Datenbank-Spektrum
https://doi.org/10.1007/s13222-024-00490-5

KURZ ERKLÄRT

Data Architectures in Cloud Environments

Maximilian Plazotta[1] · Meike Klettke[1]

Accepted: 2 October 2024

Abstract
Data architectures are an integral part for developing robust data pipelines. Cloud computing offers various features to build modern data architectures to power data engineering pipelines on a scalable, cost-effective, and secure level.
In this article we describe how cloud computing works in the field of data management. We start with brief explanations on cloud computing, general data architectures, and how both worlds fit together by introducing design principles, tools, and services for cloud data architectures.

Keywords Data architecture · Cloud computing · Data warehouse · Data lake

1 Introduction

With the upcoming of commercial cloud computing (AWS in 2006, Google Cloud in 2008, Azure in 2008) over the last 20 years, new forms of data architectures emerged with it. As a consequence several novel requirements arose from new data-intensive applications. Especially varying data formats, big data (data velocity, volume, variety, veracity), and a need for collaboration across platforms drove new types of architectures e.g. data lake, data fabric, data mesh, or data lakehouse apart from the established traditional databases or data warehouse architectures. Fig. 1 displays the popularity of data architectures over time using Google trend search (0 = lowest popularity, 100 = highest popularity). From being very popular going even back to 1990s, classical data warehouses declined over the last two decades—not saying they have become unpopular, but different approaches arose over the last years. After 2015, one can observe a huge gain in popularity of these new types of architectures. This article gives an overview on how data architectures can be set-up within cloud environments and which services therefore can be used. In Sect. 2, cloud computing is explained in detail. Sect. 3 is devoted to data architectures and how they are run in the cloud. Sect. 4 gives an outlook of future research possibilities.

✉ Maximilian Plazotta
maximilian.plazotta@gmail.com

Meike Klettke
meike.klettke@ur.de

[1] University of Regensburg, Regensburg, Germany

2 Cloud computing

"There is no cloud—it is just someone else's computer"—is a phrase often used by skeptics to downgrade cloud computing and its capabilities. But it provides much more than a typical computer: The three key offerings are **compute**, **network**, and **storage** with various services around it e.g., servers, databases, software, continuous integration/ continuous deployment (CI/CD), security, and intelligence/ analytics. It easily scales through the elasticity/on-demand nature of cloud infrastructures with no upfront investment in the underlying hardware. Forms of cloud computing are **public cloud**, **private cloud**, **hybrid cloud**, and **multi-cloud**. Public clouds are third-party offerings by hyperscaler companies that are operated by them—the technical backbone of cloud computing (firms) are data centers located around the globe [2]. Private clouds are owned and managed by individual companies; hybrid clouds are a mix of public and private cloud offerings. Multi-cloud set-ups, an architecture with multiple cloud providers, gained popularity especially over the last few years due to rising prices of cloud workloads which forces companies to compare pricing across cloud providers. Furthermore, it also helps to mitigate the risk of service outages or prevent vendor lock-in. The usage of cloud resources are grouped in four categories: **infrastructure as a service (IaaS)**, **platform as a services (PaaS)**, **software as a services (SaaS)**, and

