ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΜΙΟΥΡΓΙΑ ΚΑΤΑΝΕΜΗΜΕΝΟΥ-ΕΝΟΠΟΙΗΜΕΝΟΥ ΣΥΣΤΗΜΑΤΟΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΒΙΟΜΗΧΑΝΙΚΑ ΠΡΩΤΟΚΟΛΛΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Σκουρτσίδης

Επιβλέπων Καθηγήτρια:

Βασιλική Καντερέ

Επίκουρη καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούνιος 2023

Αυτή η σελίδα είναι σκοπίμως κενή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΜΙΟΥΡΓΙΑ ΚΑΤΑΝΕΜΗΜΕΝΟΥ-ΕΝΟΠΟΙΗΜΕΝΟΥ ΣΥΣΤΗΜΑΤΟΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΒΙΟΜΗΧΑΝΙΚΑ ΠΡΩΤΟΚΟΛΛΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Σκουρτσίδης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 04/07/2023.

Βασιλική Καντερέ	Συμεών Παπαβασιλείου	Δημήτριος Τσουμάκος
Επίκουρη Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αυτή η σελίδα είναι σκοπίμως κενή.

.....

Γεώργιος Σκουρτσίδης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Αυτή η σελίδα είναι σκοπίμως κενή.

Περίληψη

Στην παρούσα διπλωματική, σχεδιάζεται και υλοποιείται ένα κατανεμημένο σύστημα επεξεργασίας δεδομένων. Το σύστημα μπορεί να επεξεργάζεται δεδομένα που προέρχονται από διαφορετικά βιομηχανικά πρωτόκολλα, καθώς και να επεξεργαστεί δεδομένα πολλαπλών πηγών και ανομοιογενών δεδομένων σε ζωντανό χρόνο. Επιπλέον, είναι ανεκτικό σε σφάλματα, έχει τη δυνατότητα κλιμάκωσης, παρέχει εργαλεία οπτικοποίησης των δεδομένων, έχει ενσωματωμένη παρακολούθηση της απόδοσης του και εάν κάτι πάει λάθος μπορεί να ειδοποιήσει έγκαιρα το χρήστη με αποστολή e-mail. Καθώς επεξεργάζεται τα δεδομένα, εάν οι εισερχόμενες τιμές είναι ανησυχητικές, μπορεί να λάβει αυτόματα ενέργειες για την αποτροπή πιθανού ατυχήματος.

Τα δεδομένα παράγονται από προσομοίωση αισθητήρων θερμοκρασίας, πίεσης και ισχύς που υλοποιήθηκε με χρήση Python. Οι τιμές μεταφέρονται με χρήση του πρωτοκόλλου Modbus και της βιβλιοθήκης PLC4X στο σύστημα Kafka. Από εκεί οι τιμές επεξεργάζονται με χρήση του Kafka Streams. Για την αποθήκευση των επεξεργασμένων δεδομένων χρησιμοποιείται η βάση InfluxDB, ενώ οι πιο πρόσφατες τιμές που τοποθετούνται επίσης στη μνήμη στο σύστημα Redis. Η εφαρμογή είναι γραμμένη σε Java με χρήση Spring Boot. Η οπτικοποίηση γίνεται στο Grafana, ενώ η απόδοση παρακολουθείται με χρήση Prometheus, cAdvisor, Node Exporter.

Λέξεις-κλειδιά: κατανεμημένη επεξεργασία, επεξεργασία σε ζωντανό χρόνο, Modbus, plc4x, Apache Kafka, Kafka Streams, InfluxDB, Redis, Grafana, cAdvisor, Node Exporter, Spring Boot

Αυτή η σελίδα είναι σκοπίμως κενή.

Abstract

In this diploma thesis, a distributed processing system is designed and implemented. The system can process data from different industrial protocols, as well as process multi-source and heterogeneous data in real time. In addition, it is fault-tolerant, scalable, provides data visualization tools, has built-in performance monitoring, and can alert the user in a timely manner if something goes wrong, by sending an e-mail. As it processes the data, if the incoming values are alarming, it can automatically take actions to prevent a possible accident.

