Big Data Pipeline Architecture for Healthcare Analytics

Dipika Bogati
Radhika Srivastava
Sailesh Kafle
Sushant Nepal

Introduction

- Design a scalable, secure **data pipeline** to process and analyze 50TB of healthcare data

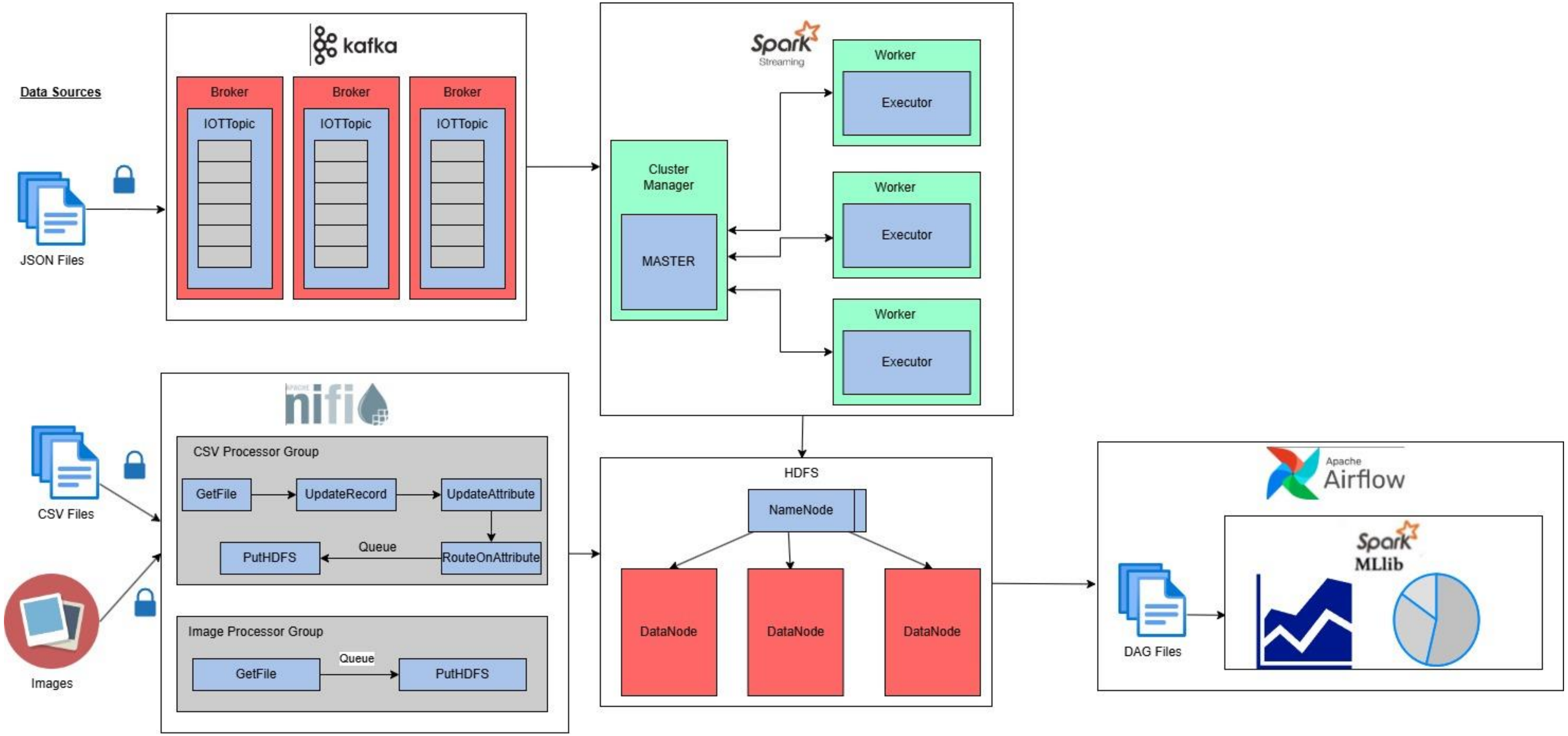
- Handle structured (CSV), semi-structured (IoT JSON), and unstructured (medical imaging) data

- Manage **50TB** of historical records and real-time data from **10,000 IoT** devices generating **1GB/hour**

- Ensure **HIPAA** compliance while managing **inconsistencies** (missing fields, duplicates)