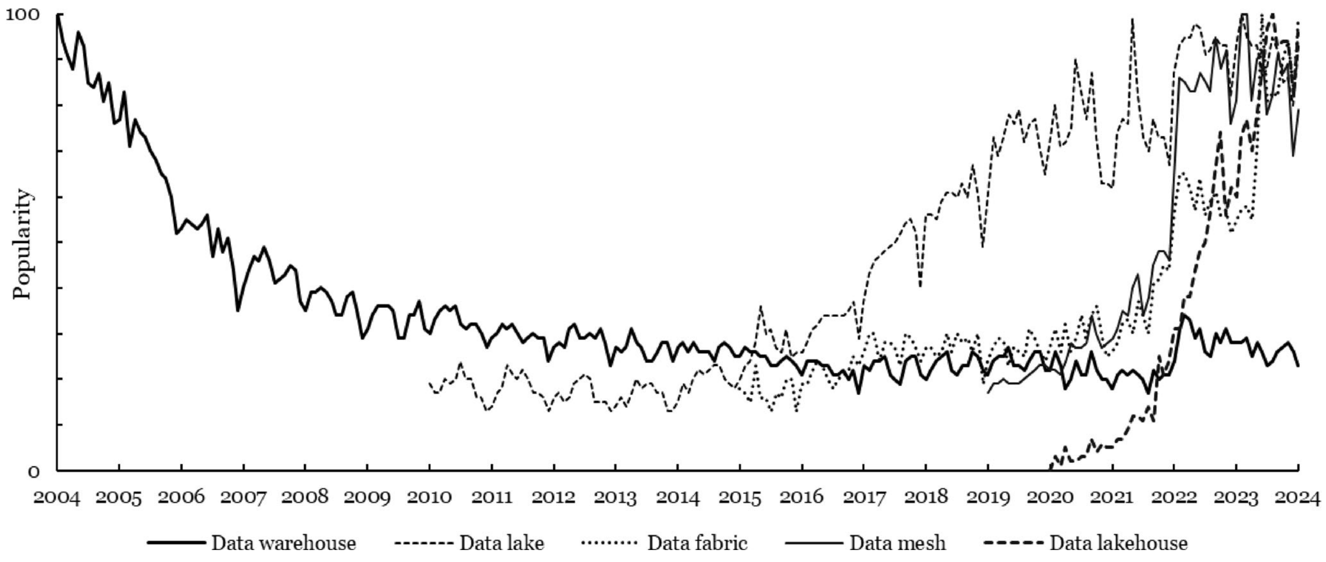

Fig. 1 Popularity of data related terms

serverless computing. IaaS are basically raw IT infrastructure services like virtual machines (VMs). Here the cloud provider operates the underlying infrastructure, but the user is responsible for managing the software (installation, maintenance, updates etc.) which runs on the VM. PaaS services build a layer on top of IaaS to enable the building, testing, and deployment of underlying software or applications. Closely related to PaaS is serverless computing, or also called function as a service (FaaS) which allows to run code without the need for managing the underlying servers. SaaS services are fully externally managed applications without any need for maintenance of infrastructure, updates/upgrades, and patching.

3 Data architectures

Following the guiding principles of the open group architecture framework (TOGAF[1]) a data architecture refers to the design, structure, and components for managing data assets. It describes how data is collected, processed, stored, distributed, and consumed across systems. Over the last 50 years different forms of data architectures became popular—due the rapid growth of data volume, more computing power, and introduction of new technologies, various new types arose especially since 2010 (see Fig. 2).

The first form are **relational databases** which can be dated back to the 1970s. The key criteria of these architectures are the usage of the relational data model [4], the usage of structured query language (SQL) [3], an optimization for transactional queries (OLTP) and the support of ACID transactions [7]. For transactional workloads, these relational database management systems (RDBMS) are still in use today.

Introduced by Inmon (1990) [8] and Kimball (1996) [10] in the 1990s, the **data warehouse** is still a widely used architecture. According to Inmon [8] "a data warehouse is a subject-oriented, integrated, non-volatile, and time variant collection of data in support of managements decisions" and offers more than traditional databases e.g., ETL capabilities, extensive storage of multi-dimensional data, and business intelligence integration. As traditional database architectures support online transaction processing (OLTP) workloads, data warehouses are optimized for online analytical processing (OLAP). Inmon's top-down approach focuses on the enterprise data warehouse (*Corporate Information Factory—CIF*) while Kimball introduces the bottom-up dimensional model where data is organized in different schemata, e.g. star, snowflake schema or a multidimensional storage. Cloud data warehouses, self-deployed or through

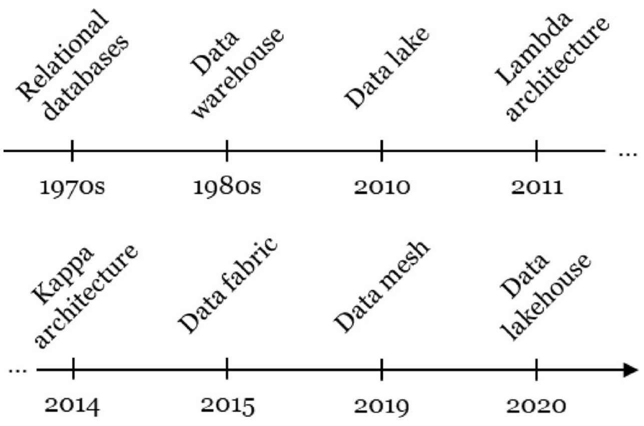

Fig. 2 Timeline data architectures

[1] https://pubs.opengroup.org/architecture/togaf91-doc/arch/chap10.html.

