Seminararbeit im Studiengang Angewandte Informatik (BSc)

Self-Adaptive Architecture in Stream Processing

Version 1.0 vom 1. Januar 2020
(Vor Abgabe entfernen)

Leon Meister

285631

meister@uni-hildesheim.de

Betreuer:
MSc Cui Qin, SSE

Arbeitsgruppe Software Systems Engineering • Institut fï¿½r Informatik
Universität Hildesheim • Universitätsplatz 1 • D-31134 Hildesheim

Eigenständigkeitserklärung

Erklärung über das selbstständige Verfassen von "Self-Adaptive Architecture in Stream Processing"

Ich versichere hiermit, dass ich die vorstehende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der obigen Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich in jedem Fall durch die Angabe der Quelle bzw. der Herkunft, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet und anderen elektronischen Text- und Datensammlungen und dergleichen. Die eingereichte Arbeit ist nicht anderweitig als Prüfungsleistung verwendet worden oder in deutscher oder einer anderen Sprache als Veröffentlichung erschienen. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschung behandelt werden.

Hildesheim, den 1. Januar 2020

Leon Meister

Kurzfassung

Eine kurze Zusammenfassung der Arbeit, die Interesse beim Leser wecken soll.

Abstract

Gerne zusätzlich oder alternativ in Englisch.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcode-Verzeichnis

Abkürzungsverzeichnis

DPS	Data Stream Processing
EDF	Elastic and Distributed DSP Framework
IDC	International Data Corporation
SPS	Stream Processing System

1 Introduction

Situation nowadays -> Lots of data (industry 4.0, other use cases, etc.)

Motivation Goal -> Research question definieren und spezifizieren, discuss the different architectures Struktur erlÃ¤utern

The advancements in technology of the past decades has lead to enormous data creation. Technology has become ubiquitous, with the evolution of cell phones to smartphones, the digitilization of industrial processes, Industry 4.0 and the increasing amount of “smart“ devices, causing creation of information to grow exponentially. It is estimated that the Global Datasphere will reach the size of 175 zettabytes by 2025, as shown in figure 1.1.

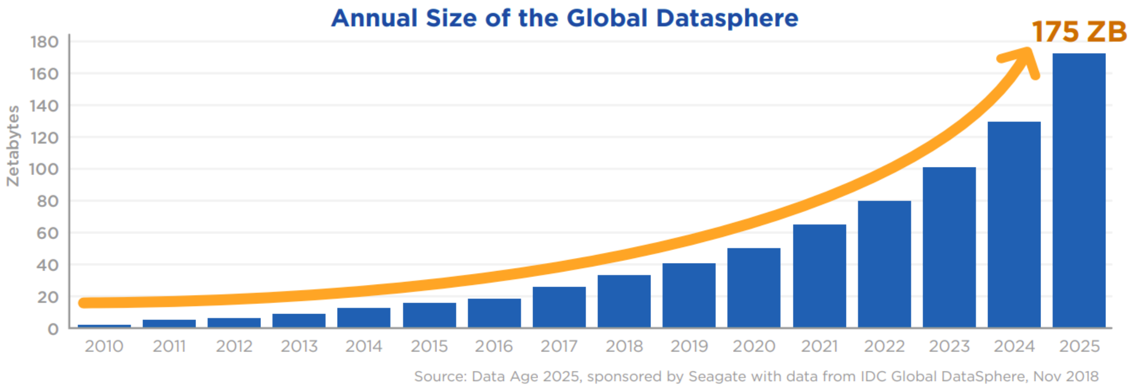

Abbildung 1.1: The Growth of the Global Datasphere [RGR18, p.6]

Data has become an important factor in decision making and optimization in virtually every industry, especially in finances. **TODO: Wieso?** The financial market is dominated by data driven decisions, with emphasis on data processing in a (near) real-time fashion. However, real-time data is becoming of importance in multiple sectors; the International Data Corporation estimates that real-time data will be responsible for a share of 30 percent of the total global datasphere by 2025, as shown in figure 1.2.

