Nutzergesteuerte Datenwertschöpfung in heterogenen Cloud-Native Anwendungsumgebungen

Lukas Wala

Matrikelnummer: 807291

Große Seminararbeit

im Modul ’Business Intelligence & Analytics’

Bachelor of Science (B.Sc.)

im Studiengang Bachelor Informatik

der FOM Hochschule für Oekonomie & Management

Lukas Wala
Zeppelinstraße 14
91052 Erlangen

vorgelegt bei

MBA Karl-Heinz Leicht

Erlangen, 31.12.2025

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

BI Business Intelligence

API Application Programming Interface

1 Einleitung

Die Geschichte der Business Intelligence reicht weiter in die Vergangenheit zurück als moderne Informationssysteme, obwohl die beiden Begriffe in gegenwärtigen Diskussionen oft synonym miteinander verwendet werden.
Quelle: Historische Fragmente einer Integrationsdisziplin – Beitrag zur Konstruktgeschichte der Business Intelligence

Vor dem Hintergrund, dass alle Wirtschaftssektoren jedoch mittlerweile mit IT-Systemen verwickelt sind und bestimmte Branchen wie das Gesundheits- oder Finanzwesen überwältigend von diesen abhängig sind
Quelle: https://www.sciencedirect.com/science/article/pii/S0166497222001304

, ist die Auswertung der Effizienz und Resilienz von betrieblichen Prozessen durch Computer und Netzwerke nicht mehr wegzudenken. Damit BI-Lösungen den aktuellen Stand von Business Prozessen abbilden können, müssen sie mit der technischen Implementierung der Prozesse kompatibel sein. In diesem Bereich hat sich in den letzten 20 Jahren einiges getan. Entwickler und Anwendungsadministratoren konnten in diesem Zeitraum viele Paradigmenveränderungen durch virtuelle Maschinen, Containerisierung und Clouddienste beobachten und zu diesen beitragen. Die moderne IT-Infrastrukturlandschaft ist vor allem seit der Veröffentlichung von Container-Engines wie Docker und Container-Orchestrierungsplattformen wie Kubernetes eine komplett andere wie zu Hochzeiten der Bare-Metal-Server, Terminalserver und Mainframes.

Um mit diesen Veränderungen schritthalten zu können, haben BI-Anwendungen wie Prometheus(Metriken, 2012), Grafana(Dashboards, 2014) oder Metabase(Datenbankvisualisierung, 2015) vorgefertigte Formate für diese neueren Plattformen veröffentlicht. Metabase bietet neben dem eigenen Clouddienst direkt Anleitungen für Self-Hosting per Docker Container. Prometheus und Grafana sind sogar in einem gemeinsamen "Helm-Chart"namens "kube-prometheus-stack"verpackt. Helm ist ein Paketmanager für Kubernetes Anwendungen und vereinfacht es Nutzern komplexe Anwendungen mit mehreren Kubernetes Ressourcen zu installieren. Alle diese Produkte sind aber grundlegend plattformagnostisch, was zwar zu mehr Flexibilität aber auch geringerer Intergration mit der Plattform führt. Da Kubernetes als modernes Produktivsystem jedoch eine professionelle und frei erweiterbare API bietet, kann diese potenziell genutzt werden um direkt BI-Lösungen zu deklarieren.

Ein weiterer Schwachpunkt von etablierten BI-Lösungen, der vor Allem bei Prometheus und Grafana auffällt, ist die hohe Komplexität bei der Erstellung von übersichtlichen Dashboards. Prometheus sammelt Metriken aus sämmtlichen Anwendungen die diese standard-

mäßig anbieten, jedoch wird hierbei keine Beschreibung mitgeliefert, was die Metriken genau bedeuten oder wie diese bestenfalls in einem Dashboard visualisiert werden können. Falls also kein fertiges Dashboard von den Entwicklern einer Anwendung mitgeliefert wird, fällt die Arbeit dieses zu entwickeln auf die Endnutzer. Da Teammitglieder, die diese Metriken für geschäftsbezogene Entscheidungen verwenden nicht immer speziell geschult sind, kommen Infrastrukturadminstratoren oder ein DevOps-Team für technische Defizite auf und implementieren die Visualisierungen. Idealerweise ist die Überwachung von Anwendungen so einfach präsentiert, dass ungeschultes Personal die Erstellung von Dashboards intuitiv lernen kann.
Quelle: Microsoft Power BI oder so

1.1 Problemstellung

Da Kubernetes ein beliebtes Kompilierungsziel für Business Intelligence Anwendungen ist, aber diese Anwendungen die Möglichkeiten einer tiefen Integration, die durch den offenen Quellcode erreichbar ist, nicht ausnutzen, soll in dieser Arbeit durch eine Fallstudie untersucht werden ob eine direkte Erweiterung der Kubernetes API dabei helfen kann eine intuitive Datenwertschöpfung für Endnutzer zu ermöglichen.