Data are generated from a simulation of temperature, pressure and power sensors implemented using Python. The values are transferred using Modbus protocol and PLC4X library to the Kafka system. Then, the values are processed using Kafka Streams. Processed data are stored in InfluxDB database, while the most recent values are also placed in memory in Redis system. The application is written in Java using Spring Boot. Visualization is done in Grafana, while performance is monitored using Prometheus, cAdvisor, Node Exporter.

Keywords: distributed processing, real time processing, Modbus, plc4x, Apache Kafka, Kafka Streams, InfluxDB, Redis, Grafana, cAdvisor, Node Exporter, Spring Boot

Πρόλογος – Ευχαριστίες

Ευχαριστώ την επιβλέπων καθηγήτρια, κα Βασιλική Καντερέ που μου εμπιστεύτηκε το θέμα της παρούσας διπλωματικής, δίνοντάς μου τη δυνατότητα να επεκτείνω τις γνώσεις μου. Επιπλέον, ευχαριστώ τον κ. Παρασκευά Κερασιώτη για την καθοδήγησή του.

Ιδιαιτέρως θέλω να ευχαριστήσω τους γονείς μου, τα αδέρφια μου και τους φίλους μου για την στήριξη καθόλη τη διάρκεια των φοιτητικών μου χρόνων. Η πρόοδος μου ως άνθρωπος δε θα μπορούσε να συντελεστεί δίχως αυτούς. Τέλος, ευχαριστώ την κοπέλα μου για την υπομονή της και τη κατανόησή της κατά της διάρκεια της συγγραφής της εργασίας.

Πίνακας περιεχομένων

Ευρετήριο Σχημάτων

Ευρετήριο Πινάκων

1 Εισαγωγή

Στην 4η Βιομηχανική επανάσταση την οποία διανύουμε, περίοπτη θέση κατέχει το Διαδίκτυο των Πραγμάτων (Internet-of-Things, IoT). Το IoT είναι ένας γενικός όρος για τον αυξανόμενο αριθμό ηλεκτρονικών συσκευών που είναι συνδεδεμένες στο διαδίκτυο και στέλνουν δεδομένα, λαμβάνουν οδηγίες ή και τα δύο. Υπάρχει ένα ευρύ φάσμα «πραγμάτων» που εμπίπτουν στην ομπρέλα του IoT. Μερικά παραδείγματα είναι αισθητήρες με δυνατότητα σύνδεσης στο διαδίκτυο που μεταμορφώνουν τα εργοστάσια, την υγειονομική περίθαλψη, τις μεταφορές, τα κέντρα διανομής και τα αγροκτήματα, ακόμη και τον οικιακό χώρο. [1]

Το IoT φέρνει τη συνδεσιμότητα στο διαδίκτυο, την επεξεργασία δεδομένων και την ανάλυση στον κόσμο των φυσικών αντικειμένων. Δισεκατομμύρια ενσωματωμένοι αισθητήρες με δυνατότητα σύνδεσης στο διαδίκτυο παγκοσμίως παρέχουν ένα πλούσιο σύνολο δεδομένων που οι εταιρείες και οι οργανισμοί μπορούν να χρησιμοποιήσουν για να βελτιώσουν την ασφάλεια των λειτουργιών τους, να παρακολουθήσουν τα περιουσιακά στοιχεία και να μειώσουν τις χειροκίνητες διαδικασίες. [2]

Τα δεδομένα από τα μηχανήματα μπορούν να χρησιμοποιηθούν για να προβλέψουν αν ο εξοπλισμός θα χαλάσει, δίνοντας στους κατασκευαστές εκ των προτέρων προειδοποίηση για την αποφυγή μεγάλων χρονικών διαστημάτων διακοπής λειτουργίας. Επιπλέον, συσκευές IoT μπορούν να χρησιμοποιηθούν για τη συλλογή δεδομένων σχετικά με τις προτιμήσεις και τη συμπεριφορά των πελατών, καθώς και για την ανάλυση αυτών. Τα παραπάνω οδηγούν στην βελτίωση των υπηρεσιών ή προϊόντος που παράγεται.