- Build a Hadoop-based ecosystem to ingest, store, and analyze this **data securely and efficiently**

Architecture Design



Data Ingestion and Storage: Apache NIFI

Handling batch ingestion of structured patient records (CSV)

Unstructured medical imaging files (binary format)

Efficient handling of large volume CSV files (50 TB)

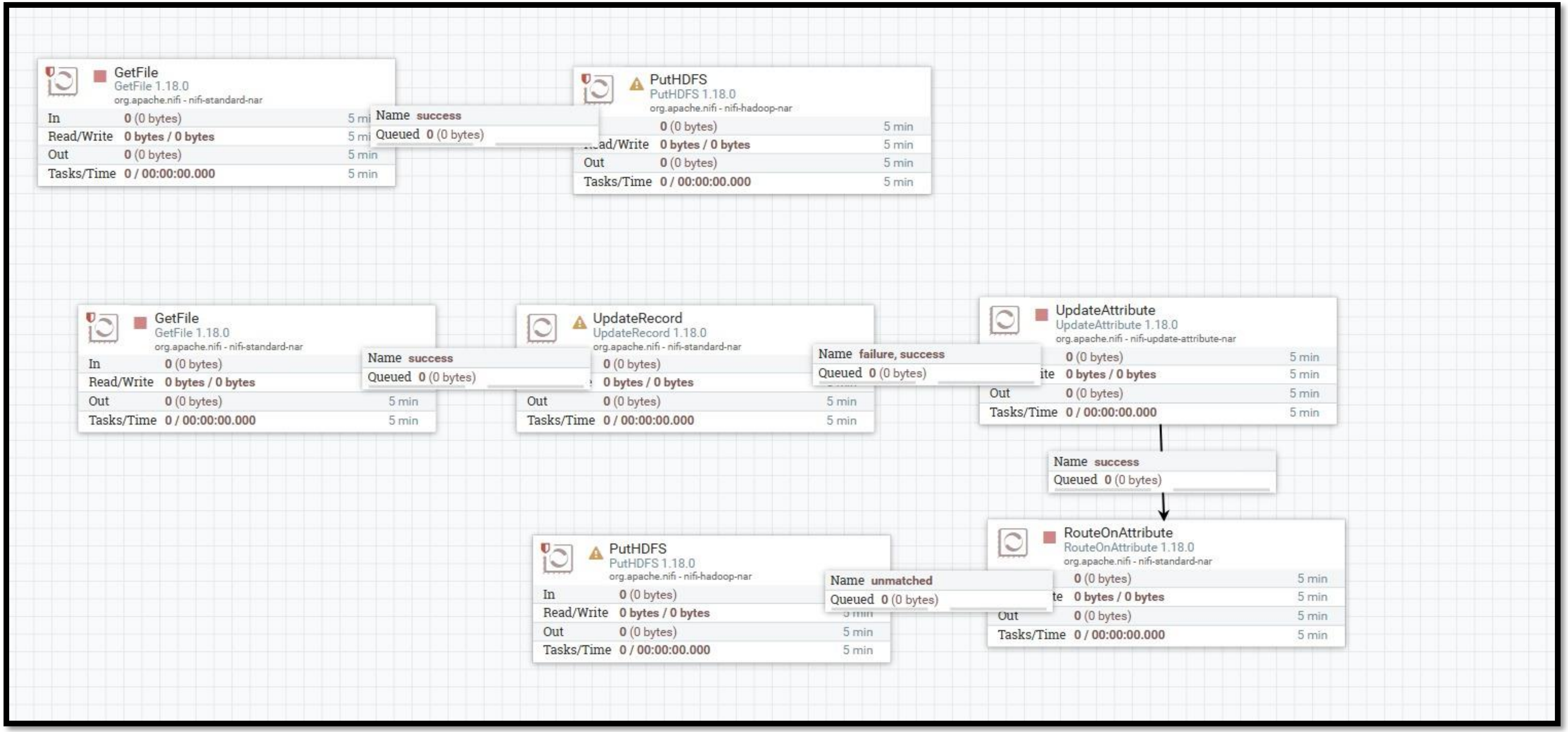
Supports binary file ingestion without loss of data quality

Scalable and fault-tolerant data flow management

Easily integrates with HDFS for data storage

Provides backpressure and retry mechanisms for stable data ingestion

Apache NIFI Design Flow



GetFile
GetFile 1.18.0
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 mi
Read/Write 0 bytes / 0 bytes 5 mi
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name success
Queued 0 (0 bytes)

