Data Lake Architecture

A Comprehensive Design Document

Prepared by: Hui Ren

January 31, 2021

TRACKER

Revision, Sign off Sheet and Key Contacts

Change Record

Date	Author	Version	Change Reference
01/31/2021	Hui Ren	1.0	Initial draft

Reviewers / Approval

Name	Version Approved	Position	Date

Key Contacts

Name	Role	Team	Email

DESIGN

Purpose

Summary

Medical Data Processing Company's current data architecture can't keep up with the growth. As the volume of data continues to grow, the existing single node SQL Server is not able to scale. The SQL Server has become a single point of failure, hosting critical customer data. The CTO is looking for a data lake solution. This is the comprehensive design document with the proposed data lake architecture.

Document Outline

- Data Lake Requirements
- Data Lake Architecture design principles
- Assumptions
- Data Lake Architecture Proposal
- Design Considerations and Rationale
- Conclusion

Target Audience

Data lake stakeholders including:

- Company leadership
- Data engineers
- Data scientists
- External partners

In Scope and Out of Scope Items

- In scope: the requirements, the assumptions, the design of the data architecture
- Out of scope: implementation of the data architecture, data governance, machine learning

Data Lake Requirements

Summary of requirements for Data Lake

- Design a system with high availability, reliability, and resiliency
- Scale easily to keep up with the growth
- Break down data silos and maintain one source of truth
- Integrate flexibly with ML frameworks, reports and dashboards

Existing Technical Environment

- 1 Master SQL DB Server
- 1 Stage SQL DB Server
    - 64 core vCPU
    - 512 GB RAM
    - 12 TB disk space (70% full, ~8.4 TB)
    - 70+ ETL jobs running to manage over 100 tables
- 3 other smaller servers for Data Ingestion (FTP Server, data and API extract agents)
- Series of web and application servers (32 GB RAM Each, 16 core vCPU)

Current Data Volume

- Data coming from over 8K facilities
- 99% zip files size ranges from 20 KB to 1.5 MB
- Edge cases - some large zip files are as large as 40 MB
- Each zip files when unzipped will provide either CSV, TXT, XML records
- In case of XML zip files, each zip file can contain anywhere from 20-300 individual XML files, each XML file with one record
- Average zip files per day: 77,000
- Average data files per day: 15,000,000
- Average zip files per hour: 3500
- Average data files per hour: 700,000
- Data Volume Growth rate: 15-20% YoY

Business Requirements

- Improve uptime of overall system
- Reduce latency of SQL queries and reports

- System should be reliable and fault tolerant
- Architecture should scale as data volume and velocity increases
- Improve business agility and speed of innovation through automation and ability to experiment with new frameworks
- Embrace open source tools, avoid proprietary solutions which can lead to vendor lock-in
- Metadata driven design - a set of common scripts should be used to process different types of incoming data sets rather than building custom scripts to process each type of data source.
- Centrally store all of the enterprise data and enable easy access

Technical Requirements

- Ability to process incoming files on the fly (instead of nightly batch loads today)
- Separate the metadata, data and compute/processing layers
- Ability to keep unlimited historical data
- Ability to scale up processing speed with increase in data volume
- System should sustain small number of individual node failures without any downtime
- Ability to perform change data capture (CDC), UPSERT support on a certain number of tables
- Ability to drive multiple use cases from same dataset, without the need to move the data or extract the data
  - Ability to integrate with different ML frameworks such as TensorFlow
  - Ability to create dashboards using tools such as PowerBI, Tableau, or Microstrategy
  - Generate daily, weekly, nightly reports using scripts or SQL
- Ad-hoc data analytics, interactive querying capability using SQL

Data Lake Architecture Design Principles

Use event sourcing to ensure data traceability and consistency

In a data lake architecture where compute and storage are separated, event sourcing should be used and an immutable log of all incoming events should be maintained on object storage. Event sourcing enables you to retrace your steps to learn about the exact transformation applied on the raw data, down to the event level. If there was an issue in your ETL code, you can easily fix it and run the new code on the immutable original data.

Layer data lake according to user's skills

In a data lake, we have the possibility to store multiple copies of the data for different use cases and consumers. By automating the ETL pipelines that ingest the raw data and perform the relevant transformations per use case we can prevent the data engineering bottleneck that might form if we rely on coding-based ETL frameworks such as Apache Spark.