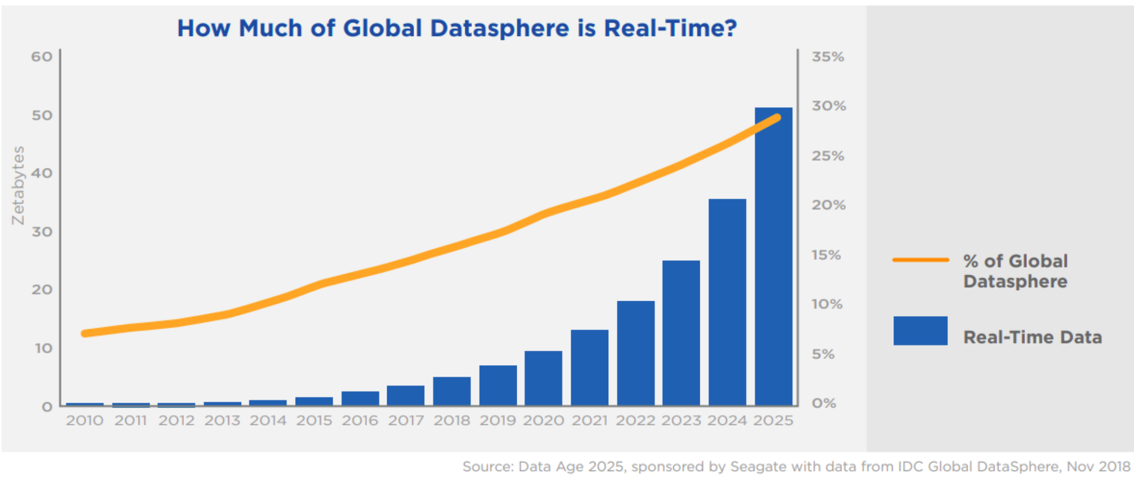

Abbildung 1.2: The growth of real-time data as part of the Global Datasphere [RGR18, p.13]

A global study led by IBM in 2012 has shown that 71 percent of the firms in the financial market use information (including big data) in order to achieve an advantage over their competitors, compared to 36 percent, which IBM has found in an earlier study conducted in 2010. [TSS13, p.1]

As it is no longer feasible to save all the data before then analyzing it (in batches), due to computational cost and lack of storage capacity, a new approach was designed in order to handle data in a (near) real-time fashion, Stream Processing Systems (SPS). **TODO: SPS Traditional DBMS.. another reason and then dsms**

2 Fundamental Concepts

In this chapter I will lay out the fundamental concepts, which are necessary in order to understand the subsequent chapters. First I will explain Stream Processing in 2.1, afterwards the concept of the MAPE-K Loop will be presented and explained in subchapter 2.2 followed by a brief explanation of self-adaptive systems in 2.3 to conclude this chapter.

2.1 Stream Processing

In this section I will split the concept of Stream Processing into three further components. In 2.1.1 I will then define Stream Processing Systems, explain how they work and give examplary fields of application. Afterwards in 2.1.2 I will then move onto the topic of Data Stream Management Systems and finally in 2.1.3 I will talk about some requirements that SPSs should meet.

2.1.1 Stream Processing Systems

In this subsection I will explain what SPSs are, what their input looks like, what SPSs are able to do and give an example. In the end of this subsection I will also go over the challenges that SPSs face.

Stream processing systems are fed continuous data streams, generated by data providers, for example GPS-sensors or EEG- and EKG-machines. A data stream S consists of a infinite, countable series of four-tuples $s :=< \nu, t^{app}, t^{sys}, b_{id} >, s \in S$ [PSZ15], with:

- $\nu \in \mathfrak{R}$ is a relational tuple
- $t^{app} \in T$, with T being the time domain, is a partially ordered application time
- $t^{sys} \in T$, with T being the time domain, is a totally ordered system time
- $b_{id} \in B$ is a batch id; batch B is a finite subset of S where all $b \in S$ have an identical t^{app}

[PSZ15] **How to cite this, as variable explanation is part of source too**

Each element s then gets processed by a number of operators, each doing their own independanst operation, e.g. filtering as shown in 2.1.

An SPS takes in one or multiple continuous streams of data, each element of the stream then gets processed by a number of operators and eventually, the SPS puts out a stream of processed data.

In order to increase efficiency, an SPS can, if (computational) resources are available, create replicas of operators to introduce parallelity. Conversely, if there is little input, it may also reduce the amount of replicas in order to save or free up additional resources, as shown in **??**.

