

MSDA9215: Big Data Analytics

Apr/May 2025

Week 1: Introduction & Big Data Ecosystem

Temitope Oguntade

Agenda

- Logistics and Introductions
- Course Description
- Course Learning Outcomes
- Student Assessment and Grading
- Course Schedule
- Academic Integrity & Well-being

Logistics

- **Instructor:** Temitope Oguntade
- **Email:** toguntad@andrew.cmu.edu
- **TAs:**
- **Class Time:** 08:30 - 17:00 Sun, 18:00 - 20:30 Wed
- **Course Credit:**
- **Prerequisite:** As may be determined
- **Code Source:**

https://github.com/toguntad/AUCA_IOT/blob/9bd866ac48a6d3071a20007e9ac5a64b3e9020a1/AUCA_IOT.ipynb

Brief

Temitope Oguntade is the CEO and Founder of Chanel's Innovations, a startup dedicated to building innovative, AI-driven metering solutions specifically designed for micro-utilities in Sub-Saharan Africa. He is also a founder of Inyst.co and a fellow of the Industry Innovation Lab, CMU and of Startup|Energy. He holds an M.Sc. in Information Technology from Carnegie Mellon University and a B.Eng in Electrical and Computer Engineering from the Federal University of Technology Minna. With a career spanning over 15 years, Temitope has developed a deep expertise in entrepreneurship, distributed computing, cloud computing, and technology management, driving significant advancements in the tech sector.



It is your turn

Let's start with you

- Who are you?
- What MSc program?
- What's your background?
- What are your expectations for this course?
- Experience with data analytics?

Course Description

The course is designed to deepen learners' expertise in Big Data Analytics, **with a focus on IoT and mobile data applications**.

- Leverages foundational platforms like HDFS and Spark alongside advanced timeseries databases to enhance Big Data processing capabilities.
- Emphasizes real-time data management using Message Brokers and MQTT, addressing the specific needs of IoT device data streams.
- Equips students with skills to analyze, visualize, and interpret Big Data using scalable machine learning algorithms, preparing them to deliver actionable insights in practical settings.
- Focuses on applying these technologies to real-world challenges, enhancing analytical and decision-making skills in IoT and mobile data contexts.

Course Learning Outcomes I

Upon completion of this course, students will:

- Grasp the broad implications of Big Data Analytics in various sectors, emphasizing its impact on IoT and mobile data.
- Demonstrate expertise in fundamental Big Data platforms such as HDFS, HBase and Spark, ensuring the capability to process and manage large datasets.
- Apply knowledge of diverse data storage systems, including key-value (KV) stores, document databases, graph databases, and timeseries databases, to optimize data structuring and retrieval in different Big Data scenarios.

Course Learning Outcomes II

Upon completion of this course, students will:

- Utilize Message Brokers and the MQTT protocol to effectively manage real-time data flows from IoT devices and mobile sources, addressing the unique requirements of streaming data.
- Employ scalable machine learning algorithms to conduct comprehensive analytics on multi-structured data from various platforms, drawing actionable insights from complex datasets.
- Develop sophisticated data visualization skills to clearly present and interpret analytical findings, catering to both general and specialized audiences including mobile analytics.

Key Topics

Introduction and Big Data Ecosystem

- Overview of Big Data Analytics: Scope and applications.
- Incorporating Message Brokers for Big Data applications.
- Understanding and using MQTT in the context of IoT
- Understanding timeseries databases in Big Data.

Data Storage Methods and Real-Time Data Handling

- Introduction to foundational platforms: Hadoop Ecosystem and Spark.
- In-depth exploration of HDFS and its role in the Hadoop ecosystem.
- Introduction to HBase: Concepts, architecture, and how it integrates with Hadoop.
- Discussion on the types and characteristics of databases: KV stores, document databases, and graph databases.

Key Topics(2)

Big Data Processing and Analytics

- Big Data processing frameworks: MapReduce and beyond.
- Analytics algorithms: Understanding the basics and applications.
- Parallel processing and scalability concerns in Big Data.
- Special focus on analytics for mobile and IoT Big Data.

Visualization and Real-world Applications

- Principles of data visualization in Big Data Analytics.
- Mobile issues and solutions in Big Data contexts.
- Case studies: Real-world Big Data challenges and solutions.
- Industry Case study: Generating insights & visualizations using Natural Language.

