

Case Study

- 1-IOT DATA PROCESSING
- 4-DATA PROCESSING OPTIMIZING

Srinija Kambhampaty
23rd December, 2024

1-IOT DATA PROCESSING

- Produce Real time Sensor data.
- Ingest Real Time Data
- Perform a Spark Batch Job for aggregation, Partitioning, Storing high- frequency data
- Automated Spark Job to delete old files
- Akka Http APIs to get data from storage
- Webpage to display the retrieved data








**Akka / Kafka / Protobuf / Spark / Google Cloud
Storage / JSON / Angular**

PRODUCING, PUBLISHING DATA

- SensorId : Random number in the range of [1,20]
- TimeStamp: Time at which the message is generated
- Temperature: Random float value in range of (-50,150)
- Humidity: Random float value in range of (0,100)
- Checksum Addition for Validation
- How ? Add Up All the Bytes, Split the Sum into Two Bytes, Return as Array
- Why? Verification, Efficiency, Low Overhead

`finalPayload = messageBytes + checksumBytes`

SPARK JOB - PROCESSING, STORAGE

-  Data Ingestion: Streams sensor data from Kafka (sensor-readings topic).
-  Validation: Validates data integrity using a checksum.
-  Deserialization: Converts Protobuf messages into structured data.
-  Partitioning: Organizes data by year, month, day, hour for efficient querying.
-  Aggregation: Computes avg, min, max values for temperature and humidity.
-  Incremental Computation: Merges new batch data with existing aggregated data, ensuring incremental updates instead of full recomputation.
-  Storage: Writes raw & aggregated data to Google Cloud Storage (GCS) in Binary & JSON formats.

APIS AND WEBPAGE

- GET /api/aggregated-data
- GET /api/aggregated-data/:sensorId

Sensor Dashboard

Q

Search by Sensor ID

REFRESH

Sensor ID	Max Temp (°C)	Min Temp (°C)	Avg Temp (°C)	Max Hum (%)	Min Hum (%)	Avg Hum (%)
sensor-14	144.38213	-25.963116	44.892041524251304	98.99461	0.4975915	46.68814388910929
sensor-5	147.28601	-49.79532	43.34910063743591	98.98124	1.3794899	52.87135496735573
sensor-15	139.5441	-35.580837	52.35545471736363	98.1404	6.514329	48.417285442352295
sensor-4	143.33984	-45.29419	46.81761108125959	99.87413	0.18576384	52.43080461238112
sensor-16	104.659836	-45.184135	26.67456595102946	82.95709	12.962145	53.984077612559
sensor-18	147.35121	-43.537594	73.32830483572823	98.63535	3.4859478	54.16736652169909
sensor-6	139.85124	-42.484486	56.032775004704796	99.24204	0.23053885	38.568687538305916
sensor-10	148.29504	-19.93959	71.45606327056885	93.91644	11.584848	51.59092456102371
sensor-11	125.08769	-40.658722	68.26686520046658	94.55674	3.6001086	55.22199477089776
sensor-2	149.95865	-49.663734	52.28431582900713	99.84322	1.3147116	50.32239919563509
sensor-3	148.58833	-46.3055	54.04702220243566	96.708206	0.08111596	49.082303358059306
sensor-17	123.55832	-35.448	63.28305080958775	95.94921	0.46437383	49.968835302761626
sensor-9	112.02287	-29.721296	23.991934299468994	96.13486	0.3094554	54.19291500250498
sensor-7	142.27602	-42.473324	55.789211016007	97.74057	10.500082	57.92227165065894