Keep the architecture open

To create an open architecture, you should:

- Store the data in open formats like Avro and Parquet which are standard, well-known and accessible by different tools.
- Retain historical data in object storage like Amazon S3.
- Use a central meta-data repository such as AWS Glue. This will allow you to centralize and manage all your meta-data in a single location, reducing operational costs in infrastructure, IT resources and engineering hours.

Plan for performance

To ensure high performance when querying data, you need to apply storage best practices to make data widely available:

- You want every file stored to contain the metadata needed in order to understand the data structure.
- Use columnar file formats such as Apache Parquet and ORC.
- Keep your data in optimal file sizes. A Hot/Cold architecture is recommended: hot – small files for good freshness; cold – merging small files into bigger files for better performance.
- Build an efficient partitioning strategy to ensure queries run optimally by only retrieving the relevant data needed in order to answer a specific analytical question.

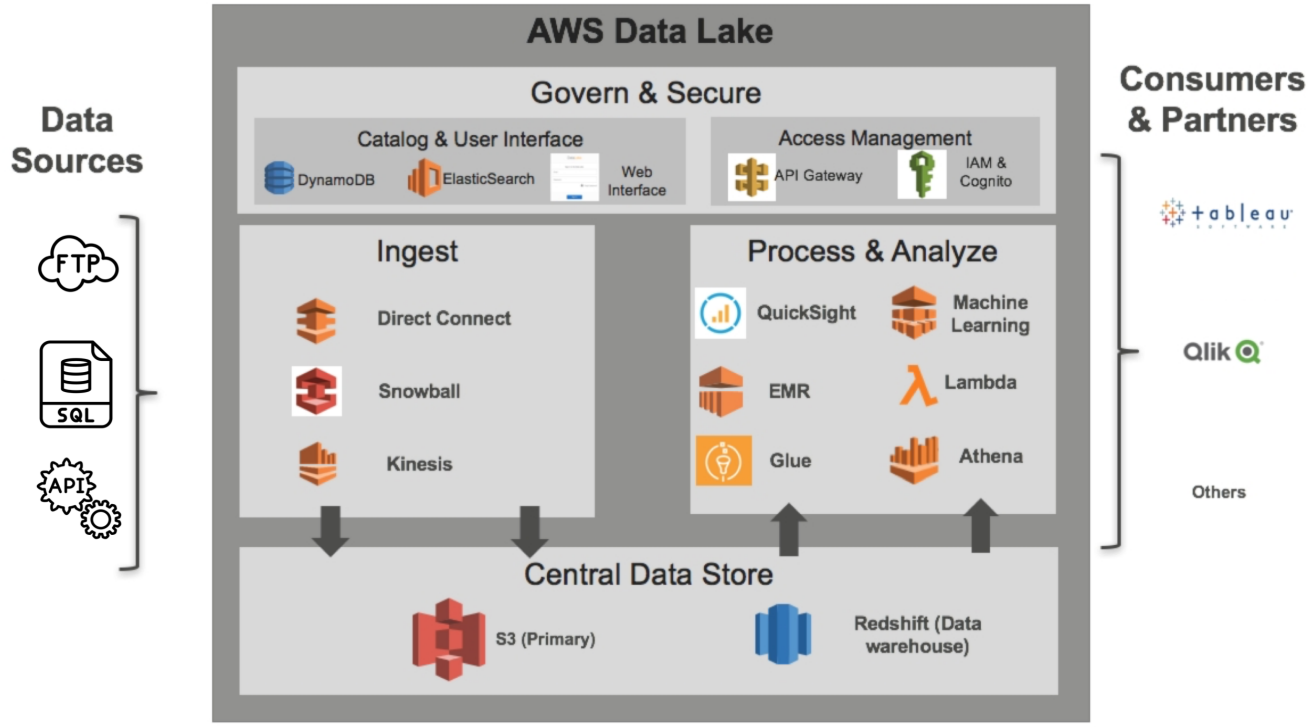
Assumptions

- Timeline for transitioning to the data lake: ASAP.
- Cloud is preferred over on-premise infrastructure. An on-premise data lake imposes challenges. Companies must build their own data pipelines, pay the ongoing management and operational costs in addition to the initial investment on servers and storage equipment, and manually add and configure their servers to scale a data lake to cater to more users or increasing data volume.
- 100% of the data is migrated to the cloud.