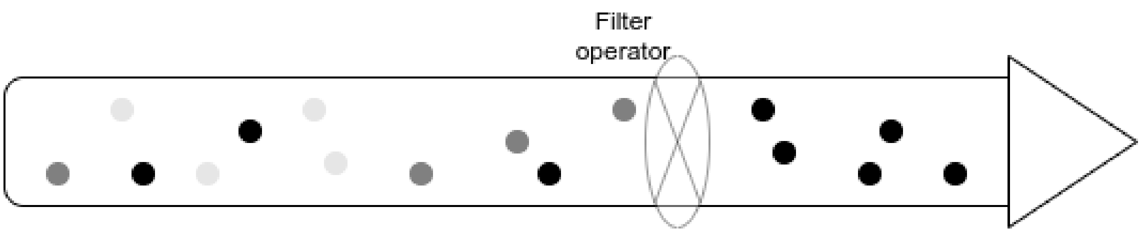

Abbildung 2.1: Stream being processed by SPS with a single filter operator, then being output for further processing

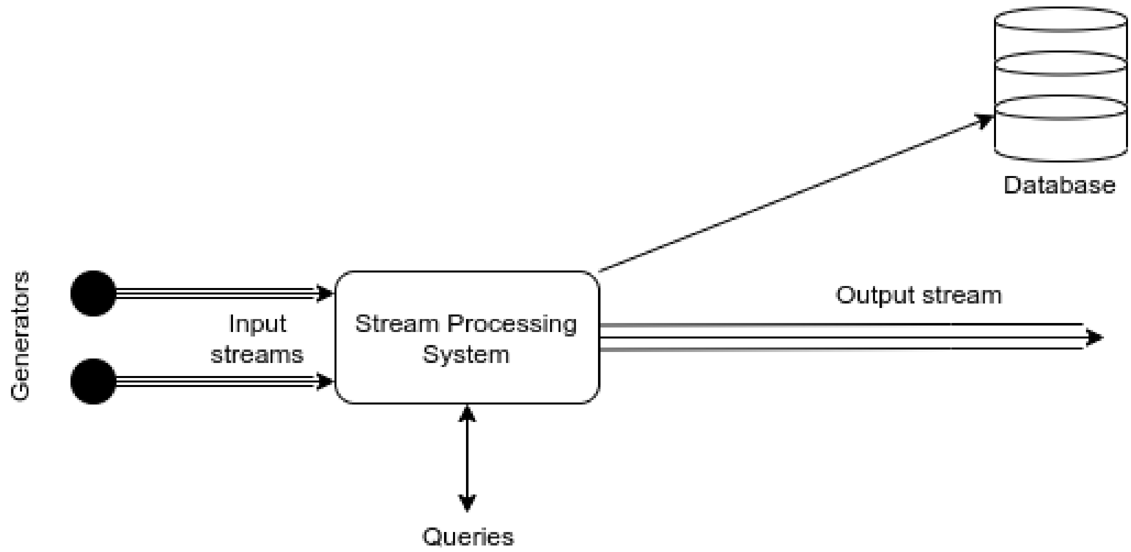

Abbildung 2.2: Overview of a basic Stream Processing System

2.1.2 Data Stream Management Systems

A Data Stream Management is a system designed to manage continuous data streams. Similar to a DBMS a DSMS can also be queried using a streaming query language, however, queries installed on a DSMS are continuous and will be executed as long as it is uninstalled. This leads to everchanging results, due to changing incoming data in contrast to querying a DBMS and receiving a static result.

Another important difference is that the data being fed into the system is not saved like in DBMSs but instead only synopses, in an attempt to summarize the data.

Most notably the data being fed into a DSMS varies greatly from the data being inserted into a DBMS; while a DBMS receives a finite predetermined amount of data, a DSMS could theoretically receive an infinite continuous stream. There are a few more functional differences, which Panigati, Schreiber and Zaniolo highlight, as shown in table 2.1.

Feature	**DBMS**	**DSMS**
Model	Persistent data	Transient Data
Table	Set or bag of tuples	Infinite sequence of tuples
Updates	All	Append only
Queries	Transient	Persistent
Query answers	Exact	Often approximate
Query evaluation	Blocking and non-blocking	Non-blocking
Operators	Fixed	Adaptive
Data processing	Synchronous	Asynchronous
Concurrency overhead	High	Low

Tabelle 2.1: Functional comparison of DBMS and DSMS[PSZ15]

2.1.3 Requirements for Stream Processing Systems

What are the requirements, why do they matter to us (Elaborate on this)

Due to the nature of the fields in which SPS are used, there are important requirements that SPS should meet in order to be viable, which Stonebraker et al. point out in [ScZ05], of which the ones most important to us can be summarized as the following:

1. **Keep the Data Moving:** In order to minimize latency, data must not be stored, as these are costly operations.

2. **Handle Stream Imperfections:** Expecting only perfect data is utopian, so one must prepare the system with built-in mechanisms for data that might be missing or out-of-order.