Assessment

Component	Weight
Final Exam/Project	40%
Midterm	30%
Assignments Quizzes Participation*	30%

- * Past students have reasonably increased their grades with participation bonus points
- Project completion is a fundamental expectation in this course. Zero point for it

Question

How can you tell that you derived value from this course at the end of it?

What is Big Data Analytics

Overview: Big Data Analytics involves examining large and varied data sets to uncover hidden patterns, unknown correlations, market trends, customer preferences, and other useful information.

Mobile Big Data as a Subset: Advances in mobile computing, mobile internet, IoT, and crowdsensing have intensified the generation of Mobile Big Data (MBD). This data comes from a vast array of mobile and wireless devices, capturing diverse information from sensors carried by moving objects, people (e.g., wearables), or vehicles.

Significance: The analysis of MBD and other Big Data sources provides critical insights that can drive decision-making and operational efficiencies across multiple sectors.

Applications: Big Data Analytics is pivotal in fields like healthcare, finance, retail, and urban planning, where large-scale, real-time data analysis can lead to impactful outcomes.

Examples & Use Cases

A-Some Examples- Transportation & urban planning

Source:

- Guo, Y., Zhang, J., & Zhang, Y. (2017, March). A method of traffic congestion state detection based on mobile big data. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)* (pp. 489-493). IEEE.

Summary:

- K-Means clustering used to cluster statistical average speeds of all vehicles on the road as *serious congestion, congestion, slight congestion, smooth*.

Use Case:

- Traffic congestion state detection
- Uses mobile big data (data generated by mobile phones' connection to base stations).
- How do you distinguish between connections from residents, pedestrians, etc. vs. those from vehicles in traffic? (By average travel speed based on switching between base stations.)

B-Some Examples- Transportation & urban planning

Source:

- Batran, M., Arai, A., Kanasugi, H., Cumbane, S., Grachane, C., Sekimoto, Y., & Shibasaki, R. (2018, July). [Urban Travel Time Estimation in Greater Maputo Using Mobile Phone Big Data](#). In 2018 IEEE 20th Conference on Business Informatics (CBI) (Vol. 2, pp. 122-127). IEEE.

Summary:

- Uses call data records (CDR)- i.e., phone activities e.g., voice calls, text messages or mobile data connections.
- CDR used to characterize mobility.
- GPS location history of 600 users collected as ground truth.

Use Case:

- Uses GPS trajectories of a set of users.
- Considers segregating travel times for different time categories: weekday rush hour, weekday non-rush hour, & weekends.
- Travel time estimation from mobile phone data.
- To help design better transportation systems.

C-Some Examples- citizen engagement

Source:

- Ruiz-Correa, S., Santani, D., Ramirez-Salazar, B., Ruiz-Correa, I., Rendon-Huerta, F. A., Olmos-Carrillo, C., ... & Gatica-Perez, D. (2017). SenseCityVity: Mobile crowdsourcing, urban awareness, and collective action in Mexico. IEEE Pervasive Computing, 16(2), 44-53.

Summary:

- Mobile crowdsourcing of geolocalized images, audio, and video.
- Analysing the data for community reflection

Use Case:

- Get youth to define, document, and reflect on their city's problems—citizen engagement.

D-Some Examples- mobile provider operations

Source:

- Xu, F., Lin, Y., Huang, J., Wu, D., Shi, H., Song, J., & Li, Y. (2016). Big data driven mobile traffic understanding and forecasting: A time series approach. IEEE transactions on services computing, 9(5), 796-805.

Summary:

- Extracts and models traffic patterns of 9,000 cellular towers in a metropolitan city.
- Uses time series approach.
- Decomposes the regular and random components of the mobile traffic.
- Uses time series prediction and correlation to forecast the traffic patterns based on the regularity components.
- Shows geographical distribution of the regularity and randomness components.
- Data collected in Shanghai in August 2014, anonymized, and obtained from one of the largest mobile operators.
- Data provides information about mobile traffic consumption by all mobile users, and each entry corresponds to a continuous data communication.

