ΒΚ 1.8) Επεξεργασία ροών δεδομένων σε πραγματικό χρόνο με χρήση των open source κατανεμημένων συστημάτων RabbitMQ ή (multiple MosquitoMQTT), Apache Flink, OpenTSDB και Grafana+websockets.

Messaging Systems-Broker: MosquitoMQTT - RabbitMQ [1] [2]

Live Streaming Processing System: Apache Flink [3]

Timeseries Database: OpenTSDB [4]

Presentation Layer: Grafana [5]

Στην συγκεκριμένη εργασία καλείστε να χρησιμοποιήσετε συνδυαστικά open source εργαλεία από τις παρακάτω ομάδες εργαλείων για την δημιουργία ενός live streaming συστήματος που θα είναι το prototype ενός πραγματικού IoT συστήματος. Συγκεκριμένα, θα δημιουργηθεί ένα σύστημα στο οποίο θα στέλνουν εικονικά δεδομένα οι αισθητήρες σε πραγματικό χρόνο στον MQTT broker (είτε το RabbitMQ ή πολλαπλά mosquitoMQTT) και στην συνέχεια θα μετατρέπονται τα δεδομένα σε ημερήσια ή άλλου είδους μετατροπές με την χρήση του Apache Flink και θα αποθηκεύονται σε μία Timeseries Βάση δεδομένων, την OpenTSDB. Τέλος, τα δεδομένα θα παρουσιάζονται σε Dashboards με την χρήση του open source εργαλείου Grafana και με την χρήση websockets θα παρουσιάζονται live.

[1] https://mosquitto.org/

[2] https://www.rabbitmq.com/
[3] https://flink.apache.org/
[4] http://opentsdb.net/
[5] https://grafana.com/

Οδηγίες για εργασίες

IoT Live Streaming

Συγκεκριμένα, θα δημιουργηθεί ένα σύστημα στο οποίο:

- **Device layer:** Θα δημιουργηθούν εικονικά δεδομένα σε πραγματικό χρόνο.
- **Messaging Broker Layer:** Θα αποσταλούν σε έναν Message Broker.
- **Live Streaming Layer:** Θα μετατρέπονται σε ημερήσια ή άλλου είδους μετατροπές με την χρήση frameworks επεξεργασίας δεδομένων σε πραγματικό χρόνο.
- **Data/Storage Layer:** Θα αποθηκεύονται σε μία NoSQL/Timeseries/In Memory Βάση δεδομένων.
- **Presentation Layer:** Θα παρουσιάζονται σε Dashboards με την χρήση open source εργαλείων.

Device layer: Δημιουργία εικονικών δεδομένων: (10 αισθητήρες)

- Δεδομένα
    - 2 αισθητήρες θερμοκρασίας: TH1, TH2
        - aggregation method: Average
        - interval : 15min
        - data value range : (12-35 ^{o}C), (12-35 ^{o}C)
    - 2 αισθητήρες μέτρησης ενέργειας κλιματιστικών: HVAC1, HVAC2
        - aggregation method: Sum
        - interval : 15min
        - data value range : (0-100 Wh), (0-200 Wh)
    - 2 αισθητήρες μέτρησης ενέργειας λοιπών ηλεκτρικών συσκευών: MiAC1, MiAC2
        - aggregation method: Sum
        - interval : 15min: (0-150 Wh), (0-200 Wh)
    - 1 αισθητήρα μέτρησης αθροιστικής ενέργειας: Etot
        - aggregation method: Max
        - interval : 1 day (end of day)
        - Κάθε μέρα αύξηση ενέργειας κατά (2600x24 ± 1000 Wh)
    - 1 αισθητήρα ανίχνευσης κίνησης: Mov1
        - aggregation method: (Sum ή Count)
        - interval : δεν έχει (1)
    - 1 αισθητήρα μέτρησης κατανάλωσης νερού: W1
        - aggregation method: Sum
        - interval : 15min
        - data value range: (0-1 lt)
    - 1 αισθητήρα μέτρησης αθροιστικής κατανάλωσης νερού: Wtot
        - aggregation method: Max
        - interval : 1 day (end of day)
        - κάθε μέρα αύξηση κατανάλωσης κατά 110 ± 10 lt

Ενδεικτικά θα πρέπει να παραχθούν ενεργειακά δεδομένα στην εξής μορφή:

TH1 :				
2020-01-01 00:15 | 12.4	2020-01-01 00:30 | 12.6	…	2020-01-01 23:45 | 12.1	
TH2 :				
2020-01-01 00:15 | 22.4	2020-01-01 00:30 | 22.6	…	2020-01-01 23:45 | 22.1	
HVAC1 :				
2020-01-01 00:15 | 0.4	2020-01-01 00:30 | 50.6	…	2020-01-01 23:45 | 0.1	
HVAC2 :				
2020-01-01 00:15 | 100.8	2020-01-01 00:30 | 100.6	…	2020-01-01 23:45 | 0.1	
Etot :				
2020-01-01 00:00 | 1500.8	2020-01-02 00:00 | 1500.8+2600x24+500	….		