PutHDFS
PutHDFS 1.18.0
org.apache.nifi - nifi-hadoop-nar
0 (0 bytes) 5 min
ad/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

GetFile
GetFile 1.18.0
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes)
Read/Write 0 bytes / 0 bytes
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name success
Queued 0 (0 bytes)

UpdateRecord
UpdateRecord 1.18.0
org.apache.nifi - nifi-standard-nar
0 (0 bytes)
0 bytes / 0 bytes
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name failure, success
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 1.18.0
org.apache.nifi - nifi-update-attribute-nar
0 (0 bytes) 5 min
ite 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name success
Queued 0 (0 bytes)

RouteOnAttribute
RouteOnAttribute 1.18.0
org.apache.nifi - nifi-standard-nar
0 (0 bytes) 5 min
te 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name unmatched
Queued 0 (0 bytes)

PutHDFS
PutHDFS 1.18.0
org.apache.nifi - nifi-hadoop-nar
In 0 (0 bytes)
Read/Write 0 bytes / 0 bytes
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Data Ingestion and Storage: Apache Kafka

Ingesting high-speed real-time JSON data from 10,000 IoT devices.

Ensures durability and fault tolerance by replicating data across multiple brokers.

Decouples data producers (IoT devices) from consumers (Spark Streaming), allowing for asynchronous processing.

Data is stored reliably in Kafka topics for processing by downstream system

Kafka partitions topics for horizontal scalability and parallel processing.

Guarantees no data loss with retention capabilities, even during processing delays.

Apache Kafka

Producer : IOT devices

Kafka Cluster:

3 brokers as 3 replication

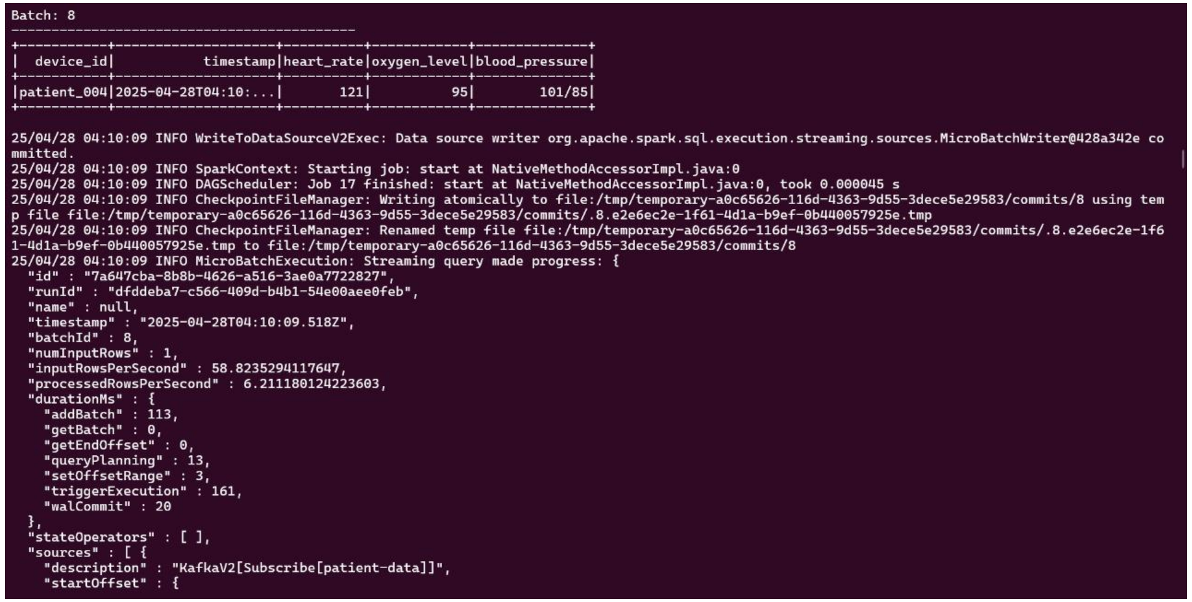
One topic (IOTTopic)

