FernUniversität Hagen

– Faculty of Mathematics and Computer Science –

Dynamically allocation of Datasets to Datastores in a Polystore Environment

Presented by

Daniel Langhann

Matriculation number: 3788687

Supervision : Prof. Dr. Ute Störl

DECLARATION

I declare that I have written the seminar paper independently and without any unauthorized assistance from third parties.

In doing so, I have only used the sources and resources specified and have identified the passages taken from these sources, either verbatim or in meaning, as such.

The assurance of independent work also applies to any drawings, sketches, or graphic representations contained therein.

The paper has not been submitted in the same or similar form to the same or any other examination authority nor has it been published.

By submitting the electronic version of the final paper, I acknowledge that it may be checked for plagiarism using a plagiarism detection service and will be stored exclusively for examination purposes.

Hagen, 01. Feb 2025

Daniel Langhann

ABSTRACT

In der vorliegenden Masterarbeit wird das Thema Dynamische Allokation von Datasets zu Datastores behandlet. Die Motivation für die Arbeit ist es, einen Vorschlag zu unterbreiten, wie in einem Umfeld aus heterogenen Datasets, für die mehr als ein Datenbank-System ideal ist, eine dynamische Zuweisung und Neuzuweisung von Datasets zu Datastores zu realisieren. In der Vergangenheit wurde bereits viel an dem Themenbereich Polystores geforscht. Polystores beinhalten mindestens zwei verschiedenartige Datastores also letztendlich Datenbank-Systeme um hetoregene Datasets, zum Beispiel transaktionsbasierte Daten auf der einen Seite und aggregierte Daten für die Analyse von zum Beispiel sogenannten Key Perfomance Indikatoren (KPI) auf der anderen Seite optimal verarbeiten zu können. Eine bislang ungelöste Herausforderung in einem polystoren Umfeld ist, die Reaktion auf sich verändernde Workloads in der Applikaton bzw. im Gesamtsystem. Wie reagiert man bezogen auf das oben genannte Beispiel, wenn in dem transaktionsbasierten Bereich des Gesamtsystems vermehrt analytische Abfragen entstehen, also GET Requests mit langen und sehr langen Laufzeiten. Dann wäre es wünschenswert, dass diese Abfragen zukünftig über den Datastore, der sich auf analytische Abfragen fokussiert, umgeleitet werden.

Genau hier setzt die vorligende Arbeit an. Es soll ein Vorschlag unterbreitet werden, wie man in einem polystoren Systemumfeld, dynamisch auf sich verändernde Workloads reagieren kann. Der erste Schwerpunkt beschäftig sich damit, bezogen auf gegebene Anforderungen an ein Gesamtsystem eine initiale Zuweisung von Datasets zu Datastores vorzuschlagen. Es wird ein Prototyp entwickelt, der algorithmisch eine entsprechende Zuweisung ermittelt. Der zweite Schwerpunkt fokussiert sich auf die permanante und dynamische Analyse von Workloads um bei sich verändernden Paramtern bezogen auf den initialen Zustand mit einer Re-Allokation der Datasets zu reagieren bzw. einen angepassten Vorschlag zu unterbreiten. Für diesen zweiten Schwerpunkt wird ebenfalls ein Prototyp entwickelt, der als Input die Analysedaten des Workloads erhält, und beim Erreichen bestimmter Schwellwerte einen Vorschlag für eine Neuzuweisung unterbreitet. Für die konkrete Ausführung der angepassten Zuweisungen wird ein theoretischer Vorschlag gemacht und diskutiert.

Die Arbeit schließt wiederum mit einer Zusammenfassung der Ergebnisse bezogen auf die Ausführung und Testung der Prototypen. Darüber hinaus sollen Vorschläge unterbereitet werden, an welchen Themen noch gearbeitet werden sollte, um tatsächlich ein komplett autonom arbeitendes Gesamtsystem auf der Basis von Polystores zu realisieren.

CONTENTS

ABKÜRZUNGSVERZEICHNIS

API Application Programming Interface

UML Unified Modeling Language

1

EINLEITUNG

Das Thema Polystores ist bereits seit mehr als zehn Jahren Gegenstand aktiver Forschung. Dies ist relativ einfach damit zu begründen, dass die digitale Welt immer vielschichiger und komplexer wird. Die Menge an zu verarbeitenden Daten steigt immer weiter. Wo es früher Ziel war, Datenbank Konzepte zu entwerfen und zu implementieren, die möglichst sparsam mit der knappen Ressource Speicherplatz umgehen, liegt heute der Fokus tendenziell auf Performance. Empfehlungs-Systeme müssen in der Lage sein in Echtzeit oder sehr schnell auf Basis von Inputs sinnvolle Vorschläge zum Beispiel im Kontext If a customer **C** bought item **A**, then they also bought item **B** or item **C** or ... :

$$\forall c \in \mathcal{C} \quad \Big(\text{Bought}(c, A) \implies \big(\text{Bought}(c, B) \vee \text{Bought}(c, C) \vee \cdots\big)\Big)$$