Αντίστοιχα θα πρέπει να παραχθούν δεδομένα για κατανάλωση νερού. Για την κίνηση θα έχουμε τυχαία δεδομένα μέσα στην ημέρα της μορφής (να παραχθούν τουλάχιστον 4 με 5 άσοι την ημέρα τυχαία):

Mov1 :			
2020-01-01 13:12 |1	2020-01-01 13:15 |1	2020-01-01 21:01 |1	2020-01-01 23:15 |1

- Τρόπος δημιουργίας τους:

  Python, Java, php , nodeJS, C, Scala ή ότι άλλη γλώσσα προγραμματισμού προτιμάτε.

  Παράδειγμα:

  1. Μία function σε python που:
     a. θα παράγει δεδομένα ανά 1 δευτερόλεπτο για κάθε αισθητήρα 15λεπτου
        i. πχ TH1, 2020-01-01 00:15, 12.4
     b. κάθε φορά θα παράγει δεδομένα για τους αισθητήρες των δεκαπενταλέπτων θα φροντίζουμε η τιμές να είναι μέσα στο επιθυμητό range.
     c. Θα παράγει δεδομένα ανά 96 (24x4x1) δευτερόλεπτα για κάθε αισθητήρα ημέρας
     d. κάθε φορά θα παράγει δεδομένα για τους αισθητήρες των ημερών που θα είναι αθροιστικές (να αυξάνονται με βάση το επιθυμητό range).
        i. Etot, 2020-01-01 00:00, 1500.8
           Etot, 2020-01-02 00:00,1500.8+2600x24+500
     e. Θα παράγει 4 με 5 άσσους μέσα στην ημέρα (στα 96 δευτερόλεπτα) τυχαία.

Προσοχή! Θα πρέπει τα timestamps να είναι αυξανόμενα και όχι να πηγαίνουμε πίσω στον χρόνο, σαν να είχατε αισθητήρες σε ένα κτίριο ή στο σπίτι σας.

- Ετεροχρονισμένα δεδομένα

Μόνο για τον αισθητήρα κατανάλωσης νερού W1 θα δημιουργήσετε ένα επιπλέον script-function που θα παράγει δεδομένα ανά 20 δευτερόλεπτα με timestamp που θα αντιστοιχεί σε 2 μέρες πίσω και ανά 120 δευτερόλεπτα timestamp που θα αντιστοιχεί σε 10 μέρες πίσω.

Πχ. Αν το script ξεκίνησε με ημερομηνία 2020-01-04 00:00 έχει τρέξει για 20 δευτερόλεπτα και είναι να δημιουργήσει δεδομένα για το 15λεπτο 2020-01-04 05:00 θα δημιουργήσει επιπλέον ένα

δεδομένο με timestamp 2020-01-02 05:00 και αντίστοιχα όταν περάσουν 120 δευτερόλεπτα 10 μέρες πίσω.

Σκοπός είναι κατά την δημιουργία των aggregations στο **Live Streaming Layer** να δημιουργηθούν τα κατάλληλα φίλτρα για τον προσδιορισμό των δεδομένων που είναι 10 μέρες πίσω ως late rejected events και των δεδομένων που είναι 2 μέρες πίσω ως late processed events.

Πως τα διαχειριζόμαστε:

Τα 2 μέρες πίσω δεδομένα τα λαμβάνουμε υπόψη στα daily aggregations (εδώ μπορείτε να δείτε τις παραμέτρους που έχουν τα window aggregations) και υπολογίζονται όλα σωστά.

Τα 10 μέρες πίσω φιλτράρονται και αποστέλλονται σε διαφορετικό stream (πχ topic στον kafka) και γράφονται και στην βάση σε διαφορετικό σημείο. Επιπλέον τα δεδομένα αυτά απεικονίζονται και στο Grafana στην συνέχεια στο αντίστοιχο table.

