16. April 2015

1 Einleitung

1.1 Motivation und Fragestellung

Der Zugriff auf Services und Medien mittels mobiler Geräte steigt beständig an. So ist im Mai 2014, 60% der Zeit, die online verbracht wird, über Handy und Tablet zugegriffen worden - davon 51% mittels mobiler Applikationen. [4]

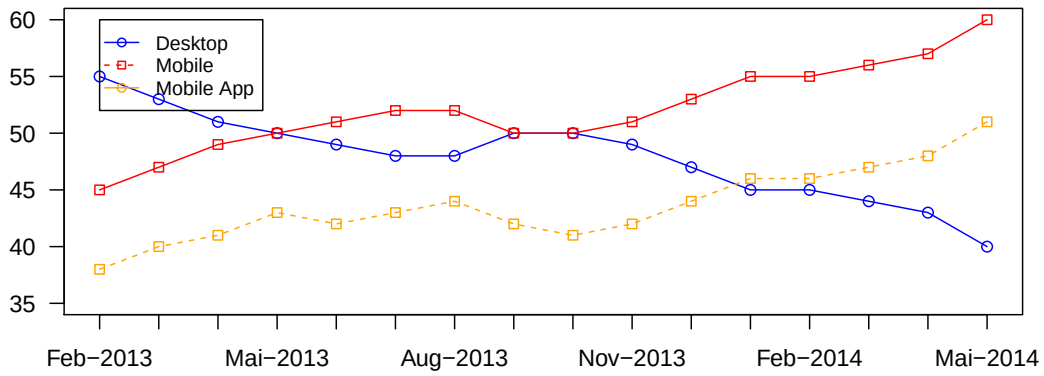

Abbildung 1.1: Verteilung der online verbrachten Zeit nach Platform (Grafik erstellt gemäss der Daten von [4])

Angesichts der grossen Verbreitung und Nutzung von Services und Medien im Internet, wird eine unterbruchsfreie Benutzbarkeit, auch ohne Internetverbindung, immer selbstverständlicher und somit auch immer wichtiger.

> „It's clear that the mobile industry has finally given up on the fantasy that an Internet connection is available to all users at all times. Reality has set in. And in

> the past month, we've seen a new wave of products and services that help us go offline and still function.“ Elgan [3]

Es stellt sich nun die Frage, wie Informationen, über Verbindungsunterbrüche hinweg, integer gehalten werden können. Und wie Daten im mobilen Umfeld synchronisiert, aktualisiert und verwaltet werden könne, so dass für den Endbenutzer schlussendlich kein Unterschied zwischen Online- und Offline-Betrieb mehr wahrnehmbar ist.

1.2 Aufgabenstellung

Die von der Leitung des Studiengangs Informatik freigegebene Aufgabenstellung ist im Appendix unter „Aufgabenstellung“ aufgeführt.

1.3 Abgrenzung der Arbeit

1.4 Begründung

1.5 Geschichtliche Einordnung in das Thema

2

Dokumentationsstruktur und Beitrag zum Forschungsgebiet

Teil i - Einleitung und Abgrenzung

Teil ii - Technische Grundlagen und Architekturen

Teil iii - Konzept und Implementierung

Teil iv - Abschluss und Ausblick

3
Recherche

Dieses Kapitel erklärt die wichtigsten Grundbegriffe und wiedergibt die während der Recherche gesammelten Informationen.

3.1 Fachbegriffe

Eine Aufführung und Erlährung der Fachbegriffe befindet sinch im Appendix unter „[Glossar]“

3.2 Erläuterung der Grundlagen

3.2.1 Datenbanken

Eine Datenbank [1] ist ein System zur Verwaltung und Speicherung von strukturierten Daten. Erst durch den Kontext des Datenbankschemas wird aus den Daten Informationen, die zur weiteren Verarbeitung genutzt werden können. Ein Datenbanksystem umfasst die beiden Komponenten Datenbankmanagementsystem (DBMS) sowie die zu veraltenden Daten selbst.
Ein DBMS muss die vier Aufgaben [2] erfüllen.

