Data Lakes – On-Premises und Cloud

Robin Groh
Technische Hochschule Mittelhessen
Fachbereich MNI
Wiesenstraße 14
35390 Gießen
E-Mail: robin.groh@mni.thm.de

Prof. Dr. Harald Ritz
Technische Hochschule Mittelhessen
Fachbereich MNI
Wiesenstraße 14
35390 Gießen
E-Mail: harald.ritz@mni.thm.de

Ingo Nobbers
Technische Hochschule Mittelhessen
Fachbereich MNI
Wiesenstraße 14
35390 Gießen
E-Mail: ingo.nobbers@w.thm.de

Kategorie

Bachelorarbeit

Schlüsselwörter

Data Lake, On-Premises, Cloud, Apache Hadoop, AWS, Big Data, NoSQL

Zusammenfassung

Die Bachelorarbeit befasst sich mit der Frage , wann ein *Cloud Data Lake* und wann ein *On-Premises Data Lake* genutzt werden soll und wie diese funktionieren.
Seit der Industrie 4.0 und dem Web 2.0 werden immer mehr Daten erstellt und genutzt. Die Industrie 4.0 benötigt diese zum Beispiel für die Automatisierung von Prozessen in der Herstellung (vgl. Feld, Fettke u.a., 2014: 262). In Web 2.0 entstehen sie, wenn zum Beispiel ein Nutzer Inhalte auf einer Sozialen-Medien-Seite hochlädt (vgl. Hass, Kilian, Walsh 2020: 3).
Um den Anforderungen der Datennutzung gerecht zu werden, benötigt es eine spezielle Infrastruktur. Relationale Datenbanken sind aufgrund ihrer Architektur nicht mehr flexibel genug, um mit dieser Menge an Daten zurecht zu kommen. Ein Grund dafür ist das ACID-Prinzip, worauf jede relationale Datenbank beruht. Das ACID-Prinzip steht für *Atomicity*, *Consistency*, *Isolation* und *Durability* (vgl. Meier 2018: 30).
Wie der Name der relationalen Datenbank aussagt, sind die einzelnen Tabellen in der Datenbank mit Relationen untereinander verbunden. Um diese Relationen einhalten zu können, müssen die Daten einem strikten Schema folgen (vgl. Grunert, Heuer u.a. 2020: 21-22).
Weiterhin muss das CAP-Theorem beachtet werden. Die Abkürzung CAP steht für *Consistency*, *Availability* und *Partition Tolerance*. Das CAP-Theorem sagt aus, dass eine Datenbank nur zwei der drei Punkte abdecken kann. Eine relationale Datenbank erfüllt nur *Consistency* und *Availability*. Sobald jedoch ein Knoten bei einer relationalen Datenbank ausfällt, kann die Datenbank nicht weiter agieren. Ein Ausfallrisiko ist damit stark erhöht und kann somit zu wirtschaftlichen Schäden führen (vgl. Meier 2018: 33-34).
Um die Datenbank vor Ausfällen zu schützen und um der hohen Auslastung gerecht werden zu können, wurden die ersten nicht relationalen Datenbanken (auch NoSQL-Datenbanken genannt) entwickelt. Nicht relationale Datenbanken erfüllen bei dem CAP-Theorem *Availability* und *Partition Tolerance.* Die Aktualität der Daten ist jedoch nicht immer gesichert (vgl. Meier 2018: 33-34).
Dadurch wird bei einer nicht relationalen Datenbank das ACID-Prinzip mit dem BASE-Prinzip ersetzt. BASE steht für *Basically Available*, *Soft State* und *Eventually Consistent*. BASE besagt, dass die Daten immer zugänglich sind, ohne festem Schema abgespeichert werden und dass die einzelnen Knoten im System irgendwann auf dem neuesten Stand sind (vgl. Meier 2018: 34-35).
Die ersten nicht relationalen Datenbanken wurden von Google und Amazon entwickelt. Google entwickelte *einen Column-Family Store* namens Bigtable und Amazon einen *Key-Value Store* namens Dynamo (vgl. Hecht 2014: 4-5).
Bei einem *Key-Value Store* werden die Daten und deren Schlüssel als zwei Datenobjekte abgelegt. Eine Struktur weitere Struktur ist nicht vorhanden. (vgl. Meier, Kaufmann 2019: 202-203).
Ein *Column-Family Store* bietet im Vergleich dazu etwas mehr Struktur. Die Daten werden aber nicht pro Zeile gespeichert wie bei einer relationalen Datenbank, sondern nach Spalte. Das bringt eine höhere Abfragegeschwindigkeit mit sich (vgl. Meier, Kaufmann 2019: 205).
Aus den neuen Möglichkeiten der Datenspeicherung entstanden *Data Lakes*. Diese bauen auf nicht relationalen Datenbanken auf, um die Möglichkeit zu schaffen alle Daten auswerten und nutzen zu können (vgl. Laurent, Laurent, Madera 2020: 7). Die Daten werden zuerst aus den Quellen extrahiert und in der Datenbank gespeichert. Da kein Schema gebraucht wird, fällt die Datentransformation beim Laden weg. Da die Formatierung viel Rechenleistung in Anspruch nimmt, werden viele Ressourcen gespart (vgl. Berisha, Meziu 2021: 3-4).
Nachdem die Rohdaten in den Data Lake eingepflegt wurden, können diese analysiert und auf Anwendungsfälle zugeschnitten werden. Mithilfe dieser neun Datensätze können neue Analysen und Auswertungen erstellt werden, welche ohne die Freiheit in der Datengestaltung nicht möglich wären (vgl. Fang 2015: 820-821).