Messaging Broker Layer

Ανάλογα την ομάδα θα χρησιμοποιηθούν διαφορετικοί message brokers για την αποστολή των δεδομένων που παρήχθησαν προηγουμένως.

Θα χρησιμοποιηθούν οι παρακάτω:

Apache Kafka, Mosquito MQTT, RabbitMQ, ActiveMQ

Θα πρέπει σε κάθε περίπτωση να δημιουργήσετε το περιβάλλον και να εγκαταστήσετε τον κάθε broker για να μπορείτε να του στείλετε τα δεδομένα. Θα είναι σημαντικό η επιλογή του τρόπου υλοποίησης να είναι τεκμηριωμένη στην τελική σας εξέταση. Πχ partitions στον kafka , topics στο MQTT ,QoS κα. Θα συζητηθούν και στις διαλέξεις που θα ακολουθήσουν

Live Streaming Layer

Ανάλογα την ομάδα θα χρησιμοποιηθούν διαφορετικά εργαλεία για την live επεξεργασία των δεδομένων που θα αντλούνται αποκλειστικά και μόνο από τους message brokers.

Θα χρησιμοποιηθούν τα παρακάτω:

Apache Flink, KStreams(Kafka), Apache Storm, Apache Spark Structured Streaming

Θα σας ζητηθεί ανά περίπτωση να τραβάτε τα δεδομένα από του message brokers, να τα επεξεργάζεστε στο αντίστοιχο framework και να τα στέλνεται στα επόμενα βήματα της εκάστοτε εργασίας.

Συγκεκριμένα θα σας ζητηθεί να κάνετε τις εξής live πράξεις αξιοποιώντας με τον καλύτερο δυνατό τρόπο που πιστεύετε, την παραμετροποίηση που κάνατε στο message broker:

- AggDay[x] - Aggregations σε day για όλους του αισθητήρες 15λέπτου

- Θα πρέπει να λάβετε υπόψη ότι κάποιοι αισθητήρες παράγουν δεδομένα που πρέπει να αθροιστούν στην μέρα ενώ άλλοι δεδομένα που πρέπει να βρεθεί ο μέσος όρος στην μέρα. Το τελικό timestamp θα πρέπει να είναι το 00:00 της επόμενης ημέρας δλδ (2022-10-20 00:15, 2022-10-20 00:30… 2022-10-20 23:45, 2022-10-21 00:00(επομένης)) να γραφτεί στο 2022-10-21 00:00(επομένης)

- ***Για ευκολία όποιος θέλει να ορίσει αρχή της ημέρας το 00:00 και τέλος το 23:45 είναι δεκτόν, απλά θα πρέπει την ημερήσια τιμή να την γράφετε στην αρχή της ημέρας δλδ στο παραπάνω παράδειγμα στις 2022-10-20 00:00.***

- AggDayDiff[y] - Aggregations διαφοράς στην μέρα για τους αισθητήρες day:

    - Θα πρέπει να βρείτε την διαφορά για κάθε μέρα και να την γράψετε όπως πριν. (2022-10-22 00:00 – 2022-10-21 00:00) να γραφτεί στο 2022-10-22 00:00.

- AggDayRest – Aggregation Διαρροής:

    - AggDayDiff[Etot] - AggDay[HVAC1] - AggDay[HVAC2] - AggDay[MiAC1] - AggDay[MiAC2]

    - AggDayDiff[Wto] – AggDay[W1]

Tip! Να χρησιμοποιήσετε window aggregations όπου είναι δυνατόν

Data/Storage Layer : Βάσεις δεδομένων (Timeseries, Document, Graph, InMemory)

Ανάλογα την ομάδα θα χρησιμοποιηθούν διαφορετικά εργαλεία για την αποθήκευση των δεδομένων που θα γίνεται αποκλειστικά και μόνο με την χρήση των streaming processing frameworks είτε με plugins πάνω σε αυτά (πχ kafka sink connectors).

Tips-Cheats:

Πολλά εργαλεία πχ το Flink παρέχουν sink connectors και σας βοηθάνε να γράφετε τα δεδομένα εύκολα και γρήγορα και είναι συνήθως πού αποτελεσματικά. Τα χρησιμοποιείται αν θέλετε.

Πολλές φορές μπορείτε να βρείτε ανάλογα την εργασία sink connectors από kafka σε βάση. Πως μπορεί να χρησιμοποιηθεί? Θα μπορούσατε να γράφετε τα aggregations στον Kafka και από εκεί να χρησιμοποιείται τον sink kafka connector και να γράφετε στην βάση. (θα πιαστεί σωστό, ακόμα και αν στην εργασία σας δεν έχετε kafka και δεν βρίσκεται άλλο τρόπο να γράψετε τα δεδομένα είναι μία τίμια λύση)

Θα αποθηκεύσετε στην βάση δεδομένων τόσο τα raw data όσο και τα aggregated.