SaaS solutions, are a commonly used form in enterprises today.

Unlike data warehouses, that enforce a specific structure to the data before storage, **data lakes** allow data to be stored as they are—in structured, semi-structured, or unstructured form. The term data lake was initially coined by James Dixon (2010) [6]. Data lake architectures are heavily used nowadays due to the low effort needed for data ingestion and to their high scalability (terabytes or petabytes of data) combined with relative low cost of storage fitting perfectly in clouds.

With the rising popularity of big data during the early-2010s, new architectures for real-time data processing were established. The **lambda architecture**, introduced by Nathan Marz first in his blog post *"How to beat the CAP theorem"* (2011) [13], and then later more elaborated in Marz and Warren (2015) [14], describes the parallelization of both batch ("cold path") and real-time ("hot path") data processing. Consisting of three layers: the batch layer, the speed layer, and the serving layer, the lambda architecture provides a scalable and fault-tolerant approach to address different types of big data use cases e.g., social media analytics [14] or traffic management [16]. The **kappa architecture** proposed by Jay Kreps (2014) [12] is a derivation of the lambda architecture which excludes the batch layer to solely focus on the streaming layer. Both forms are directly deployable within cloud environments and solve the problem of processing and analyzing large-scale, real-time data alongside historical batch data.

The term **data fabric** was coined by NetApp [9] in 2015 as they observed a rising demand in shifting data between cloud providers. This procedure is (still today) costly and complicated; often referred to as (cloud) vendor lock-in. Data fabric focuses on connecting distributed data from different (siloed) systems, environments, or applications. The glue that holds the architecture together is the metadata—it captures information on where the data is located and what inter-dependencies exist. Key criteria of data fabric architectures are data cataloging and lineage, metadata and master data management, and data integration. Data fabric is not a single technological solution like e.g. a data warehouse, it is a framework for building robust, distributed data platforms across multiple environments [15].

One of the most recently proposed architectures, the **data mesh**, relies on decentrally managed data ecosystems. The four key principles of data mesh proposed by Dehghani (2019) [5] are:

1. Domain ownership
2. Data as a product
3. Self-serve data platform
4. Federated computational governance

Compared to the previously discussed architectures where one central IT-department manages the platform, the data mesh approach designates the individual (business) teams as responsibles (domain ownership). Within these domains data is considered to be a (data) product which can be used, shared, and distributed to create new data products—all under the self-service principle with a governance framework around it.

With the introduction of the **data lakehouse** in 2020 [17] by the originators of Apache Spark and founders of Databricks, a new approach was established which combines data warehouse and data lake features. Data lakes, with their advantages in handling different data formats, low storage cost, flexibility with scalable compute, is paired with the data warehouse (primarily structured data, ETL, business intelligence) to achieve a flexible, scalable, and cost-effective data architecture. In this context Databricks also introduced the medallion architecture approach with bronze (raw data), silver (filtered, cleaned, enriched), and gold (ready for consumption) layers to clean and improve data.

3.1 Components of data architectures

As defined at the beginning of Sect. 3, a data architecture must possess integral components to manage data assets. Over time, different data engineering methods emerged and therefore became more sophisticated and technically complex. In [11], different generations of data engineering pipelines are introduced, from simple preprocessing steps to optimized pipelines. Modern data architectures must support data pipelining efforts for various data formats, velocities, and volumes. Fig. 3 gives a high-level overview of central components of data architectures where data is processed end-to-end. It often starts with data being extracted from the source system(s) into the data platform where the actual magic happens: data ingestion, data processing, data storage, and data consumption.

The proposed logic can be traced back to ETL/ELT (load stored) processes from the 1980s/1990s. Nevertheless, the details and degree of complexity of such modern platforms need to include further and more granular steps.