Motivating Examples: Smart X(X=Homes, Cities, Building, Grid)

Homes

- Security and surveillance
- Elderly monitoring
- Energy efficiency
- Ambient assisted living (AAL)
- Human activity recognition

Grid:

- Power load balancing
- Failure recovery
- Distribution management

Building:

- Energy efficiency
- HVAC

**Motivating
Examples:
crowdsensing**

Disaster mapping and
response

Traffic control

Political activity monitoring
and trouble shooting

MBD/IOT

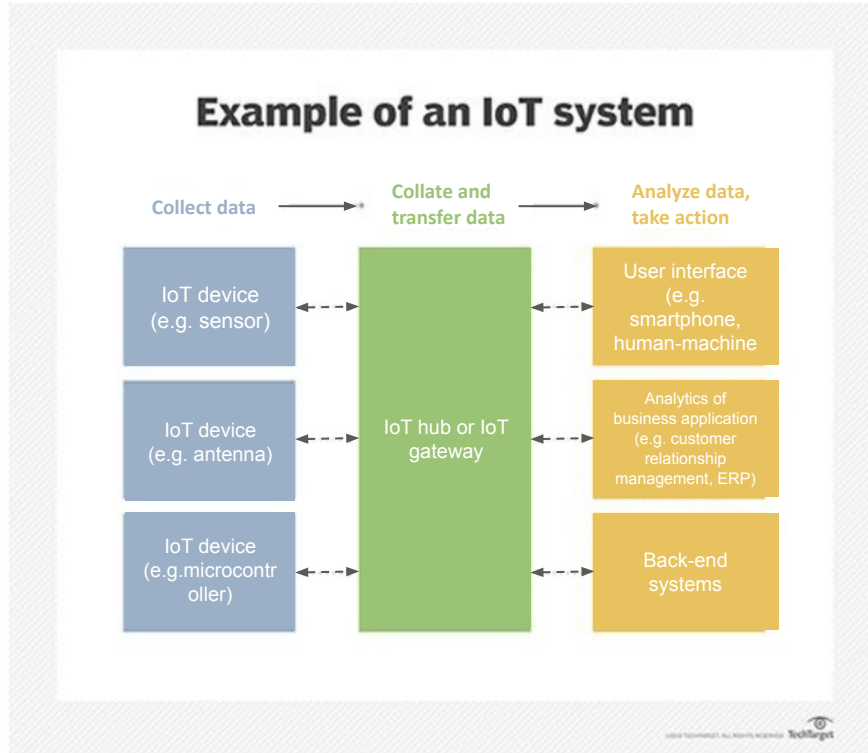
Agenda

- IOT Data
- Big Data vs Mobile Big Data(MBD)
- Characteristics of MBD
- Applications of MBD
- Mobile Big Data Analytics
- Summary

Internet of Things (IoT): Definition

- “A global infrastructure for the information society, enabling advance services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.” - -
<https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx>
- Data is exchanged between systems and devices using **the Internet or other communications networks**.

How IoT Works



- Collect data with IoT devices.
 - IoT devices associated with things (virtual or physical).
- Collate and transfer data over IoT hubs or gateways.
 - Combines data from (most likely) heterogeneous devices.
 - Transfer data over communication networks.
- Analyze data for decision making and action.
 - Support data management, data fusion, and data analysis.
 - Build interfaces for end users, e.g., mobile apps, web apps, etc.

IoT Data

- IoT generates large amounts of data from multiple components.
- Data should be analysed to enable action/decision making.
- Data management and data mining are key ***technical*** and ***managerial*** challenges in IoT development.
- Intrinsic properties of IoT data contribute to the two challenges above.

Properties of IoT Data

- Categorized into data generation, data quality, and data interoperability
- **Data generation properties:**
 - Velocity- generated at different rates
 - Scalability- large scale
 - Dynamics- changing location & environments, intermittent connections
 - Heterogeneity- different data generators, different data formats

Properties of IoT Data (2)

- **Data quality properties:**
 - Incompleteness- need to find data sources to address incompleteness
 - Semantics- need to inject semantics

Properties of IoT Data (3)

- Data interoperability properties:
 - Uncertainty- originating from different sources
 - Redundancy- multiple measures of same thing/metric
 - Ambiguity- means different things

