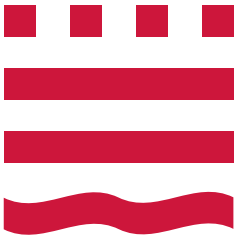

Technische Hochschule Brandenburg
University of Applied Sciences

BACHELORARBEIT

Serverless / Serverlose Architekturen für Konventionelle Webanwendungen

Vorgelegt von: Dragoljub Milasinovic
Matrikelnummer: 20140076

am: XX. Monat XXXX

zum
Erlangen des akademischen Grades

BACHELOR OF SCIENCE
(B.Sc.)

Erstbetreuer: Prof. Dr.-Ing. Schafföner
Zweitbetreuer: Jonas Brüstel, M.Sc.

Inhaltsverzeichnis

1 Quellen

OpsWorks AWS :: Deployment Strategy Cloud Design patterns :: Profi Patterns Man Trade Offs Arch :: Auswertung + Design guide <- self-adaptive arch?? AWS Sol. Arch :: Best Practices AWS vs Patterns general Amazon Web Services in Action :: Best Practices Arch Impl Cloud Design-Patterns for AWS :: Patterns list Serverless Arch AWS :: Main :: lambda: compute as a back end

2 Einleitung

Idee-Ausführung-Markt

> *" An idea is not a mockup*
> *A mockup is not a prototype*
> *A prototype is not a program*
> *A program is not a product*
> *A product is not a business*
> *And a business is not profits."*
>
> Balaji S. Srinivasan

Die vorantreibende Aspekte solcher Zustandsmaschine sind die Ausführung/Umsetzung der Idee bis zum Produkt und derer Beziehung zum Markt. Deren Details sind jedoch unbekannt und variabel.

Die Faktoren am Anfang einer technologischen Umsetzung einer Idee sind: Time-To-Market Kost of Human Resources:: Skill shortage Prof of Concept Technical technological details Profitability

2.1 Motivation

Auf dem Weg zur technologischen Umsetzung einer neuen Idee liegen unbekannte Schwierigkeiten mit der Entscheidungen über die Architektur der Anwendung/Projekt/Umsetzung?, der Drittanbieter von Software, der Auswahl der Infrastruktur usw. Schwierigkeiten die von spezialisierten Kompetenzen, Fertigkeiten und „Know-How“bedürfen. Gehören jedoch nicht immer zum Problem des Domäns der Anwendung.

Für dieses Problem wurde „FaaS“als Lösung unter der Rubrik „Serverless“von den Hauptanbieter von „Cloud“Technologien vorgestellt.

2.2 Ziel

Im Rahmen des Cloud-Computing handelt es in dieser Arbeit um eine Untersuchung der Serverless Architekturen am Beispiel einer Konventionellen Webanwendung. Dabei wird besonders geachtet ob und wie solche Technologien die Umsetzung erleichtern. Die Entwurfsmuster und die Kernfunktionalität werden mit ausschließlich Serverless Technologien am Beispiel von KOMA mit AWS umgesetzt.

@acronym MVP Minimal Viable Product : [Rad16] Als Ergebnis wird ein MVP Minimal Viable Product in form eine @acronym SPA Single Page Application vor zur Verfügung gestellt.

Start::Chars: Arch + Domän Flexibilität - Schnelle Arch Änderungen

Umsetzung der Kernfunktionalität einer Beispielanwendung mit ausschließlich „Serverlosen“Architekturen. wenn Zeit: Identifizieren von unverzichtbare Generische Funktionen für Serverless Anwendungen.

2.3 Aufbau der Arbeit

Zuerst wird den Leser in die Serverless 3.1 Technologien eingeführt, das Programmiermodell vorgestellt und die Entscheidungsprinzipien erläutert. Als Zweites wird KOMA 3.2 als vorläufiges Beispielanwendung und deren Anforderungen vorgestellt. Eine Zahl von möglichen Kombinationen von aktuellen Technologien mit Serverless werden beispielhaft dargestellt. Hinzu wird derer Analyse und Umsetzung angeführt? durchgeführt.

3 Grundlagen

Serverless ist ein @Glossar Web Dienst/Service von Cloud-Anbieter, wird auch als @Glossar FaaS bezeichnet. Deren Serverinfrastruktur wird vom Cloud-Anbieter wie Amazon Web Services verwaltet. Komplexe Probleme wie horizontale und vertikale Skalierbarkeit, Fehlertoleranz, Flexibilität werden von Kunden nur noch nach bedarf Konfiguriert.

3.1 Serverless

Prinzipien von Serverless Architeturen: [Sba17] Rechen-Dienst nach Anfrage dass in isoliert, unabhängig und granular ausgeführt wrid. . . .

Programmiermodell Dienste werden nach Anfrage ausgeführt. Die Kosten werden nach Ausführung abgerechnet.

One size fits NOT all

Vergleich mit Microservices.

Stand der Technik: Vorgehensweise bei Traditionelle Webanwendungen: Software Architektur: "What's important". Frühe, un-/schwer- veränderbare Entscheidungen. Web Services are processes that expose their interfaces to the Web so that users can invoke them. Facilitate service discovery and meaning encoded in schemas

Design: Lambda Orchestrator -> Pool of Lambdas to use

3.2 KOMA

Beispiel Anwendung

„KOMA“ist ein Akronym für Kompetenz-Matrix. Die Umsetzung der Anwendung soll die von einem Individuum erworbene und zu erwerbenden Fertigkeiten, Kompetenzen und deren Niveau nachvollziehen. <– ?

Das Modell von KOMA basiert auf dem Grundmodell von @ref „European Qualifications Framework Semantics“und dem deutschen Qualifikationsrahmen.