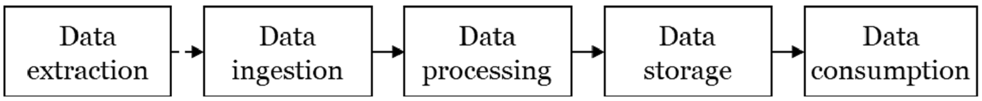

Fig. 3 Typical data pipeline

Table 1 Design principles for cloud data architectures

Design principle	Explanation
Modularity	Interchangeability of components (e.g because of changes in datasets, processing algorithms, or workflows) services, and tools related to data
Scalability	Adjustment to increasing data volumes and compute needs (e.g. auto-scaling)
Flexibility, Agility	Adaptation to changing data sources or formats
Cost-effectiveness	Resource-efficient platform with on-demand usage and flexible pricing
Automation	Automate where possible e.g. infrastructure as code (IaC), pipeline scheduling, monitoring/alerting
Reliability	Implementation of fail-over mechanisms and disaster recovery (multiple availability zones with replication); support of ACID transactions
Security	Data access rules, encryption at-rest and in-transit, authorization (e.g., multi-factor authentication)
Performance	Optimization mechanisms to enhance performance and efficiency (indexing, compression, caching etc.)

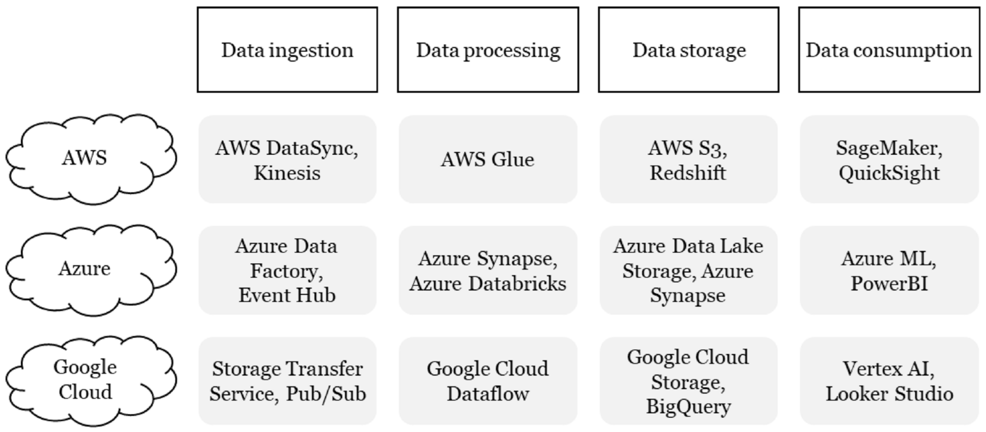

Fig. 4 Cloud data services

3.2 Design principles for data architectures deployed in the cloud

All presented data architectures from Sect. 3 can also be deployed in cloud environments. Popular transactional databases like MySQL or PostgreSQL can easily be hosted in clouds. Data warehousing and data lake solutions can be used directly from cloud providers or from third party companies. The other mentioned more sophisticated architectures are also deployable across cloud environments. To build robust and scalable cloud data architectures, some underlying design principles ought to be fulfilled (see Table 1). Data is the key variable for defining and selecting the design principles. Therefore, we focus on data security (e.g. encryption, access rules, etc.) and not on general cloud security (e.g. avoidance of open endpoints, misconfiguration, etc.) for the security principle.

3.3 Cloud data services

Cloud providers[2] offer various services to implement end-to-end data pipelines (see Fig. 4). Such pipelines and their components are needed in many data architectures shown in Fig. 2, e.g. in data warehouses, data mesh, and data lakehouses. Following the process from Sect. 3.1 and Fig. 3, the first step **data extraction** is often executed outside the cloud environment as many applications are still hosted elsewhere. Nevertheless, there exist various APIs and connectors to extract data from the source systems into the cloud platform. Once the data is accessible, a **data ingestion** service like AWS DataSync (batch data), AWS Kinesis (streaming data), Azure Data Factory (batch), Azure Event Hubs (streaming), Google Cloud Storage Transfer Service (batch), Google Cloud Pub/Sub (streaming) is used. These services support the ingestion of a variety of data sources from different environments and varying data formats for batch or streaming data.