3. **Integrate Stored and Streaming Data:** For an SPS to be able to perform comparisons between "predecessor"data and current data, operators must keep an efficiently manageable state.

4. **Guarantee Data Safety and Availability:** Recovering from a failure is detrimental for real-time data processing, so a system must be in place to guarantee the highest availability possible.

5. **Process and Respond Instantaneously:** Systems must be highly optimized in order to provide (near) real-time responses.

6. **Partition and Scale Applications Automatically:** Systems must be able to be split across multiple machines and threads. The system must also be able to automatically scale and distribute the load across the machines.

2.2 MAPE-K Loop

The MAPE-K Loop was introduced by IBM [KC03] and refers to a proposed solution for self-adaptive or autonomic systems. This model has since become the basis or reference architectural pattern for many self adaptive systems, which I will show in the third chapter. The acronym MAPE-K refers to the components that make up the model:

1. **M**onitor: The *Monitor* component gathers data about the system and its environment, aggregates and filters it. As soon as a symptom is encountered that needs to be analyzed, the information is forwarded to the *Analyze* component.

2. **A**nalyze: The *Analyze* component analyzes the previously gathered data and determines whether or not an adaptation should be performed. The decision is made based on performance or cost gain and should include the adaptation cost as well. This component's analysis is influenced by the *Knowledge* base.

3. **P**lan: If the choice to adapt the system has been made, the *Plan* component then decides how to reconfigure the system. Once the decision has been made, the information is then forwarded to the *Execute* component.

4. **E**xecute: Given the *Plan* component's decision, the *Execute* component then executes said plan and the loop returns to the initial monitoring state.

5. **K**nowledge: Represents the knowledge base, which is shared between the other components. This base is created by the *Monitor* component and contains information in the form of metrics, policies, symptoms and logs.

2.3 Self-Adaptive Systems

Cheng et al. define self-adaptive systems as

> "[...] systems that are able to adjust their behaviour in response to their perception of the environment and the system itself [...]"[CLG+09, p.1].

Self-adaptive systems are oftentimes based on the MAPE-K Loop [p.6] pattern. Adaptive Systems have a wide variety of possible application areas: adaptable user interfaces, autonomic computing, multi-agent systems [CLG+09], biologically inspired computing, robotics [SSC18], streaming applications and a lot more.

An examplary application would be a scenario, in which population and food capacities are given and evolving over time, due to births, deaths, changes in demographics and changes in weather and harvest respectively. A system would have to adapt to these changes in its environment in order to ration the food properly.

3 Approaches for Self-Adaptive Architectures in Stream Processing

Explain that this chapter showcases a few select strategies, which are then elaborated on further in the subchapters Question: Even more approaches? e.g. Master-Slave pattern or Coordinated Control pattern (Both MAPE based)? **Add and explain a few more MAPE Based architectures**

3.1 Dhalion

Quick Introduction to Dhalion, this chapter will deal with the Dhalion paper.

3.1.1 An Outline of Heron

Small outline of Heron, as Dhalion is built on top of Twitter's Heron.

3.1.2 Dhalion's Architecture

Explanation of Dhalion's Architecture **KERNPUNKT DER SECTION DHALION**

3.1.3 Discussion of Dhalion

Discuss the approach and compare it to the reference architecture (Mape?) **TODO: Maybe discuss how they evaluate, look at metrics relevant to architecture**

3.2 Hierarchical Control Architectures

Quick Introduction to hierarchical control architectures, this chapter deals with the Cardellini paper (An example for such an architecture)

3.2.1 Elastic and Distributed DSP Framework

Explanation of the EDF Architecture, their approaches

3.2.2 Possible Solutions for Controlling the Adaptation of Data Stream Processing Operators

3.2.3 Discussion of EDF

3.3 Title??

TODO: Discuss among all of them, critical thinking..

TODO: If enough material compare the architecture relevant metrics of the approaches

4 Summary And Conclusion

4.1 Summary

Summarize the paper

4.2 Conclusion

Conclude the paper

A Anhang

A.1 Beispiele

Zitat ohne Seitenangabe: [**?**]

Zitat mit Seitenangabe: [**?**, S.1]

Referenz eines Glossareintrags:

Eine normale Liste:

- Ein Punkt
- Ein anderer Punkt

Eine nummerierte Liste:

1. Erstens
2. Zweitens
3. ...

Eine Tabelle:

Tabelle A.1: Eine Tabelle

Spalte 1	Spalte 2	Spalte 3
1	2	3

Fetter Text

Kursiver Text

Eine Referenz: Siehe Tabelle A.1: Eine Tabelle auf Seite 9.