Ein Nutzungs-Fall aka. Use-Case: -Als Professor, will ich eine Auflistung der erworbene Fertigkeiten einer Klassenstufe abrufen können.

-Als Professor, will ich das Kompetenzniveau einer Kompetenz und derer Fertigkeiten abrufen können.

3.3 Anforderungen Analyse

Von Browser abrufbar.

Private Datenspeicherung. Daher Login.

Zukünftige Erweiterungen berücksichtigen.

Konsistenz Prüfung mit Unterschiedlichen Konnotationen zwischen dargestellten Konzepten. <– zu viel

3.4 Serverless Kombiniert

Spring Boot + Lambda Wildfly Swarm REST + Lambda Django + Lambda

3.5 Datenhaltung Analyse und Auswahl

–> Tabelle Vergleich:: Relationale : NoSQL : Ontology Konsistenz Erweiterbarkeit Migration Bedeutung

Da die Konnotationen von beispielsweise Schlusselkompetenzen von Kontext zu Kontext unterschiedlich sind, bietet sich eine Ontologische Datenspeicherung an. Extensible, Migriation ok. Consistency ko.

Um die Konsistenz der Semantic desRDF Grundmodells zu bewahren wird ein Relationale schema für die Ontologie benutzt.

4 Umsetzung

4.1 Komponenten

@Diagramm

4.2 Einloggen

Einloggen: 0Auth Google gibt token, der wird in Lambda überprüft, Session in oauth.com verwaltet Query: SparQL ?x, ?y, ?z WHERE ... Datenspecherung Architektur: DynamoDB: speichert :individual als Schlussel und seine relative URL Ś3: speichert die .owl Dateien.

Lambda Funktion: Maps zwischen S3 und DynamoDB.

4.3 Patterns

Valet Key [HSB+14]

Static Content Hosting ok

Sharding ok

Compute Resource Consolidation

Command and Query Responsability Segregation CQRS <- readS3UpdateDynamo.js

5 Ergebnis und Auswertung

Vorteile

Automatische Skalierung und Fehlertoleranz Automatisches Kapazitätsmanagement Flexible Ressourcenverwaltung Schnelle Bereitstellung der Ressourcen Exakte nutzungsabhängige Abrechnung der Ressourcen Konzentration auf den Kern des Source-Codes Nachteile:

Kontrollverlust Erhöhtes Lock-in Risiko

kurzlebige konfigurationen herausfinden ?? tracking? viel Konfiguration, kaum Konveniton -> .json 4 everything local testing braucht event-symulation.json

6 Zusammenfassung und Ausblick

Beispielhaftes Bild Abbildung 6.1

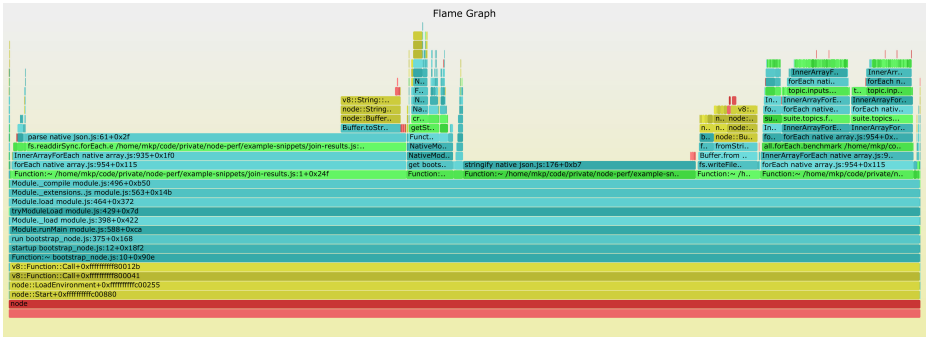

Abbildung 6.1: Beispiel Flame-Graph eines Node.js Skripts

Beispielhaftes Code-Snippet siehe Listing 6.1.

Listing 6.1: Aufnahme der „real“-Zeit

```
START=$(date +%s.%N)
node ${JS_FILE}
END=$(date +%s.%N)
DIFF=$(echo "$END - $START" | bc)
```

Hier kommt eine Bibliography-Referenz: [BME+07]

Listings

Abbildungsverzeichnis

Abkürzungen

GC Garbage Collection

„Garbage Collection“ bezeichnet die automatische Speicherwaltung zur Minimierung des Speicherbedarfes eines Programmes. Garbage Collection (GC) wird zur Laufzeit durch Identifikation von nicht mehr benötigten Speicherbereichen ausgeführt. Im Vergleich zur manuellen Speicherverwaltung benötigt GC mehr Ressourcen.

Literaturverzeichnis

[BME+07] G. Booch, R. Maksimchuk, M. Engle, J. Conallen, K. Houston, and B.Y.P. D. *Object-Oriented Analysis and Design with Applications*. Pearson Education, 2007.

[HSB+14] A. Homer, J. Sharp, L. Brader, M. Narumoto, and T. Swanson. *Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications*. Patterns & practices. Microsoft Developer Guidance, 2014.

[Rad16] B. Rady. *Serverless Single Page Apps: Fast, Scalable, and Available*. Pragmatic programmers. Pragmatic Bookshelf, 2016.

[Sba17] P. Sbarski. *Serverless Architectures on AWS: With Examples Using AWS Lambda*. Manning Publications Company, 2017.

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Masterarbeit selbstständig verfasst, ausschließlich die angegebenen Hilfsmittel benutzt und sowohl wörtliche, als auch sinngemäße entlehnte Stellen als solche kenntlich gemacht habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Brandenburg an der Havel, XX. Monat 2017

Vorname Nachname