Real-Time Processing Data Using Apache Kafka, Apache Storm, Elasticsearch and Kibana

Panteleakos Thomas  
*National Technical University of Athens  
School of Electrical and Computer Engineering*  
Athens, Greece  
thomasgpant@gmail.com Zerzelidis Nikolaos   
*National Technical University of Athens  
School of Electrical and Computer Engineering*  
Athens, Greece  
nickzerze@gmail.com

*Abstract*— In today's world of big data, real-time data processing has become a necessity for many organizations, as it is crucial for various applications, including fraud detection, predictive maintenance, and IoT analytics. Apache Kafka, Storm, Elasticsearch, and Kibana are four widely used tools in the field of real-time data processing. In this paper, we describe a real-time data processing pipeline that uses these four tools to process and visualize data in real-time. Specifically, we use a data generator that simulates data from various sources, Apache Kafka to ingest data, Apache Storm to process data, and Elasticsearch and Kibana to store and visualize data. We describe the architecture of our pipeline, the various components involved, and the benefits of using this pipeline for real-time data processing.

Keywords—Apache Kafka, Apache Storm, Elasticsearch, Kibana, messaging system, processing data.

# Introduction

In the age of big data, organizations generate vast amounts of data every day. This data can be in the form of log files, sensor data, social media data, or any other form of structured or unstructured data. In order to extract insights and value from this data, it needs to be processed in real-time. Real-time data processing refers to the ability to process data as it is generated, rather than waiting for batch processing. Apache Kafka, Apache Storm, Elasticsearch and Kibana are four technologies that can be used together to provide a highly scalable and reliable solution for real-time data processing, storage and visualization.

Apache Kafka is an open-source distributed event streaming platform that is used to store and process real-time data streams. Kafka is designed to handle large volumes of data and is highly scalable. It provides low-latency data processing, which makes it an ideal choice for real-time data streaming applications.

Apache Storm is a distributed stream data processing system, which has been used in Twitter for

various critical computations.

Apache Storm is a distributed real-time processing system that provides a platform for the processing of streaming data in real-time, which has been used in Twitter for various critical computations. It can process large volumes of data with low latency and can be used to perform complex data transformations in real-time.

Elasticsearch is a search and analytics engine that is capable of storing and indexing large volumes of data. It provides a scalable and distributed platform for storing and searching data, and it can be used to store and search data in real-time. Kibana is a visualization tool for Elasticsearch that allows users to create real-time dashboards, charts, and graphs based on the data stored in Elasticsearch. It provides a user-friendly interface for data analysis and visualization, and it allows users to interact with the data in real-time.

In this paper, we propose a solution that combines these four technologies to enable real-time processing and analysis of large volumes of data in a smart home system. We describe the architecture of our proposed solution, the data flow between the different components, and the configuration and deployment of each technology. We also present a case study where data is ingested into Apache Kafka and passed to Apache Storm for real-time processing. The processed data is then stored in Elasticsearch and visualized using Kibana.

The remainder of this paper is organized as follows. Section II provides an overview of the related work in the area of real-time data processing. Section III describes the architecture of our proposed solution, including the data flow between the different components. Section IV presents a case study where we apply our proposed solution to a real-world use in a smart-home topology. Section V describes the steps that we followed to implement the desired system and Section VI 🡪Conclusion????

# Related work

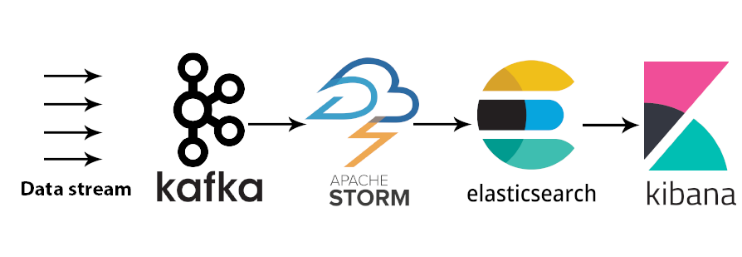
Real-time data processing is a rapidly evolving area of research, and there have been many studies exploring different approaches and technologies for real-time data processing due to its ability to process data as it is generated, allowing for immediate insights and actions.. In this section, we provide an overview of some of the related work in various domains, including smart homes, finance, transportation, healthcare, and more.