Folgende Forschungsfrage soll in dieser Arbeit beantwortet werden: „Können im Cloudnative Bereich Self-Service Paradigmen der Anwendungsentwicklung auf Business Intelligence ausgeweitet werden um Endnutzern eine einfachere Erfahrung bei der Überwachung von Diensten zu ermöglichen‘"

1.2 Lösungsansatz

Durch explorative Auseinandersetzung mit Design Mustern für Kubernetes-Erweiterungen, soll überprüft werden ob Best Practices bei deren Entwicklung dabei helfen können eine flüssigere und simplere BI-Pipeline zu entwickeln. Es soll festgestellt werden ob Nutzern durch weniger Abstraktionsschichten die Erstellung von Dashboards vereinfacht werden kann. Die Ergebnisse dieser Fallstudie sollen dabei helfen zukünftige Monitoringlösungen für Kubernetes und ähnliche Orchestrierungsplattformen zu konzipieren.

1.3 Aufbau der Arbeit

Vorerst werden grundlegende Begriffe aus dem Containerökosystem erklärt und darauf aufgebaut, wie aktuelle BI-Lösungen mit ihren Aufgaben umgehen. Anschließend wird anhand einer Fallstudie eine beispielhafte Anwendung geschrieben, die nachfolgend auf Schwachstellen und Vorteile bewertet wird. Zuletzt werden die daraus resultierenden Ergebnisse kritisch gewürdigt und die Forschungsfrage bestätigt oder widerlegt.

2 Theoretischer Hintergrund

2.1 Cloudnative Anwendungslandschaft

2.1.1 Microservices

2.1.2 Containerumgebungen

2.1.3 Kubernetes

2.2 Open Source BI-Lösungen

3 Fallstudie

3.1 Datenmodell

3.2 Kubernetes Operator

3.3 Benutzerinterface

4 Einordnung und Bewertung der Erkenntnisse

5 Kritische Würdigung

A Google Platform Status

BigQuery-Abfrage:

```
#standardSQL
SELECT date, client, pct_urls, sample_urls
FROM ‘httparchive.blink_features.usage‘
WHERE feature = ’WebAssemblyInstantiation’
AND date = (
  SELECT MAX(date)
  FROM ‘httparchive.blink_features.usage‘
)
ORDER BY date DESC, client
```

B Konferenzvorträge

Jahr	Konferenz	Vortragende Person	Titel
2017	dotJS	Sean Larkin	Webpack+WebAssembly: Under the hood
	JSConf EU	Lin Clark	A Cartoon Intro to WebAssembly
	JSConf EU	Dan Callahan	Practical WebAssembly
	GOTO	Ben Smith	We Want WebAssembly
2018	dotJS	Tejas Kumar	A Quick Recap on WebAssembly
	JSConf EU	Emil Bay	Hand-crafting WebAssembly
	JSConf EU	Lin Clark	Baby's First Rust+WebAssembly module
	JSConf Asia	David Bryant	Enabling New Web Experiences
	JSUnconf	Johann Hofmann	Putting WebAssembly in your web app today!
	GOTO	Lin Clark	A Cartoon Quest: New Adventures for WebAssembly
2019	dotJS	Vlad Filippov	Into WebAssembly
	dotJS	Sven Sauleau	More WebAssembly in your JavaScript
	JSConf EU	Max Bittker	Simulating Sand: Building Interactivity With WebAssembly
	JSConf Korea	Istvan Szmozsanszky	A WebAssembly Field Guide easily worth like 70 bottle caps
	JSConf Asia	Kas Perch	WebAssembly: The Future of JS and a Multi-Language Web
	JSConf US	Florian Rival	Native Web Apps: React, JS & WebAssembly to rewrite native apps
	JSConf Hawaii	Lin Clark / Till Schneidereit	New Adventures for WASM
	GOTO	Dan Callahan	WebAssembly Beyond the Browser
2023	GOTO	Brian Carroll	WebAssembly in Production: A Compiler in a Web Page
	YOW!	Katie Bell	Don't Trust Anything! Real-world Uses For WebAssembly
2024	dotJS	David Flanagan	The Future of Serverless is WebAssembly

C R-Skript