Ein solche Anfrage kann nur sinnvoll und performant beantwortet werden, wenn ein entsprechender Algorithmus sehr schnell Input von einer geeigneten Datenbank erhält. Eine solche Datenbank, kann zum Beispiel eine Graph-Datenbank sein. In zum Beispiel einem Web-Shop, wo ein solches Empfehlungs-System implementiert ist, entstehen häufig auch typische Datenbank Transaktionen sog. Create, Read, Update, Delete Operationen (CRUD). Solche Transaktionen können wiederum tendenziell besser über eine klassiche SQL Datenbank abgewickelt werden. Eine solches System kann als ein Polystore bezeichnet werden, da es, um performant zu funktionieren, mindestens zwei verschiedene Datenbank-Systeme bzw. Data-Stores benötigt. Teilweise sind die Übergänge fließender als in dem oben genannten Beispiel des E-Commerce Systems mit einer Datenbank für die Empfehlungs-Engine und einer Datenbank für den transaktionsbasierten Teil des Systems. So lassen sich zum Beispiel so genannte Datenströme genau so gut in eine zum Beispiel dokumentenbasierte Datenbank abwickeln wie klassiche CRUD Operationen. Es wäre denkbar, dass beide Teile eines fiktiven Systems über zum Beispiel eine MongoDB abgebildet werden. Hier wäre es sinnvoll zu messen ob ein System welches aus zwei oder mehr verschiedenen Services besteht, die sich einen Datastore bzw. ein Datenbank System teilen, bei einem sich dynamisch ändernden Workload, eine Neuzzuteilung Service bzw. Dataset zu Datastore sinnvoll ist. Dies kann zum Beispiel sinnvoll sein, wenn bestimmte Transaktionen oder Commits immer mehr Zeit in Anspruch nehmen, was sich wiederum negativ auf die Benutzererfahrung bzw. die Performance des Systems auswirkt. Grundsätzlich geht es also darum, dass in einem System S mindestens zwei verschiedene Services S existieren, die jeweils einem Datastore D zugeordnet sind und einen gewissen Workload W aufweisen. Ändert sich der Workload W bzw. werden bestimmte Grenzwerte Th überschritten, sollten die Services bzw. der betroffene Service einem anderen,

gegebenfalls besser geeigneten Datastore zugeordnet werden. Dies sollte nur passieren, wenn der Service S in Verbindung mit dem neu zugewiesenen Datastore einen verbesserten Workload aufweist. Grundsätzlich kann der vorbezeichnete Sachverhalt wie folgt beschrieben werden:

SERVICE REASSIGNMENT BASED ON WORKLOAD

We define the following:

- **Services:**

$$\mathcal{S} = \{S_1, S_2, \ldots, S_n\} \quad (n \geq 2)$$

- **Datastores:**

$$\mathcal{D} = \{D_1, D_2, \ldots, D_m\}.$$

- **Workload Function:**

$$W : \mathcal{S} \times \mathcal{D} \to \mathbb{R}_{\geq 0},$$

  where $W(S, D)$ measures the workload of service S when using datastore D (lower is better).

- **Thresholds:** Let $Th \geq 0$ be the workload threshold, and let $\epsilon > 0$ be a parameter defining a significant change in workload over time.

Let $A : \mathcal{S} \to \mathcal{D}$ be the current assignment function, meaning that a service S is assigned to a datastore $A(S)$.

For a given service S, if either:

$$\left|W(S, A(S))_t - W(S, A(S))_{t-1}\right| \geq \epsilon$$

or

$$W(S, A(S)) \geq Th,$$

and there exists another datastore $D' \in \mathcal{D}$ such that:

$$W(S, D') < W(S, A(S)),$$

then we reassign service S to D'.

This rule can be written as:

$$A'(S) = \begin{cases} D', & \text{if } \left(\left|W(S, A(S))_t - W(S, A(S))_{t-1}\right| \geq \epsilon \text{ or } W(S, A(S)) \geq Th\right) \\ & \quad \text{and there exists } D' \in \mathcal{D} \text{ with } W(S, D') < W(S, A(S)); \\ A(S), & \text{otherwise.} \end{cases}$$

Es gilt also permanent zu überprüfen wie ein System aus Datasets bzw. Services und Datastores aufgebaut sein sollte um die Anforderungen an dieses System optimal zu erfüllen. Genau diesem Thema widmet sich die forliegende Arbeit.