One of the most popular real-time data processing frameworks is Apache Spark Streaming, which provides a scalable and fault-tolerant platform for real-time data processing. Spark Streaming allows users to perform complex data transformations in real-time using a high-level programming API. However, Spark Streaming has some limitations, such as high latency and limited support for fault-tolerance.

Apache Flink is another popular real-time data processing framework that provides a highly scalable and fault-tolerant platform for real-time data processing. Flink allows users to perform real-time data processing using a variety of APIs, including streaming SQL, data stream API, and process function API. Flink is highly scalable and can handle large volumes of data with low latency.First, confirm that you have the correct template for your paper size.

Our work focuses on real-time data processing in smart homes using Apache Kafka for data ingestion, Apache Storm for processing, and Elasticsearch with Kibana for visualization and analysis.

# Proposed solution architecture

In this section, we describe the architecture of our proposed solution for real-time data processing, storage, and visualization using Apache Kafka, Apache Storm, Elasticsearch, and Kibana as we can see in figure (1):

## Data Generation

The first component of our proposed solution is a data generator that simulates data from various sources such as IoT sensors. The generated data is used to test and evaluate the performance of the real-time system. By using a data generator, we can create custom data sets with varying sizes and complexities to simulate real-world scenarios.

## Data Ingestion

The second component of our proposed solution is data ingestion. We use Apache Kafka as our data ingestion system.  
Apache Kafka is a distributed streaming platform that can handle millions of messages per second. Apache Kafka provides publish-subscribe system. The producers publish data and the subscriber consumes the data. In Kafka, the messages published by the producers are posted to a topic and the subscriber subscribes to the topic and reads the message.

Kafka is designed to be fault-tolerant and resilient to node failures. It uses a replication factor to replicate data across multiple nodes, ensuring that data is not lost in case of node failure. Furthermore, Kafka can be easily integrated with other tools and frameworks, such as Apache Storm, to build complex data processing pipelines. Finally, it allows us to handle high volumes of data with low latency, provides a reliable way to ingest data in real-time and stores the incoming data in topics, which act as a buffer between the data producers and consumers.

## Real-time Processing

The third component of our proposed solution is real-time processing. We use Apache Storm as our real-time processing system. Apache Storm is a distributed real-time processing system that provides a platform for the processing of streaming data in real-time. Storm splits data into small streams and processing each stream in parallel across a cluster of machines. Storm is highly scalable and can process large volumes of data with low latency.

Storm architecture consists of one master node which is called Nimbus and one or a couple of worker nodes which are called Supervisor. In addition, Storm relies on Zookeeper for managing nodes in Storm cluster. Zookeeper gives information of supervisor’s state to Nimbus. Then, Nimbus assign works to supervisors and each supervisor processes assigned tasks.

In our proposed solution, Storm reads data from Kafka topics and processes the data using a topology, which is a directed acyclic graph of processing nodes. Each node in the topology performs a specific processing task, such as filtering, aggregation, or enrichment. The processed data is then passed to the next node in the topology, and the output of the last node is stored in Elasticsearch.

## Data Storage

The next component of our proposed solution is data storage. We use Elasticsearch as our data storage system. Elasticsearch is a search and analytics engine that is capable of storing and indexing large volumes of data. Elasticsearch provides a scalable and distributed platform for storing and searching data, and it can be used to store and search data in real-time.

Elasticsearch stores the processed data outputted by Storm. Elasticsearch indexes the data based on certain fields, making it easy to search and retrieve the data later. Elasticsearch also provides APIs for querying the data and performing complex analytics.

## Data Visualization

The final component of our proposed solution is data visualization. We use Kibana as our data visualization tool. Kibana is a visualization tool for Elasticsearch that allows users to create real-time dashboards, charts, and graphs based on the data stored in Elasticsearch. Kibana provides a user-friendly interface for data analysis and visualization, and it allows users to interact with the data in real-time.

In our proposed solution, Kibana connects to Elasticsearch and retrieves the processed data. Kibana allows users to create real-time visualizations, such as line charts, bar charts, and pie charts, based on the data stored in Elasticsearch. Kibana also allows users to create dashboards, which are a collection of visualizations that provide a high-level view of the data.