- Atomarität
- Konsistenzerhaltung
- Isolation
- Dauerhaftigkeit

Neben den vielen neu auf den Markt erschienen Technologien wie Document Store oder Key-Value Store ist das Relationale Datenbankmodell immer noch am verbreitetsten. [2].

[1] In der Literatur oft auch als **D**aten**B**ank**S**ystemen (DBS) oder Informationssystem bezeichnet. [5, pp. 3-4]
[2] Bekannt als ACID-Prinzip [5, pp.105] umfasst es **A**tomicity, **C**onsistency, **I**solation und **D**urability.

3.2.2 Monolithische Systeme

Als Monolithisch wird ein logisches System bezeichnet, wenn es in sich geschlossen, ohne Abhängigkeiten zu anderen Systemen operiert. Alle zur Erfüllung der Aufgaben benötigten Ressourcen sind im System selbst enthalten. Es müssen also keine Ressourcen anderer Systeme alloziert werden und somit ist auch keine Kommunikation oder Vernetzung notwendig.
Das System selbst muss jedoch nicht notwendigerweise aus nur einem Rechenknoten bestehen, sondern darf auch als Cluster implementiert sein.

3.2.3 Verteilte Systeme

Man kann zwischen physisch und logisch verteilten Systemen unterscheiden. Weiter kann das System auf verschiedenen Abstraktionsstufen betrachtet werden. So sind je nach Betrachtungsvektor unterschiedliche Aspekte relevant und interessant. [8]

physisch verteilte Systeme

Rechnernetze und Cluster-Systeme werden typischerweise als physisch verteiltes System betrachtet. Die Kommunikation zwischen den einzelnen Rechenknoten erfolgt Nachrichten orientiert und ist somit asynchron ausgelegt. Jeder Rechenknoten verfügt über exklusive Speicherressourcen und einen eigenen Zeitgeber.
Durch die Implementation eines Systems über mehrere unabhängige physische Rechenknoten kann eine erhöhte Ausfallsicherheit und/oder ein Performance-Gewinn erreicht werden.

logisch verteilte Systeme

Falls innerhalb eines Rechenknoten echte Nebenläufigkeit[3] oder Modularität[4] erreicht wird, kann von einem logisch verteilten System gesprochen werden. Einzelne Rechenschritte und Aufgaben werden unabhängig voneinander auf der selben Hardware ausgeführt. Dies ermöglicht den flexiblen Austauschen[5] einzelner Aufgaben.

3.2.4 Verteilte Algorithmen

Verteilte Algorithmen sind Prozesse welche miteinander über Nachrichten (synchron oder asynchron) kommunizieren und so idealerweise ohne Zentrale Kontrolle eine Kooperation erreichen. [7]
Performance-Gewinn, bessere Skalierbarkeit und eine breitere Abdeckung der unterstützen von verschiedenen Hardware-Architekturen kann durch den Einsatz von verteilten Algorithmen erreicht werden.

[3] Von echter Nebenläufigkeit wird gesprochen, wenn verschiedene Prozesse zur selben Zeit ausgeführt werden. (Multiprozessor)
[4] Modularität beschreibt die Unabhängigkeit und Austauschbarkeit einzelner (Software-) Komponenten. (Auch Lose Kopplung gennant)
[5] Austauschbarkeit einzelner Programmteile wird durch die Einhaltung der Grundsätze von modularer Programmierung erreicht.

3.2.5 Verteilte Datenbanken

[Präsenzbibliothek ZHAW]

3.2.6 Replikation

Replikation vervielfacht ein sich möglicherweise mutierendes Objekt (Datei, Dateisystem, Datenbank usw.), um hohe Verfügbarkeit, hohe Performance, hohe Integrität oder eine beliebige Kombination davon zu erreichen. [1, p. 19]

Synchrone Replikation

Eine synchrone Replikation stellt sicher, dass zu jeder Zeit der gesamte Objektbestand auf allen Replikationsteilnehmern identisch ist.