Data types

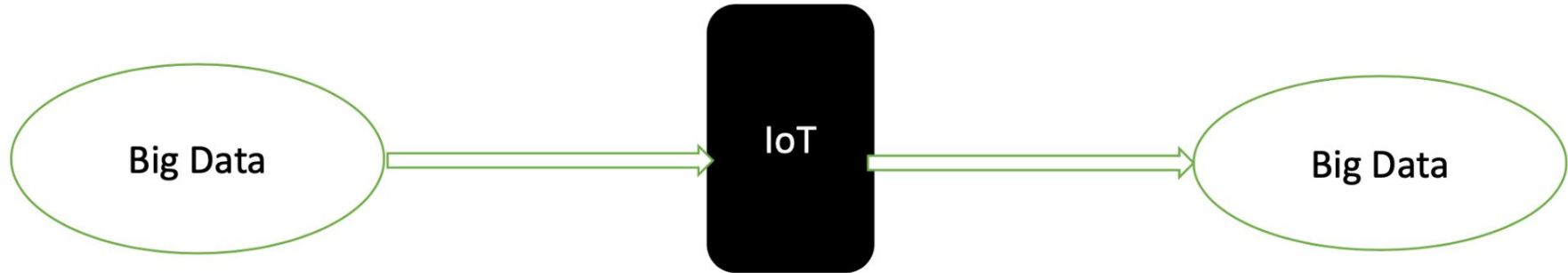
- Textual (un/semi structured)
- Time series
- Geospatial
- Numerical
- Categorical
- Multimodal (image/video/audio)

Sources of IoT Data

- Sensors and actuators
- Databases
- Users/crowd
- Web

Data types

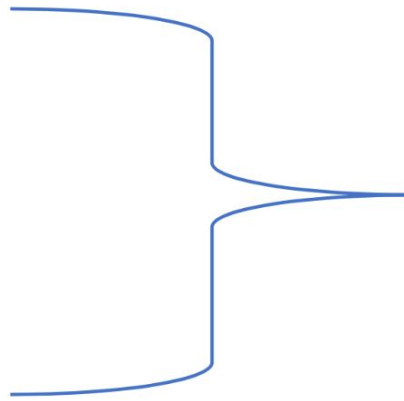
- Big data: data that is too big (volume), too fast (velocity), and too diverse (variety)
- Other characteristics: veracity, variability.
- Data can be *streaming* or *historical*.
- IoT is both a *source* and *sink* of Big Data



Big Data

- Convergence of:
 - Internet technologies,
 - Mobile Computing,
 - Cloud Computing,
 - Big Data,
 - Data Analytics, and
 - IoT
- There are challenges that are peculiar to IoT Big Data Analytics

Characteristics of Classical/Traditional Big Data

- Volume
 - Variety
 - Velocity
 - Veracity
 - Variability
- 

MBD has additional characteristics in addition to the 5Vs.

Still, the characteristics manifest differently.

Veracity

Veracity refers to the *reliability* and *trustworthiness* of data

- **Data quality and accuracy:** How correct, complete, and precise the information is
- **Uncertainty and inconsistency:** Addressing the inherent messiness in data sources
- **Trustworthiness of sources:** Evaluating how reliable the data origin points are
- **Biases and abnormalities:** Identifying skewed representations and outliers

IoT Big Data vs Big Data

	Big Data	IoT Big Data
Volume	Volume stems from large data warehouses and numerous data sources	Volume stems from <i>numerous sensors and internet-connected devices</i>
Velocity	In many cases, velocity is not a concern - tools like MapReduce work just fine.	IoT streams have <i>very high ingestion rates requiring specialized streaming engines</i> . MapReduce unsuitable.
Variety	Variety is a as a result of the need to consolidate data sources of different types	IoT applications have to <i>deal with heterogeneous sensor types and vendors</i> .
Uncertainty /variability	Uncertainty arises from processing of multiple data sources .	Uncertainty arises from <i>both signal processing and noisy nature</i> of IoT data.

MBD Characteristics: Multi-sensory

- Data comes from **multiple sensors** leading to high-dimensional data, e.g.:
 - Sensors embedded to smartphones e.g., accelerometers, gyroscopes, compasses, GPS, ambient light sensors, cameras, microphones, etc.
- **Challenges**: based on general issues with sensor data e.g., sensor or communication failure
 - Incomplete datasets
 - Missing data
 - Outliers
- **Opportunity**: to build various types of applications
 - Health monitoring, context-aware services, action/event/activity/scenario recognition.

