Data Lake Architecture -

A Comprehensive Design Document

Medical Data Processing Company

# Tracker

## Revision, Sign off Sheet and Key Contacts

## Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
| 10/11/2020 | John Mekubo | 0.1 | Initial draft |

## Reviewers / Approval

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
| FirstName LastName | 1.0 | Udacity Reviewer  Enterprise Data Lake Architect |  |

## Key Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Team | email |
| John Mekubo | Data Architect | Medical Data Processing | johnmekubo@gmail.com |

# Purpose

This document outlines the architecture and design of the Data Lake System for a Medical Record Processing company. It contains the following:

* Current technical environment
* Current Data Volumes
* Data Architecture Diagram
* Business Requirements
* technical requirements

The document is a blueprint for the development of a Data Lake solution for the medical record processing company. It will be used to appraise technical team on the design and architecture of the system.

The target audience is the technical team members such as architects, software engineers, enterprise architects and the technical directors.

**In-scope**

* Develop a system that will become single source of truth for insights regarding patient medical records with one storage location for all data.
* Develop a system that will be easy to scale when needs arise.
* Develop a system that avails ready data in formats that are ready for analytical purposes
* Develop a system that is available 24/7 and fault tolerant.

**Out of scope**

* The Medica Records Company envisages a system that would in the future allow for machine learning model building capabilities.
* Achieving near real time reporting in the dashboards displaying patient data.
* Ability

# Problem Statement

The medical records company current data volume is greater than 8TB handled by SQL Server. The company has experienced hyper growth over the past 3 years. However, as the volume of data continues to grow, the existing single node SQL Server is not able to scale. Currently the company can only process the data nightly due to the compute capacity limitations. ETL processes and SQL reporting queries are running slow due to increased data volumes. The company has tried to scale the database server vertically by adding hardware capacity. However, it has not helped significantly in terms of performance and scale. This has led to the database server crashing and rendering the whole system was offline for several hours due to a surge in data. Engineering team has recommended purging some of the older data to reduce load on the server. Also, there is no rapid back-up and recovery plan. No standard ETL or processing tools available, scripts are used instead. Database backups are taken on a nightly basis. However, in the event of a failure restoring the backups to a new database takes hours and systems would be offline during the restore process, leading to risk and poor customer experience. SQL Server has become a single point of failure, hosting critical customer data.

The company has trouble in doing analytics due to insufficient capacity in the current SQL database. No single source of truth for company data as data exists in silos. The company CTO would like to build additional capabilities with the historical data that company has such as building Machine Learning models, and near-real time dashboards containing patient data for each facility without the need to move the data from one system to another.

# 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 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

The Data Lake design follows the following key principles:

1. ***Data size and Location Existing data volume and anticipated volume. Includes definition of ingestion and storage layers.*** The medical records processing company seeks to have all its data in one central place**.** The size and location would aid in planning the capacity requirements of the Data Lake as well as the scaling options available.
2. ***Third party tools integration -*** e.g use of Hortonworks HDP(Ambari) to manage Hadoop cluster and Apache NiFi for data ingestion. The ability to integrate third party tools will offer flexibility for the technology used. Users will have flexibility to choose type of tool to use. E.g type of analytical and visualization tool for one hospital may differ from what another hospital uses.
3. ***Separation of storage from processing.*** To allow flexibility of processing tools. This allows for easy selection of tools for processing such as Spark and Hive according to the data processing requirements.

# Assumptions & Risks

* That the Medical Records company has the required technical staff to administer the Data Lake.
* Hadoop Framework runs on Linux. The company may be using Windows software.

**Questions addressed while designing the architecture**

* What size of the data to be ingested?
* Which data formats of source data are to be ingested?
* How frequent is the incoming data?
* How is the data will be accessed?
* Is the system Scalable?
* Is the system and data hosted in it Secure?
* Is the system available and accessible?
* Where is the data coming from?
* Who is going to use the data?

***Missing information in the problem statement***

Information on current and expected number of users. I made the assumption based on Size of data, existing clients and predicted growth of clients.

Data governance issues are not clear. For instance, how long patient data is to be retained in the company storage.

***potential risks that may be created now or in future based on these assumptions***

The Company may have to invest in totally new hardware to support the Hadoop cluster.

Lack of trained technical personnel to administer and manage the Data Lake may lead to difficulties in managing the system.

# Data Lake Architecture for Medical Data Processing Company

Diagram

Description automatically generated

Figure 1 Data Lake Architecture

# Design Considerations and Rationale

## Ingestion Layer

**<How do you plan to ingest different types of data?>**

Data from text, CSV and XML files will be ingested via Apache NiFi pipelines. NiFi is very efficient in ingesting such files.