Sensor Dashboard

Q

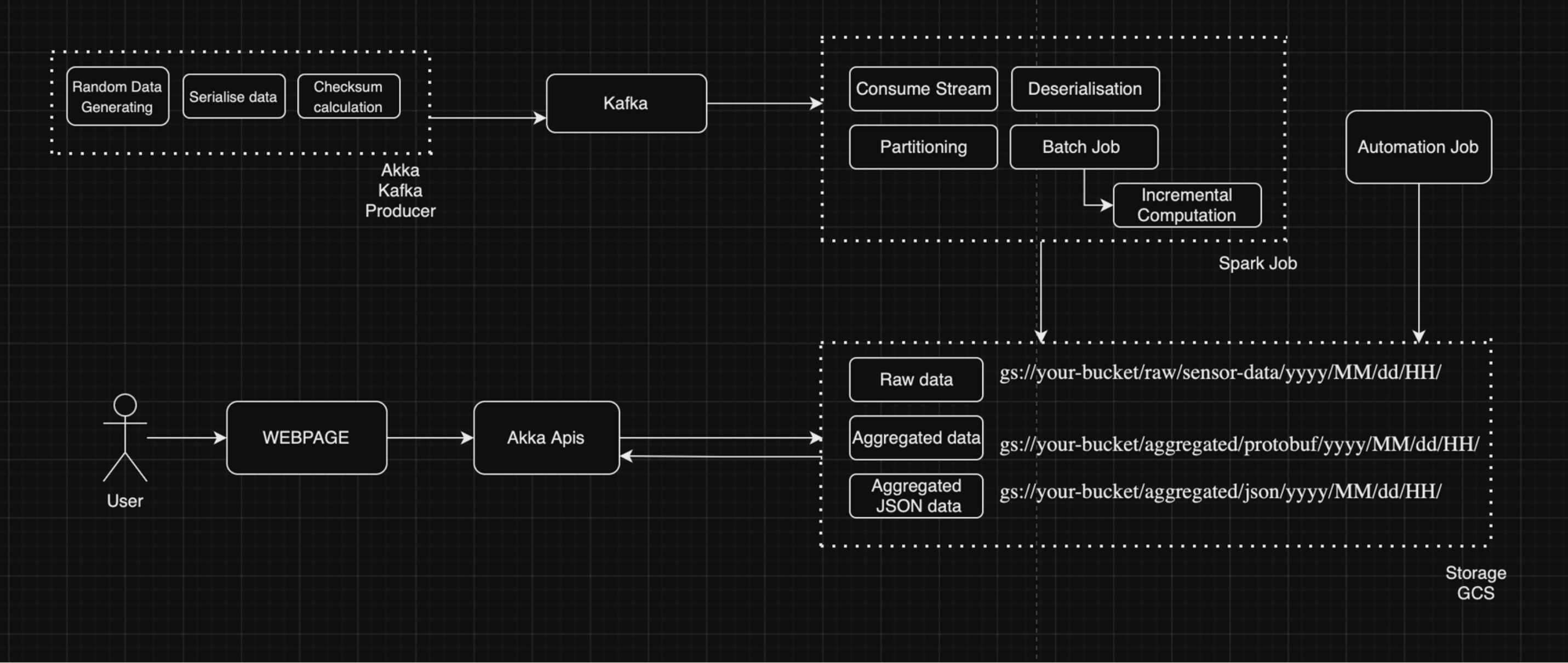
sensor-14

X

REFRESH

Sensor ID	Max Temp (°C)	Min Temp (°C)	Avg Temp (°C)	Max Hum (%)	Min Hum (%)	Avg Hum (%)
sensor-14	144.38213	-25.963116	44.892041524251304	98.99461	0.4975915	46.68814388910929

SYSTEM ARCHITECTURE



4 - DATA PROCESSING OPTIMIZATION

- Static Data, Dynamic Data Produced ,
Serialised and Published
- Subscribed, Deserialised
- Incremental Computation
- Aggregation
- Storage

**Akka / Kafka / Protobuf / Spark / Google Cloud
Storage / JSON**

PRODUCING, PUBLISHING DATA

Static Data: Walmart Recruiting DataSet

- train.csv: Contains historical weekly sales data for various stores and departments.
- features.csv: Contains additional data about stores and regions.
- stores.csv: Contains store metadata.

Dynamic Data: Random generation of more train data, and publishing into topic

- Store ID: Randomly selects a store ID between 1 and 10.
- Department ID: Randomly selects a department ID between 1 and 20.
- Date: Uses the current date (YYYY-MM-DD).
- Weekly Sales: Generates random sales values between 0 and 10,000.
- Holiday Status: Randomly assigns a boolean value (true/false).