Data Lake Architecture

I'll leverage the AWS Data Lake solution. The overall services can be grouped into the following four categories:

- Managed ingestion to onboard data from various sources and any format.
- Centralized storage that can scale as per the business needs.
- Processing and analyzing at big data scale in various programming languages.
- Governing and securing your data packages.



Design Considerations and Rationale

Govern & Secure

Governance is key in building a managed data lake. Below are the the key services used for governance:

- User and access management: AWS Data Lake solution provides a web interface to manage users for the data lake. As a data lake administrator, you can decide which user gets access to the data lake and at what level (member or administrator). You can also grant API access to specific users.
- Data Catalog: As data is used in different platforms, ETL (Extract, Transform, Load) is an important function to ensure that it is moved and understood properly. AWS Glue is an ETL engine that can be used to understand

data sources, prepare data, and load it reliably to data stores. AWS Glue discovers the data and stores the associated metadata (e.g. table definition and schema) in the AWS Glue Data Catalog. Once cataloged, the data is immediately searchable, can be queried, and is available for ETL.

- Data Catalog User interface: The data catalog is searchable using the web application. The catalog can be populated by the web interface or via the API with information about the various packages for the data lake, and this information is stored in DynamoDB. Once the datasets are registered, they are automatically indexed to Elasticsearch and are searchable by the web interface.
- Security and compliance: AWS has a deep suite of security offerings like Amazon Macie, a security service that uses machine learning to automatically discover, classify, and protect sensitive data. The data center and network architecture were built to meet the requirements of the most security-sensitive organizations. AWS also actively manages dozens of compliance programs in its infrastructure, helping organizations to meet compliance standards such as PCI DSS, HIPAA, and FedRAMP. Below are the best practices to secure the data:
    1) Classify the data:
        1) Identify the data within your workload
        2) Automate identification and classification
        3) Define data lifecycle management
    2) Protect data in rest:
        1) Implement secure key management
        2) Enforce encryption at rest
        3) Automate data at rest protection
        4) Enforce access control
    3) Protect data in transit:
        1) Implement secure key and certificate management
        2) Enforce encryption in transit
        3) Automate detection of unintended data access
        4) Authenticate network communications

<u>Ingest</u>

Amazon has several tools that can ingest data to S3 and Redshift. They can be easily scaled to meet the demand.

- Direct Connect establishes private connectivity between AWS and the enterprise data center and provides an easy way to move data files from the applications to S3.

- Snowball imports hundreds of terabytes of data quickly into AWS using Amazon-provided secure appliances for secure transport.
- Kinesis and Kinesis Firehose enable the building of custom applications that process or analyze streaming data.

I considered open-source tools including Apache Sqoop, Flume, Kafka and Nifi but decided to use managed services Amazon provided.

- Apache Sqoop is a tool designed for efficiently transferring data between structured, semi-structured, and unstructured data sources. Sqoop works on top of Hadoop HDFS and once a Sqoop job is submitted, it gets converted to a MapReduce job.
- Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. Flume, just like Scoop Leverages Hadoop. All the Flume jobs are internally converted into MapReduce. Apache Flume architecture consists of three major components, sources, channels, and sinks. A Flume source consumes events delivered to it by an external source like a web server. The external source sends events to the flume in a format that is recognized by the target Flume source. When a flume source receives an event, it stores it into one or more channels, which keep the event until it is consumed by a flume sink.
- Apache Kafka, is an open source distributed event streaming platform. Kafka is used by thousands of companies, for high performance data pipelines and streaming analytics. Kafka has a very diverse set of use cases. Kafka is horizontally scalable, fault-tolerant, fast framework, for building real time data pipelines.
- Apache NiFi is an integrated data logistics platform for automating the movement of data between disparate systems. It provides real-time control that makes it easy to manage the movement of data between any source and any destination.

Store

This data lake architecture is based on Amazon S3 as the primary persistent data store. We can separate compute and storage, and resize on demand with no data loss. It is also highly durable and low-cost with several options to connect. Data will be stored in the native formats upon ingestion. After ETL, data can be stored in open formats that are standard, well-known and accessible by different tools. Data can be shared with multiple data lakes through the S3 buckets. Data can be replicated in different regions for back-up and recovery. Because SQL-based access is needed, Redshift is added to the storage layer. Redshift is a fast and fully managed petabyte-scale data warehouse that costs less than $1,000 per terabyte per year.