Ο τομέας του IoT αντιμετωπίζει πληθώρα προκλήσεων, συμπεριλαμβανομένων του μεγάλου αριθμού ετερογενών συσκευών, την τυποποίηση των εργαλείων, του λογισμικού, και των αισθητήρων και την έλλειψη ενοποιημένων πρωτοκόλλων επικοινωνίας και προτύπων. Επιπλέον, το IoT αξιοποιεί τεράστιο όγκο δεδομένων που συλλέγονται και συσσωρεύονται μέσω έξυπνων συσκευών. Κάθε δύο περίπου

χρόνια το μέγεθος των δεδομένων διπλασιάζεται στον κόσμο [3], θα πρέπει να αναπτυχθούν τεχνικές που θα μετατρέπουν αυτά τα δεδομένα σε αξιοποιήσιμη γνώση. [2], [4]

Στην παρούσα εργασία θα αναπτύξουμε ένα πλήρες σύστημα IoT. Τα δεδομένα αφορούν δεδομένα από συσκευές PLC, οι οποίες χρησιμοποιούνται σήμερα ευρέως στην βιομηχανία. Ο τρόπος λειτουργίας αυτών των συσκευών θα αναλυθεί σε επόμενο κεφάλαιο. Με χρήση βιομηχανικών πρωτόκολλων επικοινωνίας, τα δεδομένα μεταφέρονται στη δική μας πλατφόρμα, όπου επεξεργάζονται, μετασχηματίζονται, αποθηκεύονται και απεικονίζονται σε ζωντανό χρόνο. Οι τεχνολογίες που χρησιμοποιούνται έχουν σχεδιαστεί προκειμένου να δίνουν λύσεις σε γνωστά προβλήματα των δεδομένων του IoT, όπως είναι η ταχύτητα που πρέπει να επεξεργαστούν τα δεδομένα, η ποικιλία τους, η εγκυρότητα, ο όγκος τους, καθώς και η αξία τους [3].

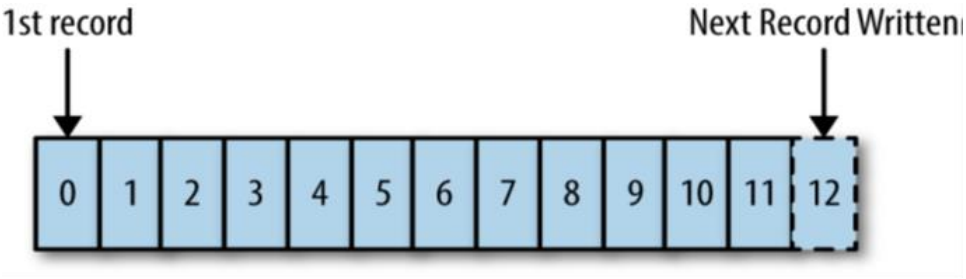
Αρχεία καταγραφής (Log files)

Στην παρούσα εργασία, θα αναφερθούμε σε αρχεία καταγραφής. Είναι απαραίτητο να διασαφηνιστεί ο όρος.

Εδώ δε γίνεται αναφορά στα κλασσικά αρχεία καταγραφής που χρησιμοποιούνται για την αποθήκευση των μηνυμάτων εφαρμογών που τρέχουν, όπως αυτά της *εικόνας 1-1* που παρουσιάζεται παρακάτω.