Die Anschaffung eines *Data Lakes ist* mit viel Aufwand verbunden. Deswegen ist die Planung und Auswahl der richtigen Lösung für das Unternehmen von großer Wichtigkeit.
Der hohe Aufwand entsteht zum Beispiel durch fehlendes Wissen über *Data Lakes* und der hohen Komplexität eben dieser. Besonders bei der Implementierung einer *On-Premises*-Lösung können Fehler verheerend sein, da diese zu hohen Kosten oder zum Scheitern der Implementation führen können.
Um die Architektur des *Data Lake* besser an die Anforderungen anzupassen, beziehungsweise diesen besser nach den Anforderungen auszusuchen, wurde ein Kriterienkatalog erstellt. Dieser Kriterienkatalog kann dafür genutzt werden einzelne *Data Lake*-Lösungen miteinander zu vergleichen.
Jede Anforderung bekommt dabei eine Gewichtung und eine Begründung für diese. Die Gesamtheit der Gewichtung muss dabei auf 100 kommen. Die Gewichtung des Kriterienkataloges in dieser Arbeit wurde anhand eines Fallbeispiels erstellt. Die einzelnen Lösungen bekommen Punkte gemessen daran, wie gut sie die Anforderungen erfüllen. Diese Punkte addieren sich dabei in Zehnerschritten auf einer Skala von Null bis Hundert.
In dieser Arbeit wurden dafür eine *On-Premises*-Lösung und eine *Cloud*-Lösung ausgewertet und miteinander verglichen. Die *On-Premises*-Lösung basiert auf Apache Hadoop und die *Cloud*-Lösung auf AWS Lake Formation.
Die *Cloud*-Lösung erzielte mit 94,6 von 100 Punkten eine höhere Punktzahl als die *On-Premises*-Lösung. Diese hatte nur 87,2 von 100 Punkten. Das Ergebnis besagt jedoch nicht, dass *Cloud*-Lösungen immer besser sind als *On-Premises*-Lösungen. Es hat sich vielmehr herauskristallisiert, dass eine *Cloud*-Lösung gut als Einstieg geeignet ist. Grund dafür ist das Wegfallen der Hardware und die leichtere Implementierung des *Data Lakes.* Weiterhin eignet sich eine *Cloud*-Lösung auch gut für eine Auslagerung des *Data Lakes*.
Eine *On-Premises*-Lösung lohnt sich eher dann, wenn ein genauer Plan vorgegeben ist und die einzelnen Anwendungsfälle klar definiert sind.
Die Implementierung wird durch die genaue Planung komplizierter, kann dadurch allerdings besser auf die einzelnen Anwendungsfälle angepasst werden.
Die Kosten entstehen bei den Lösungen unterschiedlich. Bei einer *Cloud*-Lösung fallen die Kosten meist monatlich an und ergeben sich aus der Nutzung. Dadurch bleiben die initialen Kosten klein, während sie langfristig gleichbleiben oder steigen können.
Die *On-Premises*-Lösung hat zu Beginn hohe Kosten, bleibt aber langfristig günstig, da sich die Kosten vor allem aus der Hardware und der Implementierung ergeben.