1.1 MOTIVATION

Es existieren bislang nur wenig bis gar keine polystoren Systeme die es über den Status Prototyp geschafft haben. Das Projet was wohl am fortgeschrittensten ist ist Polyphenie DB. Ein Erklärungsansatz für den mangelhafte Verbreitung von Polystore Systemen und oder deren produktiven Einsatz ist, dass die Implmentierung immer noch schwierig und aufwendig ist.

Ein anderer Erklärungsansatz ist, dass die Vorteile eines Polystores erst zum tragen kommen, wenn dieses in einem dynamischen Umfeld bzw. System zum Einsatz kommt. In einem dynamischen Umfeld ändert sich der Workload und damit ist wie bereits beschrieben die initiale oder aktuelle Zuordnung von Datastores zu Datasets nicht mehr valide bzw. optimal.

Nun ist es schwer vorstellbar, dass in einem produktiven Betrieb permanent manuell Datasets anderen Datenbanken und Datenbank Systemen zugeordnet werden. Ein solcher Eingriff ist komplex, erfordert eine sorgfältige Vorbereitung und beansprucht in den meisten Fällen einen nicht unerheblichen Teil der Ressourcen (insbesondere Personal wie Datenbank Administratoren und Backend-Entwickler etc.). Des weiteren geht mit einer solchen Umstellung ein nicht zu unterschätendes Risiko einher. Dieses Risiko kann durch bestimmte Verfahren wie das Testen der Umstellung in einer gekapselten Testumgebung, einer schrittweisen Umstellung etc. minimiert werden, dennoch sind Anpassungen am Datenmodell bzw. an der Persistenz-Schicht eines Systems immer heikel und unterliegen häufig einem sehr statischen Setup.

Des Weiteren stellt sich immer die Frage wenn festgestellt wird, dass eine Zuordnung Datasets zu Datastores nicht mehr optimal ist, was die bessere Lösung wäre und wie sich der Workload durch eine Unmstellung potentiell verändern würde. Eine manuelle Überprüfung der Workloads ist zwar denkbar aber schwer mit dem Alltag von Entwicklungs-Teams vereinbar. Wenn man feststellt, dass die Performance aber auch andere parameter wie Verfügbarkeit, also letztendlich Service Level Agreements (SLA) nicht mehr den Anforderungen entsprechen, stellt sich die Frage, wie das System umgestellt werden sollte oder umgestellt werden kann um die Anforderungen wieder zu erfüllen.

Die weiterführende Vision bzw. Motivation dieser Arbeit ist, eine Polystore System welches auf Basis von Workloads, Applikationsparametern und SLA automatisch eine initial Zuordnung von Datasets zu Datastores vornimmt und dieses Setup kontinuierlich und automatisiert überwacht und im Fall von Abweichungen sich selbst optimiert. Ein solches System wäre gerade für komplexe Anwendungen mit heterogenen Anforderungen an die entsprechenden Datastores ein Mehrwert und trägt der Erkenntnis Rech-

nung das im Bereich Datastores und Datenbank eine One Fits All Lösung nicht bzw. bislang niht existiert. So entsteht für den Benutzer ein optimiertes Nutzererlebnis und das Entwicklungs-Team kann sich voll auf die Implmentierung von User Interfaces und Business Logik fokussieren.

1.2 ZIEL DER ARBEIT

Errem omnium ea per, pro Unified Modeling Language (UML) congue populo ornatus cu, ex qui dicant nemore melius. No pri diam iriure euismod. Graecis eleifend appellantur quo id. Id corpora inimicus nam, facer nonummy ne pro, kasd repudiandae ei mei. Mea menandri mediocrem dissentiet cu, ex nominati imperdiet nec, sea odio duis vocent ei. Tempor everti appareat cu ius, ridens audiam an qui, aliquid admodum conceptam ne qui. Vis ea melius nostrum, mel alienum ac elit id nibh pretium pulvina euripidis eu.

1.3 GLIEDERUNG

nulla fastidii ea ius, exerci suscipit instructior te nam, in ullum postulant quo. Congue quaestio philosophia his at, sea odio autem vulputate ex. Cu usu mucius iisque voluptua. Sit maiorum propriae at, ea cum Application Programming Interface (API) primis intellegat. Hinc cotidieque reprehendunt eu nec. Autem timeam deleniti usu id, in nec nibh altera.

2

TERMINOLOGISCHE GRUNDLAGEN

3

INITIAL ZUORDNUNG VON DATASETS ZU DATASTORES

4

DYNAMISCHE ALLOKATION UND RE-ALLOKATION VON DATASETS ZU DATASTORES

5

DATENMODELL UND DATENBANKSEITIGE UMSETZUNG VON DATASTORE RE-ALLOKATIONEN

6

SCHLUSSBETRACHTUNG