Εικόνα 1-1: Στιγμιότυπο από αρχείο καταγραφής της εφαρμογής

Ως αρχείο καταγραφής ορίζεται ένα αρχείο στο οποίο μπορούν μόνο να προσαρτώνται δεδομένα στο τέλος του. Τα αρχείο περιέχει μια ακολουθία εγγραφών ταξινομημένων σύμφωνα με το χρόνο, όπως φαίνεται στην *εικόνα 1-2 παρακάτω.*



Εικόνα 1-2: Σημασιολογική απεικόνιση αρχείου καταγραφής. Παρατηρείται πως οι εγγραφές είναι ταξινομημένες, σύμφωνα με το χρόνο εγγραφής τους. Οι νεότερες εγγραφές τοποθετούνται στο τέλος του αρχείου [5]

Κάθε κουτάκι της εικόνας 1-2 αντιπροσωπεύει μια εγγραφή που προσαρτήθηκε στο αρχείο καταγραφής. Οι εγγραφές αποθηκεύονται με τη σειρά με την οποία προσαρτήθηκαν. Η ανάγνωση γίνεται από αριστερά προς τα δεξιά. Κάθε εγγραφή που προσαρτάται στο αρχείο καταγραφής εκχωρείται ένας μοναδικός, διαδοχικός αριθμός καταχώρησης που λειτουργεί ως μοναδικό κλειδί. Το περιεχόμενο και η μορφή των εγγραφών δεν είναι σημαντικά για τους σκοπούς αυτής της συζήτησης. Η ταξινόμηση των εγγραφών ορίζει την έννοια του *χρόνου*, δεδομένου ότι οι εγγραφές στα αριστερά ορίζονται να είναι παλαιότερες από τις εγγραφές στα δεξιά. Ο αριθμός της καταχώρησης μπορεί να θεωρηθεί ως η *χρονοσφραγίδα (timestamp)* της καταχώρησης. Η περιγραφή αυτής της διάταξης ως έννοια του χρόνου έχει τη βολική ιδιότητα να είναι αποσυνδεδεμένη από οποιαδήποτε φυσικό ρολόι. Αυτή η ιδιότητα θα αποδειχθεί απαραίτητη, καθώς στην παρούσα εργασία γίνεται χρήση κατανεμημένων συστημάτων, όπως το Apache Kafka.

2 Βιομηχανικά πρωτόκολλα επικοινωνίας

Εισαγωγή

Τα βιομηχανικά πρωτόκολλα επικοινωνίας είναι ένα σύνολο κανόνων και προτύπων που διευκολύνουν την επικοινωνία και την ανταλλαγή δεδομένων μεταξύ συσκευών, μηχανών και συστημάτων σε ένα βιομηχανικό περιβάλλον. Αυτά τα πρωτόκολλα είναι απαραίτητα για τη διασφάλιση της απρόσκοπτης, αξιόπιστης και αποτελεσματικής επικοινωνίας σε διάφορες βιομηχανικές διαδικασίες, όπως η κατασκευή, ο αυτοματισμός και τα συστήματα ελέγχου. Σε ένα βιομηχανικό περιβάλλον, πληθώρα συσκευών χρησιμοποιούν βιομηχανικά πρωτόκολλα επικοινωνίας για την επικοινωνία και την ανταλλαγή δεδομένων. Οι πιο συχνά χρησιμοποιούμενες σήμερα, είναι οι συσκευές PLC.

Προγραμματιζόμενος Λογικός Ελεγκτής (PLC)

Ένας προγραμματιζόμενος λογικός ελεγκτής (PLC) είναι ένας εξειδικευμένος υπολογιστής που έχει σχεδιαστεί για τον έλεγχο βιομηχανικών διεργασιών και εξοπλισμού αυτοματισμού. Τα PLC χρησιμοποιούνται συνήθως σε εργοστασιακές μονάδες, χημικά εργοστάσια, σταθμούς παραγωγής ενέργειας και άλλες βιομηχανίες που απαιτούν ακριβή έλεγχο του εξοπλισμού και των διαδικασιών. Τα PLC αφορούν τις συσκευές οι οποίες παράγουν τα δεδομένα, στα πλαίσια της τωρινής εργασίας. Για το λόγο αυτό κρίθηκε απαραίτητη μια σύντομη περιγραφή αυτών.

