Proposed Architecture for Real-Time ICPS Monitoring System.

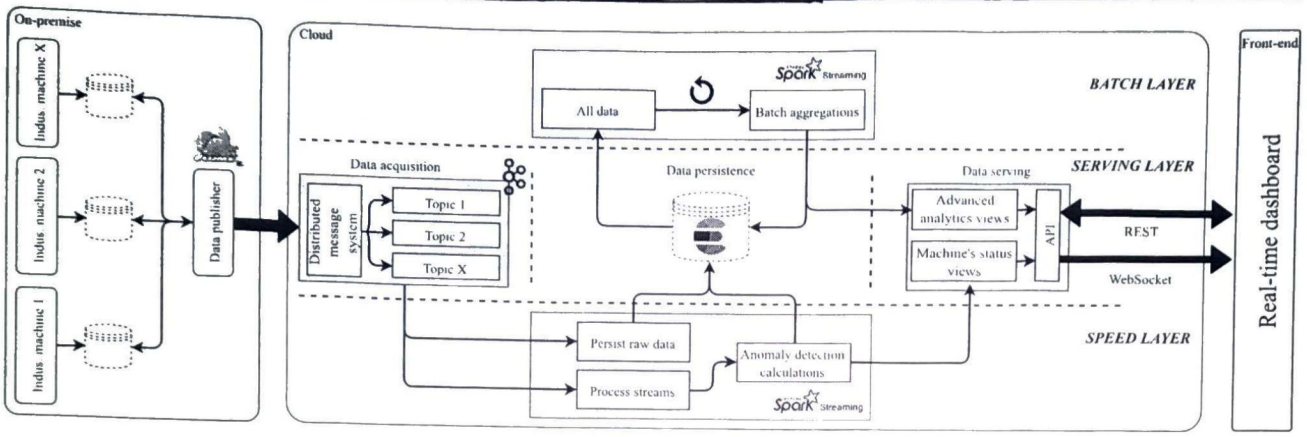


FIGURE 1. Architecture of the proposed ICPS real-time monitoring system.

- Stages :
  1. Data Acquisition
  2. Processing
  3. Persistence
  4. Data Serving.

- Scalable Compute.

- Layers : ① Kafka for Data Acquisition : publish- subscribe model + Flume.
  ② Realtime + Batch processing using Spark Streaming (detect Anomalies).
  ③ Persistence : Distributed storage + querying : Elastic Search + Influx DB.
  ④ Zookeeper : Cloud Resource Management.
  ⑤ Serving : Rest API : Querying + Websocket : Dashboard + communication.

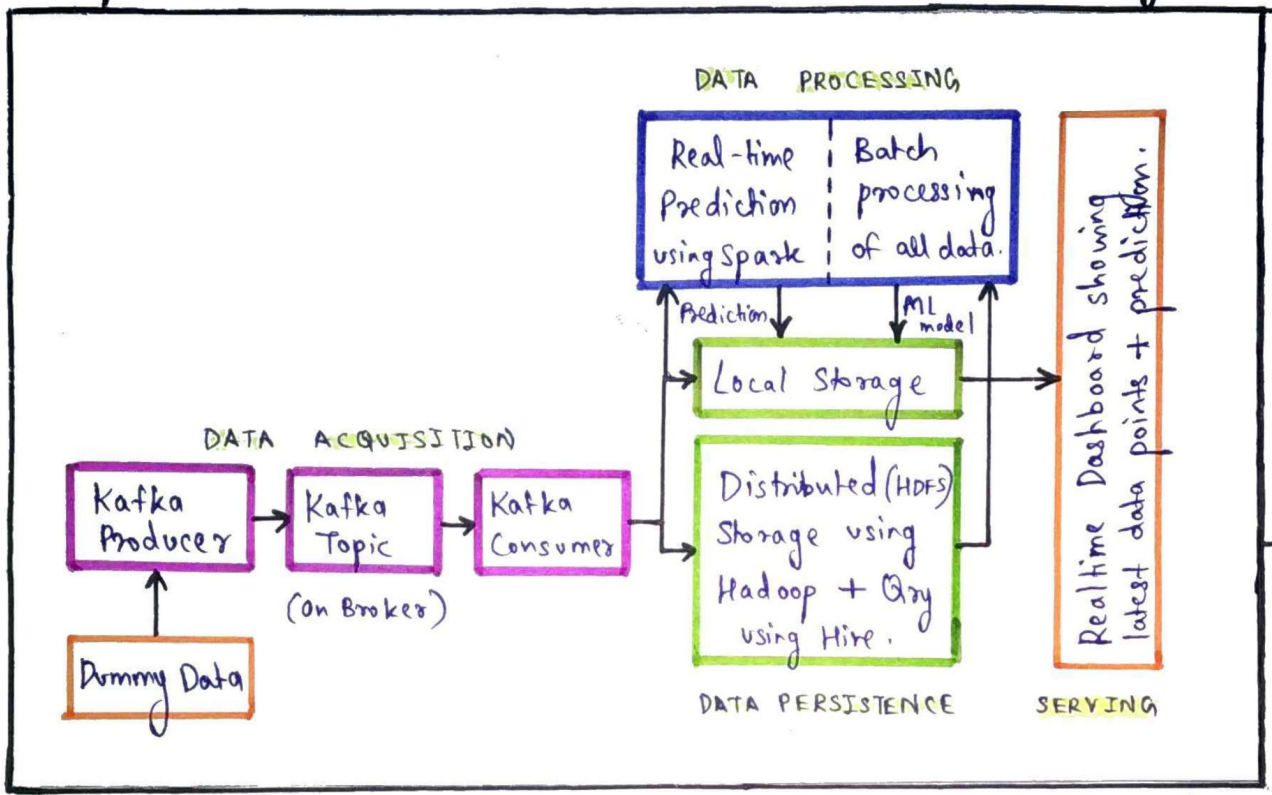
- Detection: ① Single Data Point (SDAD)
  ② Multiple /Batch (MDAD).