MBD Characteristics: Multi-dimensional

- Data has **spatial and temporal dimensions** in addition to **other dimensions**.
- Different sensors may measure and generate **multiple features describing the same thing**.
- For example, **location information** can be obtained **from GPS** or be derived from **base station location** extracted from call data records.

MBD Characteristics: Personalized

- Data contains some form of **user identity**.
 - The user identity could be **explicit** or could be **inferred**.
- The data is **highly personalized** and **relevant to the users' location and overall context**.
 - Makes it possible to provide **personalized services**, e.g., smart context-aware personal services (e.g., recommendations on places to visit in a museum, aisles to check in a supermarket, etc.)

MBD Characteristics: Real-time

- Users expect to receive location-based and highly personalized information and services **in near real-time**.
 - Users expect the information and services to be delivered in a **timely fashion** when such information and services are **still relevant (and useful); not late, irrelevant, or expired**.
- **Challenge:** Mobile devices have limited computation and storage capabilities.
 - **Mobile cloud computing** comes in to address the storage and computation needs (and share computation and storage with the mobile devices).

MBD Characteristics: Spatio-temporal

- Data contains both inherent **location** and **time/temporal** information.
- Useful for context-aware services.
- But context-awareness is complex, e.g.,
 - What formalisms should be used to represent and reason with time?
 - How should we encode temporal semantics e.g., parallel/concurrent events, sequential events, etc.?
 - What is the spatial resolution that is most accurate but less computationally expensive?
 - How informative are the spatial features?

IoT Analytics aka MBD Analytics

- **Definition**: *“The analysis of data from multiple IoT data sources, including sensors, actuators, smart devices and other internet connected objects.”*
- Aims to make *IoT smarter* by “adding value to integrated data and context from the IoT, and producing higher value insights”

Issues with IoT Analytics (1/2)

- **Huge amounts of data to analyse**
- **Heterogeneous IoT data streams:**
 - IoT data streams are **multi-modal** and of **heterogeneous formats, semantics** and **velocities**.
 - May need *semantic interoperability* and multiple means(drivers, connectors, middleware, etc.) of accessing the IoT data sources.
- **Varying data quality:**
 - Due to noisy and incomplete IoT data streams.
 - Problems with devices e.g. power/battery related, faulty readings, software bugs, hardware failures
 - May necessitate statistical and probabilistic techniques in analysis.
 - Also, how do you account for varying reliability in different IoT data streams?

Issues with IoT Analytics (2/2)

- *Real-time nature of IoT data streams:*
 - Often, there is need for real-time processing.
 - Strategies for handling high velocity data needed
- *Time and location dependencies of IoT data streams:*
 - Requires context-aware computation for value to be realized.
- *Difficulty in extracting complex knowledge*

PROJECT

Problem Statement 1

- You have just been hired by the World Bank to monitor **50 million** energy meters in Sub-Saharan Africa.
- Each energy meter sends consumption data (*I, V, Hz, kW, kWh, timestamp*) every **15 minutes**.
- Your first task is to **capture** and **save** these records for efficient retrieval and analysis.

Challenges? Issues?