Literatur

Feld, Thomas, Fettke, Peter u.a. (2014): Industrie 4.0 in: Wirtschaftsinformatik. Ausgabe 4 2014. S. 262. https://link.springer.com/content/pdf/10.1007/s11576-014-0424-4.pdf (06.09.2022)

Hass, Berthold H.; Kilian, Thomas; Walsh, Gianfranco (2020): Grundlagen des Web 2.0. Heidelberg: Springer. https://link.springer.com/content/pdf/10.1007/978-3-642-13787-7.pdf (11.10.2022)

Meier, Andreas (2018): Werkzeuge der digitalen Wirtschaft: Big Data, NoSQL & Co. Eine Einführung in relationale und nicht-relationale Datenbanken. Auflage 1. Wiesbaden: Springer. https://link.springer.com/content/pdf/10.1007/978-3-658-20337-5.pdf (11.07.2022)

Hecht, Robin Stefan (2014): Konzeptuelle und methodische Aufarbeitung von NOSQL-Datenbanksystemen. https://epub.uni-bayreuth.de/1847/1/Dissertation%20Robin%20Hecht%20-Konzeptuelle%20und%20Methodische%20Aufarbeitung%20von%20NoSQL-Datenbanksystemen.pdf (15.07.2022)

Grunert, Heuer, Meyer, Saake, Sattler (2020): Datenbanken: Kompaktkurs. Frechen: MITP. https://web.p.ebscohost.com/ehost/ebookviewer/ebook/bmxlYmtfXzI2MzIwNTdfX0FO0?sid=33c5359e-565c-4d7b-a3bd-52d5e9e58153@redis&vid=0&format=EB&lpid=lp_1&rid=0 (04.07.2022)

Meier, Andreas / Kaufmann, Michael (2019) SQL & NoSQL Databases. Models, Languages, Consistency Options and Architectures for Big Data Management. Auflage 1. Wiesbaden: Springer. https://link.springer.com/content/pdf/10.1007/978-3-658-24549-8.pdf (15.07.2022)

Berisha, Blend / Meziu, Endrit (2021): Big Data Analytics in Cloud Computing: An overview. https://www.researchgate.net/profile/Blend-Berisha/publication/348937287_Big_Data_Analytics_in_Cloud_Computing_An_overview/links/601825fb92851c2d4d0c3d94/Big-Data-Analytics-in-Cloud-Computing-An-overview.pdf (18.07.2022)

Laurent, Anna / Laurent, Dominique / Madera, Cédrine (2020): Data Lakes. Databases and Big Data Set. Auflage 2. London: ISTA. https://onlinelibrary.wiley.com/doi/epub/10.1002/9781119720430 (07.08.2022)

Fang, Huang (2015): Managing Data Lakes in Big Data Era. What's a data lake and why has it became popular in data management ecosystem. in: The 5th Annual IEEE International Conference on Cyber Technology in Automatic, Control and Intelligent System. S. 820-824.