Data from client FTP servers and APIs will be consumed by Kafka as this data is continuous and would be best captured as streams (data in motion).

Relational database data will be ingested using scoop as it is the best tool to move data in this case.

**Ingestion layer design scale**

The way the ingestion layer is separated from other system components. It allows for easy addition of tools. I case of additional requirement to add a tool like Flume, it will be just installed and configured with ease.

**Other tools considered**

There are similar tools that can do the same data ingestion function. Gobblin for instance is an ETL tool developed by LinkedIn that ingest data from FTP/SFTP servers. Flume is also a good stream ingestion tool especially when dealing with back-pressure. Apache Storm is another stream ingestion tool that may be used in this layer.

These tools were not chosen as NiFi, Kafka and Scoop work and integrate better with Hadoop.

## Storage Layer

Hadoop distributed file system is the most appropriate storage technology to use in the storage layer. Hadoop is open source and easy to scale as it only requires additional commodity hardware to scale horizontally. This will easily cater for the 20% YoY Data Growth rate.

Hadoop is designed for fault tolerance. The distributed nature of Hadoop nodes will ensure 24/7 availability and reliability. The name node can be increased to two or three in case active name node goes down another will take over offering round the clock availability.

Metadata can be stored in Postgres database server which is separate from the Hadoop storage. Hive metastore can be stored in postgres as well. The ideal data format for the system will be Parquet as it column oriented and efficient in querying. It has high compression rate and that reduce storage requirements.

**Security**

Hadoop security can be managed using Ambari, a framework for managing Hadoop in cloudera/hortonworks platform. In Ambari, Ranger can be used to manage entire Hadoop platform security for the company users. Apache Knox proxy can be used to control access from remote connections/users to extend Hadoop access to users outside of Hadoop.

**Other tools considered**

Alternative tools that may be used for this use case include Snowflake. Snowflake is a cloud framework for ETL, staging, processing and storage of data in the cloud. Another option is use of Amazon EMR framework together with its S3 storage. However Amazon EMR comes with high costs of maintenance.

Hadoop was chosen because it is open-source and can easily deliver same results.

## Processing Layer

The data processing will be done using a combination of batch and stream processing. The CSV, XML and text files can be processed in batches then ingested into HDFS/ Hive or HBase depending on the needs. The data may be processed further by aggregation, cleaning and transformation using hive and store in HBase. For SFT/SFTP and API data, processing can be done using Spark streaming to ensure fast processing speed. Once the streams are processed into required format the data can be accessed by SparkMlib for machine learning. This Data Lake architecture allows for ad-hoc interactive querying of data stored in HDFS or Hive data warehouse via terminal using HiveSQL, or even querying the data store- HBase by use of SQL.

**Other tools considered**

Tools used in processing include Hive, Spark, SparkMlib. Other tools that may be used in processing include MapReduce, Apache Pig and Hue which use SQL like querying in terminal and GUI query editor. Hive is the native data warehousing solution for Hadoop. Spark is better than MapReduce as it utilizes memory catching hence reducing the need to always access disk for frequently accessed queries. This makes it efficient and faster.

The Data Lake architecture proposed is flexible and allows for adding more processing power. For instance, Machine learning processing capability e.g Spark ML can be easily added to facilitate use of libraries like TensorFlow. Processing power can be increased by adding additional cheap commodity hardware to the cluster.

## Serving Layer

The serving layer is where data is accessed by applications that visualize and analyze data to derive insights and generate reports. Data stored in serving layer include processed data stored in HBase (NoSQL) database and Postgres relational database. The data stored in serving layer is cleaned, processed, aggregated, and summarized according to various needs.The data in the serving layer may be consumed by data analytics and visualization applications like Tableau, PowerBI and Qlik Sense. Web and mobile applications can access data via APIs for real-time or near real-time analysis and visualization. The layer provides flexibility for the users to choose a tool that they like for their own analysis and dashboarding. Additionally, there is room for interactive querying of data using tools like HiveQL, SparkSQL and Jupiter.

# 7. Conclusion

The Data Lake architecture and design outlined in this document if implemented fully as is will solve problems currently faced by the medical records data processing company. Once approved, the next steps will be setting up a project management team that will be tasked with implementation.

# 8. References

*1. Problem Statement mostly based on information contained in the* ***company profile document*** *provided as resource for this project.*

*2.* [*https://aws.amazon.com/emr/*](https://aws.amazon.com/emr/)

*3.* [*https://www.mssqltips.com/sqlservertutorial/9284/snowflake-tutorial-overview/*](https://www.mssqltips.com/sqlservertutorial/9284/snowflake-tutorial-overview/)