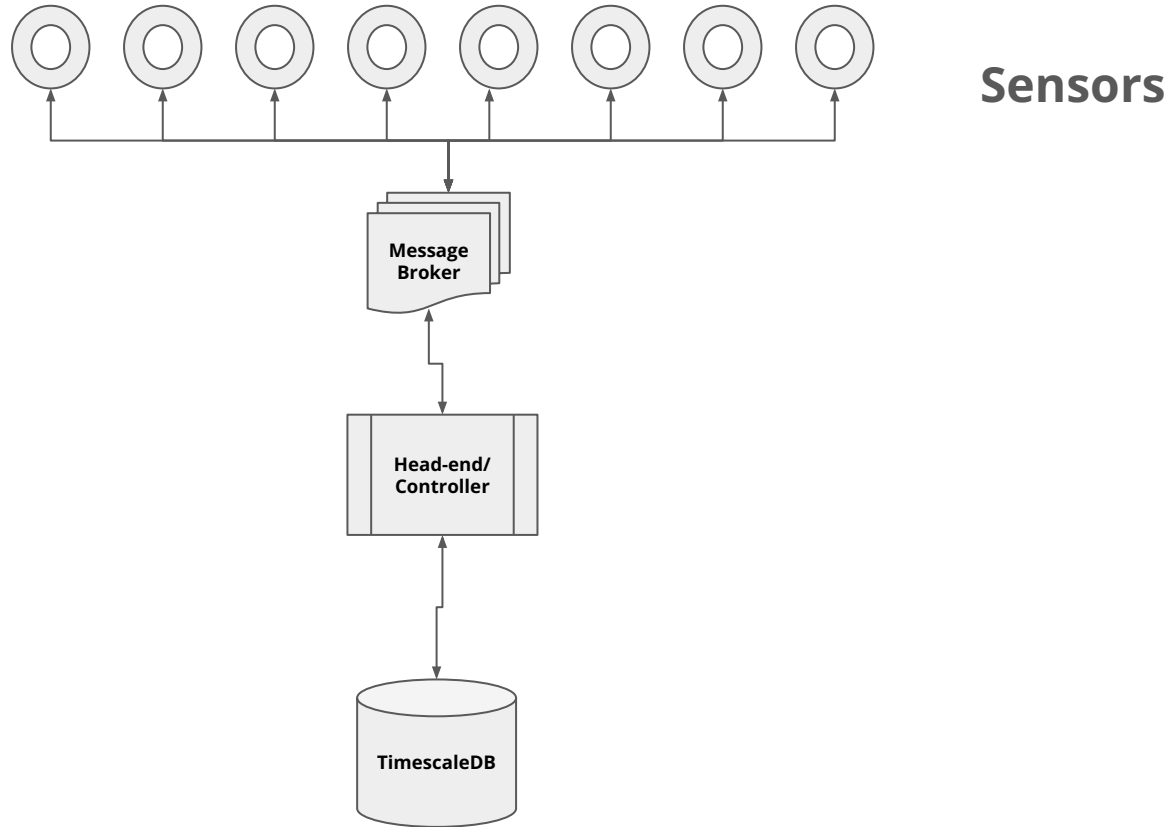
Five points for each.

Let's go!

Technologies & Tools

- Protocol: Message Queuing Telemetry Transport (MQTT)
- MQTT Comm API: Paho
- MQTT Broker: EMQX (download at <https://docs.emqx.com/en/emqx/latest/deploy/install-ubuntu-ce.html>)
- Controller
- Timeseries Database: TimescaleDB

System Architecture



Device Layer

- IoT Devices (sensors, actuators, gateways)
 - Communicate with the MQTT Broker using the Paho MQTT Comm API
 - Publish telemetry data to MQTT topics
 - Subscribe to control commands from the Controller

MQTT Broker

- EMQX
 - Receives and forwards MQTT messages between devices and the Controller
 - Handles device connections, authentication, and topic subscriptions

Headend System

- Server-side application
 - Subscribes to MQTT topics for device telemetry data
 - Processes and analyzes device data
 - Sends control commands to devices via MQTT
 - Stores time-series data in TimescaleDB

Timeseries Database

- TimescaleDB
 - Stores and manages large amounts of time-series data from IoT devices
 - Optimized for efficient storage and querying of time-series data

MQTT Crash Course 1/4

- **Protocol Basics:**

- MQTT is a lightweight, publish-subscribe network protocol that transports messages between devices.
- It is designed for connections with remote locations where a "small code footprint" is required or network bandwidth is limited

- **Publish-Subscribe Model:**

- Unlike traditional client-server models, MQTT uses a broker-based publish-subscribe pattern.
- In this model, clients (publishers) do not send messages directly to other clients (subscribers). Instead, they publish messages to a broker, which then distributes these messages to interested subscribers based on the topic of the messages.

MQTT Crash Course 2/4

- **Quality of Service Levels:**

- MQTT supports three levels of quality of service (QoS) to deliver messages:
 - **QoS 0:** At most once delivery (fire-and-forget).
 - **QoS 1:** At least once delivery (ensures the message is delivered at least once).
 - **QoS 2:** Exactly once delivery (ensures the message is delivered one time only).
- These levels allow for message delivery guarantees according to the requirements of different applications.