Μπορείτε για τα raw να χρησιμοποιήσετε και άλλες μεθόδους όπως αυτή που αναφέρω παραπάνω μέσω kafka sink connectors!!

Μπορείτε να χρησιμοποιήσετε και άλλα εργαλεία παρόμοια, πχ για την influxdb μπορείτε να χρησιμοποιήσετε το telegraf κ.α

Presentation Layer: Απεικόνιση δεδομένων

Ανάλογα την περίπτωση θα χρησιμοποιηθούν δύο πολύ γνωστά open source εργαλεία Grafana kai Kibana για την παρουσίαση των δεδομένων.

Θα πρέπει σε κάθε περίπτωση να απεικονιστεί ένα chart με κάποια μέτρηση 15λεπτου, 1 chart με κάποια μέτρηση ημέρας, 1 table με τιμές για κάποια μέτρηση 15λεπτου, 1 table με τα late rejected events.

Οτιδήποτε επιπλέον θα σας δώσει επιπλέον βαθμό σε αυτό που έχετε.

Tips

Αν δεν βρίσκεται τρόπο να απεικονίσετε τα δεδομένα στο Grafana με κάποιο έτοιμο plugin μην διστάσετε να χρησιμοποιήσετε το JSON plugin όπου μπορείτε εύκολα να φτιάξετε ένα API και να ζητάτε τα δεδομένα. Δεν χρειάζεται να το κάνετε παραμετρικό αν πάτε σε αυτήν την περίπτωση. Αρκεί να φέρει τα δεδομένα που θέλετε και η μόνη παράμετρος να είναι ο χρόνος που τα ζητάει το Grafana.

Για τα live streaming

Το Grafana έχει από μόνο web sockets και αρκεί να στέλνετε σε συγκεκριμένο path τα δεδομένα που θέλετε. Θα γίνει παρουσίαση και στο μάθημα για το πως γίνεται.

IoT Flow Automation

Θα ακολουθήσουν οδηγίες τις επόμενες ημέρες.

1. IoT Live Streaming

Στις συγκεκριμένες εργασίες καλείστε να χρησιμοποιήσετε συνδυαστικά open source εργαλεία για την δημιουργία ενός live streaming συστήματος που θα είναι το prototype ενός πραγματικού IoT συστήματος.

Συγκεκριμένα, θα δημιουργηθεί ένα σύστημα στο οποίο:

1. **Device layer:** Θα δημιουργηθούν εικονικά δεδομένα σε πραγματικό χρόνο.
2. **Messaging Broker Layer:** Θα αποσταλούν σε έναν Message Broker.
3. **Live Streaming Layer:** Θα μετατρέπονται σε ημερήσια ή άλλου είδους μετατροπές με την χρήση frameworks επεξεργασίας δεδομένων σε πραγματικό χρόνο.
4. **Data/Storage Layer:** Θα αποθηκεύονται σε μία NoSQL/Timeseries/In Memory Βάση δεδομένων.
5. **Presentation Layer:** Θα παρουσιάζονται σε Dashboards με την χρήση open source εργαλείων.

Βήματα:

1. Αποστολή εικονικών δεδομένων στο Message Layer (Python, Java, Scala κ.α)

    a. Παραγωγή δεδομένων σειριακά στο χρόνο (15 λεπτών) 1.0/10
    b. Random παραγωγή δεδομένων μη σειριακά στο χρόνο (1 στα 30 δεδομένα) **Late Events** 0.2/10
    c. Αποστολή δεδομένων στον broker, παραμετροποίηση broker, scalability 1.0/10

    2.2/10

2. Επεξεργασία δεδομένων

    a. Aggregations 3.0/10
    b. **Late events** (1.b) detection και αποθήκευση 0.4/10

    3.4/10

3. Αποθήκευση σε Βάση δεδομένων 2.5/10
    - Αποθήκευση raw data
    - Αποθήκευση aggregated Data
    - Αποθήκευση Late Event Data

    2.5/10

4. Απεικόνιση σε Dashboard

    a. Charts + Tables 2.0/10
    b. Table with late events 0.2/10
    c. Websockets για live απεικόνιση δεδομένων 0.8/10

    3.0/10

11.1/10