For **data processing** AWS Glue is the central service to prepare, transform, and load data in AWS. In Microsoft Azure one can use Azure Synapse (Analytics) or (Azure) Databricks; in Google Cloud it is called Dataflow. For **storing data**, there are basically two forms: the data warehouse and the data lake. The classic AWS data warehouse is called AWS Redshift, for Azure it is Synapse, and for Google Cloud BigQuery. The data lake storage in AWS is S3; Azure Data Lake Storage (ADLS), and Google Cloud Storage, respectively. Once the data is ready for **consumption**, it can be visualized through business intelligence ser-

[2] Besides cloud providers, there also exist 3rd party companies who offer fully managed data platforms as SaaS like Snowflake, Databricks, Starburst, and Dremio.

vices (AWS QuickSight, Microsoft PowerBI, Google Cloud looker studio) or advanced analytics like machine learning (AWS SageMaker, Azure Machine Learning, Google Cloud Vertex AI) can be applied to the data. It is important to mention that some services possess overlapping functionality e.g., Azure Synapse is also able to ingest, process, and store data. Services are not randomly interchangeable across cloud providers, but data is shareable to some extent with e.g., delta lake [1] across cloud object stores (S3, ADLS, GCS).

4 Open Research Questions

This article gives a broad overview of significant data architectures and their deployment in cloud environments with guiding principles on design and functionality. As illustrated in Fig. 2, the development of data architectures is inevitable due to the introduction of new technologies, approaches, or technical advancements. So, what is the next big architectural approach after the recent data lakehouse? Obviously, efficient data sharing across multiple (cloud) systems is not solved satisfactorily but there are some promising concepts like data spaces or delta share. Egress cost, cost that arise from moving data from a data center to the internet, are still a huge roadblock when it comes to (outbound) data sharing. This year (11.01.2024) the European Commission introduced the EU Data Act which upon other terms enforces cloud providers to remove any egress cost by January 2027 (Chap. VI, Art. 29.1). In our opinion, an ecosystem for developing data pipelines in the cloud based on specific application requirements and migration to another cloud provider are also research tasks for the coming years. Furthermore, the impact of artificial intelligence (e.g. in forms of large language models) on data engineering pipelines offers huge potential to improve data quality or optimize performance.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Armbrust M, Das T, Sun L, Xin R, Zhu S, Ghodsi A, Yavuz B, Murthy M, Torres J, Sun L, Boncz PA, Mokhtar M, Hovell HV, Ionescu A, Luszczak A, Switakowski M, Ueshin T, Li X, Paranjpye S, Szafranski M, Senster P, Zaharia M (2020) Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores. Proc Vldb Endow 13(12):3411–3424
2. Barroso LA, Hölzle U, Ranganathan P (2018) The Datacenter as a Computer: Designing Warehouse-Scale Machines. Morgan, Claypool Publishers
3. Chamberlin DD, Boyce RF (1974) SEQUEL: A Structured English Query Language. In: Proceedings of ACM-SIGMOD Workshop on Data Description, Access and Control, pp 249–264
4. Codd EF (1970) A Relational Model of Data for Large Shared Data Banks. Commun ACM 13(6):377–387
5. Dehghani Z, Fowler M (2022) Data Mesh: Delivering Data-driven Value at Scale. O'Reilly Media
6. Dixon J (2010) Pentaho, Hadoop, and Data Lakes. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/
7. Härder T, Reuter A (1983) Principles of Transaction-Oriented Database Recovery. ACM Comput Surv 15(4):287–317
8. Inmon WH (1990) Building the Data Warehouse, 1st edn. John Wiley & Sons, Inc, USA
9. Kidd J (2015) Realize the Full Potential of Cloud with the Data Fabric. https://community.netapp.com/t5/Tech-ONTAP-Articles/Realize-the-Full-Potential-of-Cloud-with-the-Data-Fabric/ta-p/101344
10. Kimball R (1996) The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley
11. Klettke M, Störl U (2022) Four Generations in Data Engineering for Data Science. Datenbank Spektrum 22(1):59–66
12. Kreps J (2014) Questioning the Lambda Architecture. https://www.oreilly.com/radar/questioning-the-lambda-architecture/
13. Marz N (2011) How to beat the CAP theorem. http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html
14. Marz N, Warren J (2015) Big Data: Principles and best practices of scalable realtime data systems. Manning
15. Strengholt P (2023) Data Management at Scale. O'Reilly
16. Yousfi S, Rhanoui M, Chiadmi D (2019) Towards a Generic Multimodal Architecture for Batch and Streaming Big Data Integration. J Comput Sci 15(1):207–220
17. Zaharia M, Ghodsi A, Xin R, Armbrust M (2021) Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. CIDR