I considered the cloud storage from Microsoft and Google. But Amazon S3 is a better choice in terms of the cost and performance.

Process

For data processing and analysis, the services from AWS will be used. They have built-in scalability.

- Lambda runs code without provisioning. It can be triggered upon arrival of data in S3 and for any streaming sources from Kinesis.
- Glue is a service that enables ETL and makes it easy to transform and move data from S3 to the consumers. It is integrated with S3, Redshift, and other JDBC compliant data sources and auto-suggests schemas and transformations, which improve developer productivity. You can also view and edit the code it generates in popular languages such as Python and Spark, with the ability to share the code with your peers. Glue schedules the ETL jobs and auto-provisions and scales the infrastructure based on the job requirements.
- EMR provides a distributed compute framework that makes it an easy, fast, and cost-effective way to process data on S3 at scale and on demand. AWS now provides options for spot instances that are offered at lower cost. They are best used for application tests and use cases that don't have hard SLA's.
- Athena is a query service that makes it easy to analyze data directly from files in S3 using standard SQL statements. Athena is server-less, which makes it really stand out since there is no additional infrastructure to be provisioned. Athena can be used as the ad-hoc query tool.
- QuickSight is fast, cloud-powered business intelligence (BI) service.
- Machine learning provides visualization tools and wizards for creating machine learning models and executing them on big data.

I considered Apache MapReduce, Pig, Hive and Spark as the processing tool, but decided to use managed services Amazon provided.

- MapReduce orchestras the processing by marshaling the distributor servers, running various tasks in parallel. MapReduce is a programming model and an associated implementation for processing and generating big datasets with a parallel distributed algorithms on a cluster. MapReduce also manages all the communication and the data transfers between the various parts of the system, providing redundancy and fault tolerance. A MapReduce program executes in three stages, namely maps stage, shuffle and sort stage, and the reduce stage. In map stage, mapper's job is to process the input data. Generally, the input data is in a form of a file or a directory and is stored on Hadoop file system. The input file is passed to the mapper function line by line. The mapper processes the data and create several small chunks of the data. The output of the map stage is

processed using shuffle and sort stage. Where, the data is shuffled across the network of nodes and sorted based on designated key. The output of shuffle and sort phase is used by the reducer phase. In the reducer stage, data is aggregated and then the final output is stored on HDFS. The MapReduce framework manages all the details of data passing, such as issuing task, verifying task completion, and copying the data around the cluster between the nodes. This really simplifies the data processing across a distributed system.

- Apache Pig is simple to use, yet very powerful framework. Pig is a platform for analyzing large data sets that consists of a high level language for expressing data analysis programs. Apache Pig works on top of HDFS, coupled with infrastructure for evaluating this program such as MapReduce. Pig offers ease of programming. Complex task comprised of multiple interrelated data transformations are very easy to write, understand, and maintain. For example, a simple word count program that can take 60 to 70 lines of Java MapReduce code can be expressed with less than ten lines of Apache Pig code. Pig framework also optimize the execution automatically, allowing you to focus on the semantics rather than the efficiency. Pig framework is very flexible and offers great deal of extensibility. If you are doing a transformation, and if the built-in Pig functions are not enough, you can create your own functions and do spatial purpose processing. These functions are called user defined functions.
- Apache Hive is a data warehouse software which allows you to use SQL language to read, write, and manage large datasets residing on Hadoop distributed storage. Hive was originally developed by Facebook, but later it was open sourced. Hive is a very popular tool and used by some of the largest tech companies around the globe. Hive gives a SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. Hive comes with command line tool and JDBC driver to connect the users to Hive programmatically. Hive provides necessary SQL abstraction to integrate SQL like queries, also known as Hive query language into the underlying Java without the need of actually implementing the queries in a low level Java MapReduce API. Basically, to interact data stored on Hadoop, you do not need to write really long and complicated MapReduce code in Java. You can interact with the same data by writing SQL queries and Apache Hive framework will convert the SQL queries into MapReduce code behind the scenes.
- Spark provides fast unified data processing capabilities on a large amount of data stored on Hadoop. Spark is a unified analytics engine for large-scale data processing. Before spark, developers will have to use different set of tools depending on the use case. If they had batch data processing, they will use Pig or Hive. If they wanted to do real-time data processing, then they will use Apache Storm. However, Spark brings all of them together, and now you can use Spark to do real-time as well as batch data processing use cases under one single unified framework. Spark can run workloads 100 times faster than MapReduce. You can use it interactively from the programming language of your choice, such as Scala, Python, R, and SQL.

