HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

CAD-Project - Mom based Information Live Flow

Paul Drautzburg, Lukas Hansen, Georg Mohr, Kim De Souza, Sebastian Thuemmel, Sascha Drobig

Vorwort

Das vorliegende Dokument beschreibt grob eine Idee und das Vorgehen für die Umsetzung für das Projekt im Rahmen der Master Veranstaltung *Cloud Application Development*.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Problemstellung

Wir erhalten eine große Menge an Sensordaten die zur Verarbeitung und Erfassung von Wetterdaten, Berechnung von Statistiken und Erstellung von Wetterwarnungen verwendet werden sollen. Der Eingang der Wetterdaten erfolgt kontinuierlich und Belastungsspitzen sind nur in Ausnahmefällen zu erwarten. Zusätzlich sollen beliebige zusätzliche Wetterdaten in den Verarbeitungsprozess integriert werden können. Da zum Empfang der Daten und anschließenden Verarbeitung kein Server vollständig ausgelastet wird, soll das System in einer Cloud-Lösung ausgelagert werden. Des Weiteren ist die Cloudlösung nötig, da wir keine eigene Server-Infrastruktur betreiben wollen oder können, da dies aus finanzieller, organisatorischer und infrastruktureller Sicht vorteilhaft ist.

2 Lösungsansatz

In diesem Abschnitt wird vorerst auf eine kurze Beschreibung für die im vorhergehenden Kapitel beschriebenen Problemstellung eingegangen. Anschließend wird aus der allgemeinen Beschreibung eine erste Anforderungsdefinition an eine Softwarelösung abgeleitet.

2.1 Allgemein

Der Ansatz einer angemessenen Softwarelösung schließt drei heterogene Systeme mit ein.

Diese drei heterogenen Systeme bestehen aus

- Sender von Wetterdaten,
- Anwendung für *Complex Event Processing*,
- Clients zum Empfang und zum Darstellen der aufbereiteten Wetterdaten und Warnungen,

sollen durch eine *Message-oriented Middleware* (MoM) kommunizieren. Die MoM soll im allgemeinen Sensordaten verarbeiten können, welche aus prinzipiell jeder Art von Binärdaten bestehen können, somit kann diese MoM gleichzeitig für andere Szenarien verwendet werden.

Im vorliegendem Fall beschränken sich die zu verarbeitenden Daten auf Wetterdaten aus einer Wetter-API. Diese angesprochenen Wetterdaten können sowohl als Json wie auch als XML abgezogen werden. Um die Auslastung einer Instanz steuern zu können, wird die MoM auch fingierte Wetterdaten weiterleiten können und durch Complex Event Processing (CEP) verarbeiten können. Ein weiterer Hintergrund für diese Entscheidung

war die Richtigkeit der errechneten Daten. Die Wetterdaten der API können, zu keinem Zeitpunkt, genau vorhergesagt werden und somit können die Ergebnisse des CEP nicht auf Richtigkeit validiert werden.

Ferner soll die Lösung bei einer vorher definierten Systemauslastung weitere Ressourcen dazuschalten können.

Die Abbildung 1. zeigt das Ergebnis eines Brainstormings, welches als Grundlage für einen Lösungsansatz dient.

based Information Live Flow.png

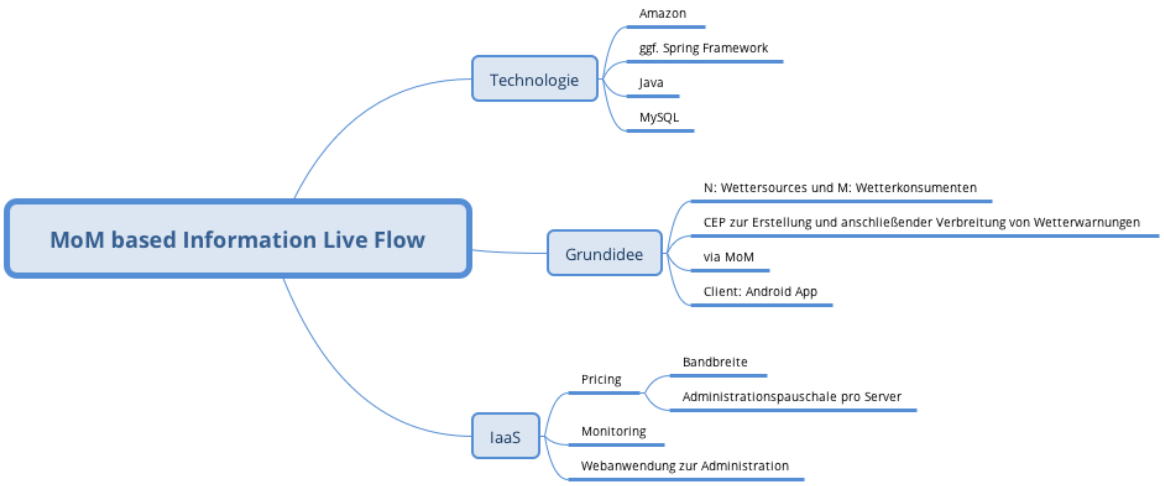

Abbildung 1: Brainstorming

Die folgende Abbildung 2. zeigt die grobe System-Architektur. Die wichtigsten Bestandteile dieser sind:

- Sensoren (Dateninput)
- MoM
- CEP
- Database
- Android-Client
- Web-Client

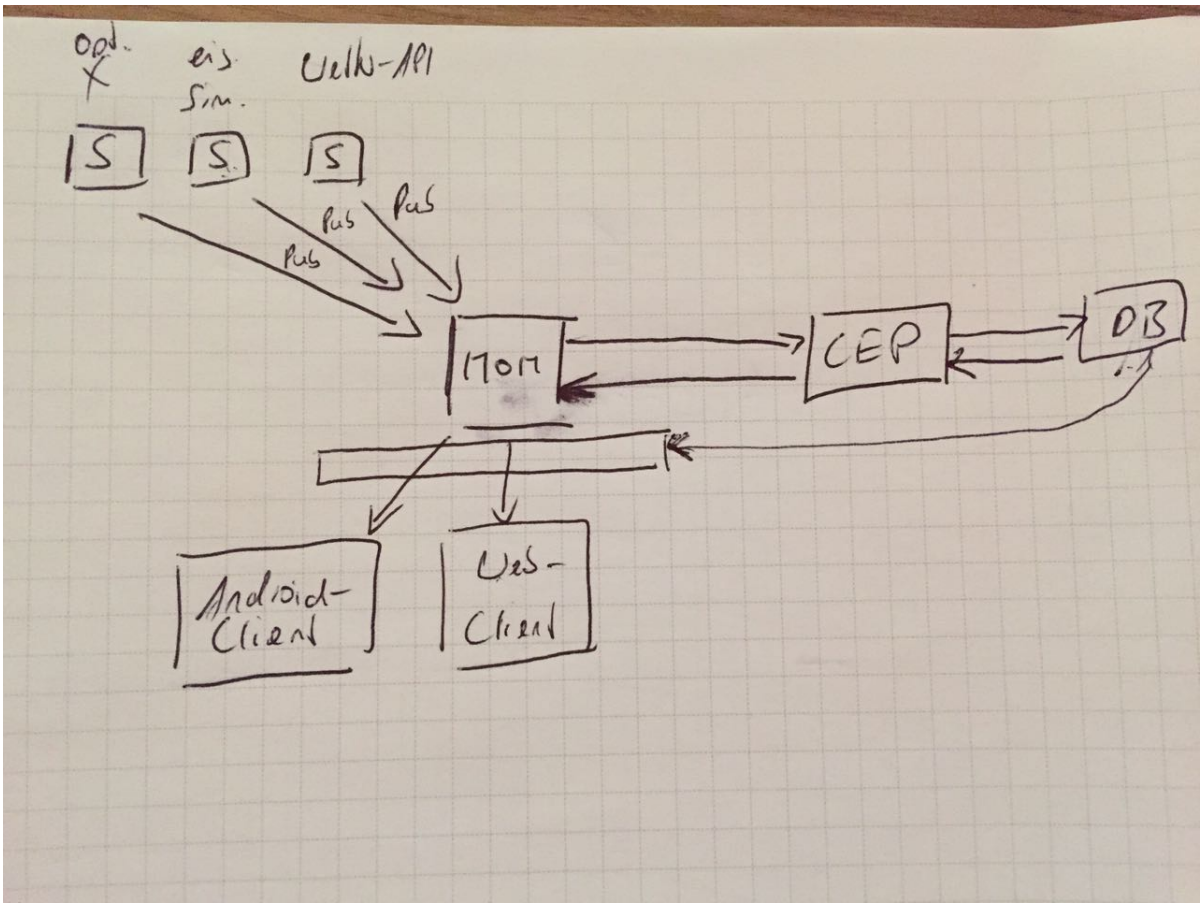

Abbildung 2: Grobe System-Architektur

2.2 Anforderungsdefiniton

Aus dem im letzten Abschnitt allgemein Beschriebenen können folgende grundlegenden nicht-funktionalen und funktionalen Anforderungen an ein System abgeleitet werden.