Ein Bild:

Abbildung A.1: Das Logo der SUH

Eine abgesetzte Formel:

$$\sum_{n=0}^{3} n = 6 \tag{A.1}$$

Eine Formel im Fließtext: $2^2 = 4$

Ein Listing aus einer Datei:

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Listing A.1: HelloWorld

Ein 'on-the-fly' erstelltes Listing:

```
(Sun Sep 13 23:02:20 2009): ODBC Driver <system32>\wbemdr32.dll not present
(Sun Sep 13 23:02:20 2009): Successfully  verified  WBEM OBDC adapter (incompatible version removed if it was detected).
(Sun Sep 13 23:02:20 2009): Wbemupgd.dll Registration completed.
```

Listing A.2: Beispiel eines Log-Eintrags

Glossar

Global Datasphere The Global DataSphere quantifies and analyzes the amount of data created, captured, and replicated in any given year across the world [SOURCE https://www.idc.com/getdoc.jsp?co

Literaturverzeichnis

[CLG+09] Cheng, Betty H. ; Lemos, Rogério ; Giese, Holger ; Inverardi, Paola ; Magee, Jeff ; Andersson, Jesper ; Becker, Basil ; Bencomo, Nelly ; Brun, Yuriy ; Cukic, Bojan ; Marzo Serugendo, Giovanna ; Dustdar, Schahram ; Finkelstein, Anthony ; Gacek, Cristina ; Geihs, Kurt ; Grassi, Vincenzo ; Karsai, Gabor ; Kienle, Holger M. ; Kramer, Jeff ; Litoiu, Marin ; Malek, Sam ; Mirandola, Raffaela ; Müller, Hausi A. ; Park, Sooyong ; Shaw, Mary ; Tichy, Matthias ; Tivoli, Massimo ; Weyns, Danny ; Whittle, Jon: Software Engineering for Self-Adaptive Systems. Version: 2009. http://dx.doi.org/10.1007/978-3-642-02161-9_1. Berlin, Heidelberg : Springer-Verlag, 2009. – DOI 10.1007/978–3–642–02161–9$_1. - - ISBN 978 - - 3 - - 642 - - 02160 - - 2, KapitelSoftwareEngineeringforSelf - AdaptiveSystems : AResearchRoadmap, 1 - - 26$

[KC03] Kephart, Jeffrey O. ; Chess, David M.: The Vision of Autonomic Computing. In: *Computer* 36 (2003), Januar, Nr. 1, 41–50. http://dx.doi.org/10.1109/MC.2003.1160055. – DOI 10.1109/MC.2003.1160055. – ISSN 0018–9162

[PSZ15] In: Panigati, Emanuele ; Schreiber, Fabio A. ; Zaniolo, Carlo: *Data Streams and Data Stream Management Systems and Languages*. Cham : Springer International Publishing, 2015. – ISBN 978–3–319–20062–0, 93–111

[RGR18] Reinsel, David ; Gantz, John ; Rydning, John: The Digitization of the World. Version: November 2018. https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf. International Data Corporation, November 2018. – Forschungsbericht

[ScZ05] Stonebraker, Michael ; Çetintemel, Uğur ; Zdonik, Stan: The 8 Requirements of Real-time Stream Processing. In: *SIGMOD Rec.* 34 (2005), Dezember, Nr. 4, 42–47. http://dx.doi.org/10.1145/1107499.1107504. – DOI 10.1145/1107499.1107504. – ISSN 0163–5808

[SSC18] Sabatucci, Luca ; Seidita, Valeria ; Cossentino, Massimo: The Four Types of Self-adaptive Systems: A Metamodel. In: De Pietro, Giuseppe (Hrsg.) ; Gallo, Luigi (Hrsg.) ; Howlett, Robert J. (Hrsg.) ; Jain, Lakhmi C. (Hrsg.): *Intelligent Interactive Multimedia Systems and Services 2017*. Cham : Springer International Publishing, 2018. – ISBN 978–3–319–59480–4, S. 440–450

[TSS13] Turner, David ; Schroeck, Michael ; Shockley, Rebecca: Analytics: The real-world use of big data in financial services. Version: Mai 2013. https://www.ibm.com/downloads/cas/E4BWZ1PY. IBM, Mai 2013. – Forschungsbericht