# Use case scenario

Consider a smart home system that uses various sensors to monitor and control the home's temperature, energy consumption, motion detection and water usage. The system includes the following sensors:

* Temperature sensors: Installed two sensors to monitor the temperature in an 15 minutes interval and send the data to the central system for analysis and control.
* Air conditioner energy measurement sensor: Installed two sensors to measure the energy consumption of the air conditioning system in a 15 minutes interval. The central system can use this information to adjust the air conditioning settings and optimize energy usage.
* Energy measurement sensors of other electrical devices: Two sensors have installed also on two various electrical devices such as the refrigerator, oven, and washing machine to measure their energy consumption every 15 minutes. The central system can use this information to track energy usage and identify energy-efficient appliances.
* Cumulative energy measurement sensor: Installed at the main power panel to measure the daily total energy consumption of the house. The central system can use this information to track energy usage and identify areas for improvement.
* Motion detection sensor: One sensor of detection has installed so the system can use this information to trigger security alerts, turn on lights, and adjust other settings based on the occupants' behavior.
* Water consumption measurement sensor: Installed in the main water line to measure the water usage of the house every 15 minutes. The central system can use this information to track water usage and identify areas for improvement.
* Cumulative water consumption measurement sensor: Installed at the main water panel to measure the daily total water consumption of the house. The central system can use this information to track water usage and identify areas for improvement.

Using these sensors, the central system can monitor and optimize the home's energy consumption, temperature, and water usage to reduce waste and save money. For example, the system can automatically adjust the thermostat based on the average temperature of the house, turn off lights and appliances when no one is in the room, and alert the homeowners when water usage is abnormally high.

# IMPEMENTATION

In this section we will describe the steps that we had to follow to setup the desired real-time data processing system.

First, we had to create the mock\_generator.py file that contain the code that generates the random data that simulates the sensors described above. After that we install Zookeeper that is prerequisite for Apache Kafka to work. We had to make the required changes to the configuration of Zookeper and then we launch it.

Next, Apache Kafka has to be installed. We downloaded the binaries from the official website and followed the installation instructions for operating system. We also created the topics~~: average, sum, max ?????(ΘΑ βαλουμε άλλο εκτός του average??)~~ that used from Kafka to pass data to Storm.

Then we downloaded Apache Storm binary from the official webasite and installed it with the appropriate configurations. The topology of Storm is created in LocalCluster mode and contains the following components:

* The KafkaSpout that receives the data from the Kafka producer and pass them to the Bolts,
* Three custom Bolts (tumblingavg, slidingmax, tumblingsum) that receives data from KafkaSpout and process it,
* The custom EsBolt (myesbolt) that receives the precessed data and pass it to the Elasticsearch server.

After that we downloaded the Elasticsearch and Kibana binaries from the official websites and installed. We launched both and connected to the Kibana User Interface (localhost:5601) to create the Indexes: temperature, hvac, miac, mov, water, energy\_total and water\_total. Finally from the Dashboards of Kibana we created the Data Views with the appropriate filters to see the data we want.

# SUMMARY

Θα προσθέσω κι άλλα!!!

We demonstrate the performance of our system through experiments with real-world datasets, showcasing its ability to handle high volumes of data with low latency and high scalability. Our system provides a powerful platform for real-time data processing and analysis, with potential applications in various fields such as IoT, finance, and cybersecurity.

##### References

1. Apache Kafka Documentation, < <https://kafka.apache.org/33/documentation.html> >.
2. Apache Storm Documentation, < <https://storm.apache.org/releases/2.4.0/index.html> >
3. N. Narkhede, G. Shapira, T. Palino, “Kafka The Definitive Guide,” O’Reilly Media, Sebastopol, 2017.
4. M. Kumar, C. Singh, “Building Data Streaming Applications with Apache Kafka,” Packt Publishing, Birmingham, 2017.
5. J. Ankit, “Mastering Apache Storm,” Packt Publishing, Birmingham, 2017.
6. B. Priyadarshini, “Real-Time Streaming with Apache Kafka, Spark, and Storm,” BPB Publications, India, 2022.