6 partitions

Consumer : Spark Streaming

Spark Streaming

- Consumer of the Apache Kafka.
- Processing real-time data.
- Provides data validation, cleaning, and real-time
    - Anomaly detection (e.g., heart rate spikes, low oxygen levels)
    - Null Value
    - Duplicate Value
- Processes and writes cleaned data to HDFS in Parquet format for further analysis.



```
Batch: 8
-------------------------------------------
+-----------+--------------------+----------+------------+--------------+
|  device_id|           timestamp|heart_rate|oxygen_level|blood_pressure|
+-----------+--------------------+----------+------------+--------------+
|patient_004|2025-04-28T04:10:...|       121|          95|        101/85|
+-----------+--------------------+----------+------------+--------------+

25/04/28 04:10:09 INFO WriteToDataSourceV2Exec: Data source writer org.apache.spark.sql.execution.streaming.sources.MicroBatchWriter@428a342e co
mmitted.
25/04/28 04:10:09 INFO SparkContext: Starting job: start at NativeMethodAccessorImpl.java:0
25/04/28 04:10:09 INFO DAGScheduler: Job 17 finished: start at NativeMethodAccessorImpl.java:0, took 0.000045 s
25/04/28 04:10:09 INFO CheckpointFileManager: Writing atomically to file:/tmp/temporary-a0c65626-116d-4363-9d55-3dece5e29583/commits/8 using tem
p file file:/tmp/temporary-a0c65626-116d-4363-9d55-3dece5e29583/commits/.8.e2e6ec2e-1f61-4d1a-b9ef-0b440057925e.tmp
25/04/28 04:10:09 INFO CheckpointFileManager: Renamed temp file file:/tmp/temporary-a0c65626-116d-4363-9d55-3dece5e29583/commits/.8.e2e6ec2e-1f6
1-4d1a-b9ef-0b440057925e.tmp to file:/tmp/temporary-a0c65626-116d-4363-9d55-3dece5e29583/commits/8
25/04/28 04:10:09 INFO MicroBatchExecution: Streaming query made progress: {
  "id" : "7a647cba-8b8b-4626-a516-3ae0a7722827",
  "runId" : "dfddeba7-c566-409d-b4b1-54e00aee0feb",
  "name" : null,
  "timestamp" : "2025-04-28T04:10:09.518Z",
  "batchId" : 8,
  "numInputRows" : 1,
  "inputRowsPerSecond" : 58.8235294117647,
  "processedRowsPerSecond" : 6.211180124223603,
  "durationMs" : {
    "addBatch" : 113,
    "getBatch" : 0,
    "getEndOffset" : 0,
    "queryPlanning" : 13,
    "setOffsetRange" : 3,
    "triggerExecution" : 161,
    "walCommit" : 20
  },
  "stateOperators" : [ ],
  "sources" : [ {
    "description" : "KafkaV2[Subscribe[patient-data]]",
    "startOffset" : {
```

Data Ingestion and Storage: Hadoop Distributed File System

Storing structured and unstructured data for large datasets (50 TB patient records, IoT data, and medical images) for scalability and reliability.

Ensures high-availability by removing secondary-point of failure and adding secondary name node.

Added warm tier and cold tier to archieve the old data that is more cost – friendly and does not delete the old data as well.

Apache AirFlow

Ensures efficient workflow management at scale with built-in fault tolerance

Automates the schedule of Apache SparkMLLib jobs on the period of time.

Monitoring of the visualization pipeline is efficient and easy in Apache Airflow.

SparkML

SparkML jobs are scheduled to run hourly to process and analyze the latest data

The program retrieves the processed data from HDFS for analysis

SparkML is used to build and train machine learning models on healthcare data to generate insights

The models provide intelligence that helps in solving healthcare problems, such as identifying patient trends and predicting health risks

Visualizations based on model outputs are generated to monitor and track health trends over time

Security Consideration

Role-based access control for ingestion, HDFS permissions for data storage, and detailed audit logging

Masking of fields (e.g., patient names, SSNs) during ingestion and processing, ensuring privacy protection.

Transparent Data Encryption (TDE) for data at rest in HDFS, ensuring protection against storage device breaches.

Real-time monitoring of the system, with centralized secret management and alerts for suspicious activities

Conclusion

Warm tier and Cold tier for old data to reduce storage and cost.

Minimum privileges, masking PII data, SSL/TLS data ingestion maintain the HIPAA policy.

Decided the resources in such a way that there will be no resource over-head and efficient for the data flow.

Pipeline is scalable as resources can be added in the system without designing if needed in the future.

Any Questions?