Die folgende Tabelle beschreibt die Kernanforderungen jeweils getrennt in nicht-funktionale und funktionale Anforderungen,

ID	Anforderung	Beschreibung
Nicht-funktionale Anforderungen		
1.1	Multi-tenancy	Das System muss eine fehlerfreie Mehrnutzer Funktionalität erfüllen
1.2	Scalability	Das System muss in der Lage sein zu jeder Zeit skalieren zu können
1.3	Fault-tolerance	Das System muss auf Fehler von Nutzer in angemessenem Rahmen reagieren
1.4	Cloudarchitektur	Das System muss vollständig in einer Cloud-basierten Umgebung lauffähig sein
Funktionale Anforderungen		
1.4	Binäre Daten annehmen	Das System soll binäre Daten annehmen können
1.5	Daten aufbereiten	Die Daten werden nach einem Muster aufbereitet
1.6	Daten abonnieren	Die Daten können von Abonnenten bezogen werden.
1.7	Nutzerverwaltung	Die User sollen in einer Datenbank verwaltet werden können.
1.9	Statistikfunktion	Die gespeicherten Daten sollen mit Hilfe einer Statistikfunktion aufbereitet werden können.
1.9	Datenquelle	Die Daten sollen via Wetter App bezogen werden können
2	Datenquelle	Die Daten sollen mit einem Generator fingiert werden können
2.1	Complex Event Processing	Aus den analysierten Daten sollen mit Hilfe von CEP verschiedene Ereignisse ausgelöst werden können.

Tab. 1: Funktionale Anforderungen an den Prototyp

3 Testing

Im letzten Abschnitt dieser Projektbeschreibung wird auf das Testing eingegangen. Hierbei wird zwischen den Kernpunkten Skalierbarkeit und Ausfallsicherheit/Multi-Tendency unterschieden, auf diese wird in den folgenden Abschnitten eingegangen.

3.1 Skalierbarkeit

- Normallast: definierter Input durch Wetterdaten API mit bestimmter Clientzahl für Performancebalancing Bestimmung von Grenzwerten (Scalingparameter) für Optimale Systemsicherheit (Datenkonsistenz/Datentransferraten etc.)
- Client - Stresstest: definierte Menge an Topics werden zur Verfügung gestellt. Schrittweise wird die Anzahl der Clientanschlüsse (virtuell) erhöht. Mithilfe der Scalingparameter (Normallast) muss das Cloud System weitere Ressourcen in Anspruch nehmen und der Output muss bewältigt werden(Datensicherheit /Zugriffsteuerungen/etc.) können. Der Latenzvergleich mit optimaler Normallast ergibt das Maß für Skalierbarkeit des Systems.
- Inputdaten - Stresstest: eine definierte Anzahl an Clients verbindet sich mit dem System und bezieht Daten aus der MoM. Mit Hilfe einer eigenen Softwarelösung werden nun fingierte Wetterdatenmengen in das System (MOM) eingespeist.
  - Skalierungsparameter (Normallast) Input muss bewältigt werden
  - Latenzvergleich (Datentransfer von Input bis Output auf einem Client) mit Optimaler Normallast ergibt Maß für Skalierbarkeit des Systems
- Abnahmetest: Test und finale Analyse des gesamten Cloudsystems Verschiedene Szenarien (Normallast, Client-Veränderungen, InputDaten-Veränderungen) werden in verschiedenen Kombinationen und Ausprägungen in das System gespeist.
- Finale Testinganalyse (Gesamtsystem)
- Modulparameter Fine-tuning der AWS Cloud Architecture

3.2 Ausfallsicherheit/Multi-Tendency

- MoM- Clients: Topics von anderem User dürfen nicht zur Verfügung gestellt werden. Zeitlich simultane Abfrage gleicher oder minimal abweichender Topics.
  - Test des Zugriffsverfahrens IAM (Amazon Web Services Identity and Access Management) Hinweis: Analog zu Keystone Identity Modul (OpenStack)
- CEP - Datenbank:Datenkonsistenz bei Lastverhalten (siehe Szenarien der der Skalierbarkeit)
  - Prüfung der Ergebnisse (Topics)
  - Prüfung über Statistikvergleiche möglich (Rahmen für zu erwartenden Wert versus geliefert/berechneter Wert)
- Clients - Datenbank: User hat die Möglichkeit ein Userprofil anzulegen (Suchkriterien zu speichern/Statistiken anzusehen)
  - Test des Userzugriffs Datensicherheit (Test der Oracle Userrollenzugriffe bzw. Test der Einstellungen im Modul IAM)