MQTT Crash Course 3/4

- **Topics and Wildcards:**

- MQTT uses topics to filter messages for each connected client. Clients subscribe to a topic or topics, and messages are sent to clients based on their subscriptions.
- MQTT also supports wildcards in topic subscription, allowing for greater flexibility in message delivery to subscribers.

- **Security Features:**

- Although MQTT itself does not provide intrinsic security features, it supports secure transmission via SSL/TLS.
- Additional security measures such as user name/password authentication and access control can be implemented at the broker level.

MQTT Crash Course 4/4

- **Use Cases:**

- Ideal for IoT applications, telemetry in low-bandwidth scenarios, and any application where minimal network overhead and low power consumption are required.
- Commonly used in real-time analytics, monitoring of remote sensors, controlling devices over networks, and various M2M (machine-to-machine) contexts.

- **Last Will and Testament:**

- A unique feature where a "last will" message is defined in case of an unexpected disconnection of the client. This message is sent by the broker to notify other clients about the disconnection.

Broker Setup Information (Use yours)

- **Broker Url:** tcp://3.138.185.79 :1883
- **QoS:** 2
- **Clean Session:** true
- **Username:** auca
- **Password:** gishushu

Setup Python Env: Install Paho

- `pip install paho-mqtt==1.6.1`

Python Code: Publish 1/3

```
import paho.mqtt.client as mqtt

# MQTT settings

broker_url = "3.138.185.79"

broker_port = 1883

username = "auca"

password = "gishushu"

topic = "auca_class"

client_id = "my_mqtt_client"


# Callback when the client receives a CONNACK response from the server

def on_connect(client, userdata, flags, rc):

    if rc == 0:

        print("Connected successfully to broker")

    else:

        print(f"Failed to connect, return code {rc}\n")

        # If the client fails to connect then we should stop the loop

        client.loop_stop()
```

Python Code: Publish 2/3

```
# Create a new instance of the MQTT client with a specific client ID

client = mqtt.Client(client_id, clean_session=True)

client.on_connect = on_connect # attach the callback function to the client

client.username_pw_set(username, password) # set username and password


client.connect(broker_url, broker_port, 60) # connect to the broker


# Start the network loop in a separate thread

client.loop_start()


try:

    while True:

        message = input("Enter message to publish or type 'exit' to quit: ")

        if message.lower() == 'exit':

            break

        client.publish(topic, message, qos=2)

except KeyboardInterrupt:

    print("Program interrupted by user, exiting...")
```

Python Code: Publish 3/3

Stop the network loop and disconnect

client.loop_stop()

client.disconnect()

Python Code: Subscribe 1/2

```
import paho.mqtt.client as mqtt

# MQTT settings

broker_url = "3.138.185.79"

broker_port = 1883

username = "auca"

password = "gishushu"

topic = "auca_class"

client_id = "my_mqtt_client_subscriber"


# Callback when the client receives a CONNACK response from the server

def on_connect(client, userdata, flags, rc):

    if rc == 0:

        print("Connected successfully to broker")

        # Subscribe to the topic once connected

        client.subscribe(topic, qos=2)

    else:

        print(f"Failed to connect, return code {rc}\n")
```

Python Code: Subscribe 1/2

Callback for when a PUBLISH message is received from the server

def on_message(client, userdata, msg):

print(f"Message received on topic {msg.topic}: {msg.payload.decode()}")

Create a new instance of the MQTT client with a specific client ID

client = mqtt.Client(client_id, clean_session=True)

client.on_connect = on_connect # attach the connection callback function to the client

client.on_message = on_message # attach the message callback function to the client

client.username_pw_set(username, password) # set username and password

client.connect(broker_url, broker_port, 60) # connect to the broker

Start the network loop in a separate thread

client.loop_forever()

Hypertables

- **Hypertables** are PostgreSQL tables with special features that make it easy to handle time-series data. Anything you can do with regular PostgreSQL tables, you can do with hypertables. In addition, you get the benefits of improved performance and user experience for time-series data.
- They automatically partition your data by time.
- In Timescale, hypertables exist alongside regular PostgreSQL tables. ***Use hypertables to store time-series data***. This gives you improved insert and query performance, and access to useful time-series features. Use regular PostgreSQL tables for other relational data.

