malik Patrick, "Hochverfügbarkeit." April 2016.

[3] Davit Hakobyan, "Authentication and Authorization Systems in Cloud Environments," 2012.

[4] Andreas Ernhofer, Jakub Kopec, "Autorisierung & Authentifizierung." November 2015.

[5] Erik Brändli, Michael Weinberger, "Synchronisierung & Konsistenz." Februar 2016.

[6] Alexander Kölbl, Thomas Taschner, "Load Balancing." Februar 2016.

[7] "Heartbeat - Linux-HA." `http://linux-ha.org/wiki/Heartbeat`, Jänner 2010.

[8] Douglas Mauro, Kevin Schmidt, *Essential SNMP: Help for System and Network Administrators 2. Auflage*. September 2005.

[9] "Split Brain - Linux-HA." `http://linux-ha.org/wiki/Split_Brain`, Jänner 2010.

[10] "Fencing and Stonith." `http://clusterlabs.org/doc/crm_fencing.html`, Jänner 2015.

[11] "STONITH - Linux-HA." `http://www.linux-ha.org/wiki/STONITH`, Februar 2010.