Wird ein Objekt eines Replikationsteilnehmer mutiert, wird zum erfolgreichen Abschluss dieser Transaktion, von allen anderen Replikationsteilnehmern verlangt, dass sie diese Operation ebenfalls erfolgreich abschliessen.
Üblicherweise wird dies über ein Primary-Backup Verfahren realisiert. Andere Verfahren wie der 2-Phase-Commit und 3-Phase-Commit ermöglichen darüber hinaus auch das editieren von Objekten auf allen Replikationsteilnehmern. [1, p. 23ff, 134ff]

Asynchrone Replikation

Eine asynchrone Replikation, stellt periodisch sicher, dass der gesamte Objektbestand auf allen Replikationsteilnehmern identisch ist. Mutationen können nur auf dem Master-Knoten durchgeführt werden. Einer oder mehrere Backup-Knoten übernehmen dann periodisch die Mutationen. Entgegen der synchronen Replikation müssen nicht alle Replikationsteilnehmer zu jedem Zeitpunkt verfügbar sein[6].

Merge Replikation

Die merge Replikation erlaubt das mutieren des Objekts auf allen Replikationsteilnehmern.
Mutationen an einem einzelnen Replikationsteilnehmer werden allen übrigen Replikationsteilnehmern mitgeteilt. Da ein Objekt zwischenzeitlich[7] auch auf anderen Teilnehmern mutiert worden sein kann, müssen während des Synchronisationsvorgang[8] eventuell auftretenden Konflikte aufgelöst werden.

[6] So kann der Backup-Knoten nur Nachts über verfügbar sein, damit die dazwischen liegende Verbindung Tags über nicht belastet wird.

[7] Zwischen der lokalen Mutation und der Publikation dieser an die übrigen Replikationsteilnehmer, liegt eine beliebige Latenz.

[8] Da die Replikation nicht notwendigerweise nur unidirektional, sondern im Falle von einem Multi-Master Setup auch bidirektional durchgeführt werden kann, wird hier von einer Synchronisation gesprochen.

3.2.7 Block-Chain

Die Block-Chain ist eine verteilte Datenbank die ohne Zentrale Kontrolle auskommt. Jede Transaktion wird kryptographisch gesichert, der Kette von Transaktionen hinzugefügt. So ist das entfernen oder ändern vorhergehender Einträge nicht mehr möglich[9]. Jeder Teilnehmer darf also alle Einträge lesen und neue Einträge hinzufügen. Da Einträge nur hinzugefügt werden und nie ein Eintrag geändert wird, kann eine Block-Chain immer ohne Synchronisationskonflikte repliziert werden. Konflikte können nur in den darüberlegenden logischen Schichten[10] auftreten. [6]

[9] Das ändern vorhergehender Einträge benötigt mehr Rechenzeit, als alle anderen Teilnehmer ab diesem Zeitpunkt zusammen aufgewendet haben.

[10] So prüft die Bitcoin-Implementation ob eine Transaktion (Überweisung eines Betrags) bereits schon einmal ausgeführt wurde, und verweigert gegebenenfalls eine erneute Ausführung.

4 Analyse

4.1 Datenanalyse

4.2 Diskussion bekannter Verfahren

4.3 Anforderungsanalyse

4.4 Vorgehensweise

4.4.1 Use-Cases

4.4.2 Anforderungen

4.4.3 Akzeptanzkriterien

4.4.4 Bewertung der Anforderungen

4.5 Risiken

5 Konzept

5.1 Konfliktverhinderung

5.1.1 2PC/3PC

5.1.2 Block-Chain

5.1.3 Update Transformation

5.2 Konfliktauflösung

5.2.1 Merge

5.2.2 Normalized Merge

5.2.3 Wiederholbare Transaktion

5.2.4 Kausalitäts Merge