Hypertable Partitioning

- When you create and use a hypertable, it automatically partitions data by time, and optionally by space.
- Each hypertable is made up of child tables called chunks.
- Each chunk is assigned a range of time, and only contains data from that range. If the hypertable is also partitioned by space, each chunk is also assigned a subset of the space values.

Time Partitioning

- Each chunk of a hypertable only holds data from a specific time range.
- When you insert data from a time range that doesn't yet have a chunk, Timescale automatically creates a chunk to store it.

Hypertables

chunk_time_interval = "1 day"

Normal table

time	value
2021-01-02 00:00:00	36
2021-01-02 06:00:00	5
2021-01-02 23:00:00	29
2021-01-03 00:00:00	17
2021-01-03 06:00:00	8
2021-01-03 23:00:00	6
2021-01-04 00:00:00	41
2021-01-04 06:00:00	14
2021-01-04 23:00:00	5

Hypertable

time	value
Chunk ID 1	
2021-01-02 00:00:00	36
2021-01-02 06:00:00	5
2021-01-02 23:00:00	29
Chunk ID 2	
2021-01-03 00:00:00	17
2021-01-03 06:00:00	8
2021-01-03 23:00:00	6
Chunk ID 3	
2021-01-04 00:00:00	41
2021-01-04 06:00:00	14
2021-01-04 23:00:00	5

Create a Hypertable 1/2

- To create a hypertable, you need to create a standard PostgreSQL table, and then convert it into a hypertable.
- Create a standard PostgreSQL table.

```
CREATE TABLE sensor (  
    time    TIMESTAMPTZ    NOT NULL,  
    location TEXT          NOT NULL,  
    device  TEXT           NOT NULL,  
    voltage DOUBLE PRECISION NOT NULL,  
    current DOUBLE PRECISION NOT NULL,  
    frequency DOUBLE PRECISION NOT NULL,  
    power   DOUBLE PRECISION NOT NULL,  
    energy  DOUBLE PRECISION NOT NULL  
);
```

Create a Hypertable 2/2

- Convert the table to a hypertable. Specify the name of the table you want to convert, and the column that holds its time values.

```
SELECT create_hypertable('sensor', 'time', chunk_time_interval => interval '1 week');
```

- Some possible chunk intervals
 - '1 hour'
 - '1 day'
 - '1 month'
 - '1 year'

Hypertable Auto Compression

- Enable Compression

```
ALTER TABLE sensor SET (timescaledb.compress, timescaledb.compress_orderby =  
'time');
```

- Add compression Policy

```
SELECT add_compression_policy('sensor', INTERVAL '7 days');
```

Hypertable Manual Compression

- Find Chunks (*all chunks older than x minutes*)

```
SELECT show_chunks('sensor', older_than => INTERVAL '3 minutes');
```

- Compression all chunks older than 3 minutes

```
SELECT compress_chunk(i)
```

```
FROM show_chunks('sensor', older_than => INTERVAL '3 minutes') AS i;
```

Timescale: Essential Commands 1/2

- Disk Size of a hypertable; both compressed and uncompressed chunks
`SELECT hypertable_size('sensor');`
- Retrieves the name and size of each hypertable present in the database
`SELECT sensor, hypertable_size(format('%I.%I', hypertable_schema, hypertable_name)::regclass) FROM timescaledb_information.hypertables;`
- Detailed view of the disk space usage of a hypertable; If running on a distributed hypertable, ordering by `node_name` would show the size distribution across different data nodes
`SELECT * FROM hypertable_detailed_size('sensor') ORDER BY node_name;`

Timescale: Essential Commands 2/2

- Manually compresses a specific chunk identified by its internal name.

```
SELECT compress_chunk( '_timescaledb_internal._hyper_1_1_chunk');
```

- Provides compression statistics for all chunks of the specified hypertable.

```
SELECT * FROM chunk_compression_stats('sensor');
```

- Attempts to compress chunks of the 'sensor' hypertable that were created between **three weeks** ago and **one week** ago.

```
SELECT compress_chunk(i) from show_chunks('sensor', now() - interval '1 week', now() - interval '3 weeks') i;
```

TimescaleDB Installation

<https://docs.timescale.com/self-hosted/latest/install/>

With Docker:

<https://docs.timescale.com/self-hosted/latest/install/installation-docker/>