- Future: Predict early repair work to increase OEE (Overall Equipment Effectiveness)

- Evaluation: (Scale tests)
  - Input rate
  - Scheduling delay
  - Processing time
  - Delay Time.

- Reference: 10.1109/ACCESS.2019.2911979

Implemented Architecture for ICPS Anomaly Detection System.



- Layers:

① Data acquisition:

A stream of data points is simulated using Kafka's producer and consumer.

② Persistence:

The data points are stored in HDFS, in a distributed manner. ③ Data Processing: The stored data is batch-processed using Apache Spark. Also, the output of it: A trained ML model is saved for predicting the labels of the incoming stream of data points.

④ Data Serving: The latest data points are visualized using a Dashboard.

- Tech Stack:
  - Hadoop (HDFS, YARN, Hive, Zookeeper)
  - Spark
  - Pyspark
  - Kafka
  - Python 3
  - Plotly-Dash
  - Colab / Jupyter NB.
  - Bash
  - Hosted on AWS EC2 virtual machines.

Dataset Used : Electra_Modbus Dataset (12 hr traffic captured) — 4 nodes.

- **Features:**
  1. time : timestamp of packet.
  2. smac : Source MAC address.
  3. dmac : Destination MAC address.
  4. sip : Source IP address.
  5. dip : Destination IP address.
  6. request : 1 = master to slave comm^{n}.
  7. fc : function code. (Read holding register (3) / Read input (2) ... others = anomalies).
  8. address : Memory address to perform Read / Write operation.
  9. error : Indicates an error in Read / Write operation.
  10. Label : Labels for attacks / normal samples.
  11. data : Read / written data.

- **Labels:**
  1. Normal
  2. MITM_UNALTERED ≈ Normal (Man in the Middle node is active but does no change).
  3. READ_ATTACK : fc = 2 — packet generation.
  4. WRITE_ATTACK : Wrong data packet generated.
  5. FUNCTION_CODE_RECOGNITION_ATTACK : Generating wrong fc packets.
  6. REPLAY_ATTACK : Replaying normal packets at wrong time intervals.
  7. RESPONSE_MODIFICATION_ATTACK : Changing data inside a normal packet (IP changes).
  8. FORCE_ERROR_ATTACK : Modifying error field of normal packet.

- **Experiments:** (ML) fc = 3
  1. SVM : 99% precision score.
  2. Random Forest : 97%.
  3. Neural Network : 94%

- **Reference:** 10.1109, ACCESS.2019.2958284.

Future Scope :

1. Real-time push notifications / alerts.
2. Auto-scaling of cluster size.
3. Using advanced deep learning models for batch-processing.
4. Automating the entire pipeline using scripts.
5. Deploying the system in actual industrial system use cases.

- Samruddhi Khairnar, TE-29.