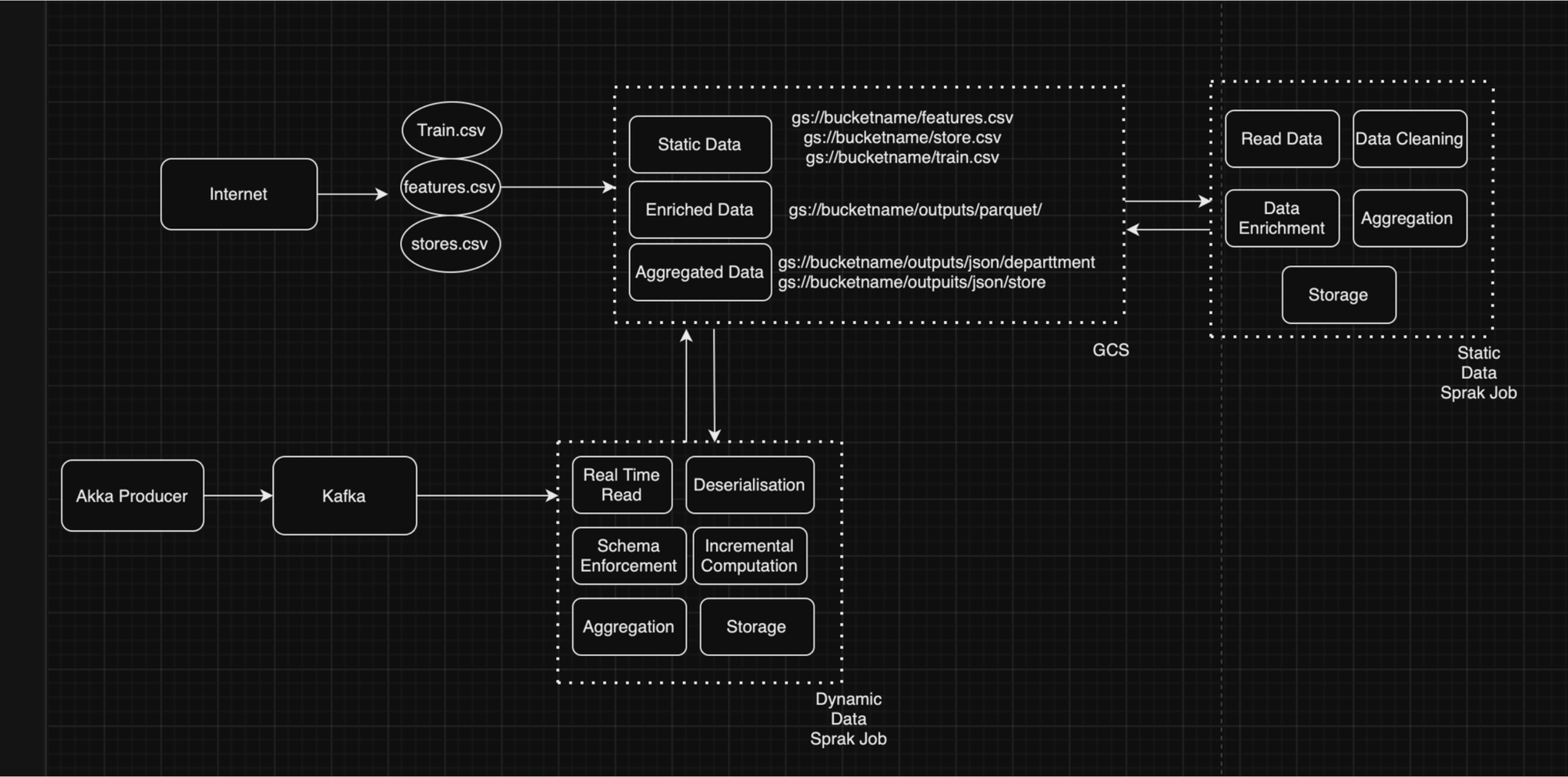
SPARK JOB - STATIC DATA

- Data Sources: Static datasets (train.csv, features.csv, stores.csv) from Google Cloud Storage (GCS).
- Validation & Cleaning: Ensures no missing or invalid values.
- Data Enrichment: Joins train, features, and stores datasets for richer insights.
- Aggregations:
 - Store-Level Metrics: Total, Average, Top Weekly Sales.
 - Department-Level Metrics: Total Sales, Holiday vs Non-Holiday Sales.
- Storage:
 - Partitioned Parquet (Store, Date) for enriched data.
 - JSON for aggregated metrics (Store, Dept, Date)

SPARK JOB - DYNAMIC DATA

- Real-Time Ingestion: Stream data from Kafka Topic (weekly-sales-data) into Spark Streaming.
- Deserialization: Convert Protobuf messages into structured Spark DataFrame.
- Schema Enforcement: Validate schema consistency with historical data.
- Incremental Computation: Merge incoming data with historical data.
- Perform aggregations incrementally for Store and Department (same as before)
- Storage:
 - Update historical data in CSV format.
 - Save aggregated metrics as JSON in partitioned directories (Store, Dept, Date).

SYSTEM ARCHITECTURE



Thank you!