Serve

AWS offers a comprehensive suite of cloud-native solutions to allow consumers to query and analyze the data. Common analytics approaches are covered, including Amazon Redshift for data warehousing; Amazon Athena for SQL querying on demand; Amazon EMR for running popular open source frameworks such as Hadoop, Spark, Presto, Flink, and HBase; Amazon QuickSight for business intelligence; and Amazon Elasticsearch Service for logs and text. There is no need to migrate data to different operating environments, helping to avoid the accompanying overhead, costs, effort, or delays. Amazon EMR, Amazon Redshift Spectrum, and Amazon Athena allow consumers to query data directly in Amazon S3 and, with AWS Glue, they all can share the same data catalog. Additionally, artificial intelligence (AI) and machine learning are becoming increasingly popular tools for building smart applications such as predictive analytics and deep learning. To make it more accessible, Amazon Machine Learning abstracts away from the algorithms with wizards, APIs, and guidance. AI services include Amazon Polly for text-to-speech, Amazon Lex for natural language processing and conversational bots, and Amazon Rekognition for image identification and classification.

8. Conclusion

To summarize, below is the architecture of the proposed data lake:

- A data lake on AWS leverages S3 for secure, cost-effective, durable, and scalable storage. Amazon S3 also offers an extensive set of features to provide strong security for the data lake, including access controls and policies, data transfer over SSL, encryption for data at rest and in motion, logging and monitoring, and more. Redshift is added to the storage layer for SQL-based access and costs less than $1,000 per terabyte per year.
- Data can be quickly and easily ingested into Amazon S3 from the SQL Server, FTP server and APIs by Direct Connect, AWS Snowball, and Amazon Kinesis.
- For processing and analyzing the data stored in Amazon S3, AWS provides fast access to flexible and low-cost services, like Amazon EMR, Amazon Redshift with Redshift Spectrum, Amazon Athena, and Amazon AI services, so any analytical solution can be rapidly scaled to power any big data applications, meet demand, and improve innovation.
- For governing and securing the data, AWS Glue, Amazon DynamoDB, and Amazon ElasticSearch can be leveraged to catalog and index the data in Amazon S3. Using AWS Lambda functions that are directly triggered by Amazon S3 in response to events such as new data being uploaded, the catalog can be easily kept up-to-date. With Amazon API Gateway, API can be created that acts as a “front door” for applications to access data quickly and securely by authorizing access via AWS Identity and Access Management (IAM) and Amazon Cognito.

Prior to the data lake implementation, the next step is to pin down the use cases. Defining a few most impactful use cases for the data lake should take priority over deciding the technology involved. The defined use cases will help to showcase some immediate returns and business impact of the data lake, which will be key to maintaining project support from the higher up the chain of command, and project momentum.

9. References

1) Data Architecture Design Principles
2) Amazon Direct Connect
3) Amazon Snowball
4) Amazon Kinesis
5) Apache Sqoop: Import/Export data from/to Relational Data Base Systems to HDFS
6) Apache Flume: Ingest log data (any non-relational data) to HDFS
7) Apache Kafka: Ingest real-tome streaming data to HDFS
8) Apache Nifi: Supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic
9) AWS (S3) Simple Storage Service
10) Amazon Redshift
11) Blob Storage
12) Data Lake Storage
13) GCP Cloud Storage
14) Amazon EMR (Elastic Map Reduce): Easily run and scale Apache Spark, Hive, Presto, and other big data frameworks
15) Amazon Athena
16) Amazon Machine Learning
17) Amazon QuickSight
18) Amazon SageMaker
19) Amazon Lambda
20) Apache MapReduce

21) Apache Pig
22) Apache Hive
23) Apache Spark
24) Amazon Glue
25) Amazon DynamoDB
26) Amazon Elasticsearch
27) Amazon API Gateway
28) Amazon IAM
29) Amazon Cognito