Οι πρώτοι προγραμματιζόμενοι λογικοί ελεγκτές σχεδιάστηκαν και αναπτύχθηκαν από την εταιρεία Modicon ως αντικαταστάτες ρελέ για την GM και τη Landis. Το πρώτο PLC, μοντέλο 084, εφευρέθηκε από τον Dick Morley το 1969. Το πρώτο εμπορικά επιτυχημένο PLC, το 184, παρουσιάστηκε το 1973 και σχεδιάστηκε από τον Michael Greenberg. Τα PLC έλαβαν ευρεία αποδοχή, καθώς είναι εύκολο να προγραμματιστούν και να συντηρηθούν, σε αντίθεση με τα πολύπλοκα ηλεκτρονικά

κυκλώματα που χρησιμοποιούνταν παλαιότερα στη βιομηχανία. Στην εικόνα 2-1 βλέπουμε πως μοιάζει ένα σύγχρονο PLC.

Ένα PLC αποτελείται από τρία κύρια στοιχεία: μονάδες εισόδου/εξόδου (I/O), κεντρική μονάδα επεξεργασίας (ΚΜΑ, CPU) και λογισμικό προγραμματισμού. Οι μονάδες εισόδου/εξόδου συνδέονται με αισθητήρες και ενεργοποιητές (actuators) στον ελεγχόμενο εξοπλισμό, ενώ η ΚΜΑ επεξεργάζεται τα σήματα που λαμβάνονται από τις μονάδες εισόδου/εξόδου και εκτελεί το πρόγραμμα που έχει οριστεί από τον χρήστη και είναι αποθηκευμένο στη μνήμη της. Το πρόγραμμα αυτό είναι αποθηκευμένο σε *μη-πτητική* μνήμη, πράγμα που σημαίνει ότι δε θα χαθεί εάν διακοπεί η τροφοδοσία ρεύματος.

Οι μονάδες εισόδου/εξόδου (I/O) μεταδίδουν τις απαραίτητες πληροφορίες στην CPU και επικοινωνούν την απαιτούμενη εργασία σε έναν συνεχή βρόχο. Οι καταχωρητές (registers) εισόδου και εξόδου μπορεί να είναι είτε ψηφιακής είτε αναλογικής μορφής: οι ψηφιακοί καταχωρητές είναι πεπερασμένες τιμές που αναπαρίστανται σε τιμές 1 ή 0, οι αναλογικοί καταχωρητές μετρούν εύρος ρευμάτων ή τάσεων. Οι είσοδοι είναι διακόπτες (switches), αισθητήρες και έξυπνες συσκευές σε αναλογική ή ψηφιακή μορφή. Οι έξοδοι μπορεί για παράδειγμα να είναι εκκινητές κινητήρων ή φώτων ή να ελέγχουν το άνοιγμα βαλβίδων.

Εικόνα 2-1: Ένας σύγχρονος προγραμματιζόμενος λογικός ελεγκτής (PLC)

Ένα πρόγραμμα για PLC κατασκευάζεται συνήθως με τη χρήση μιας οπτικής γλώσσας προγραμματισμού, η οποία συνήθως είναι η γλώσσα ηλεκτρολογικών γραφικών (ladder logic). Η προαναφερθείσα γλώσσα, αν και η δημοφιλέστερη, δεν είναι η μοναδική για προγραμματισμό σε PLC. Υπάρχουν επίσης οι γλώσσες Function Block Diagram (FBD), Sequential Function Chart, Structured Text και Instruction List, αν και οι δύο τελευταίες δεν είναι οπτικές γλώσσες προγραμματισμού. Οι γλώσσες αυτές επιτρέπουν στους χρήστες να δημιουργούν και να επεξεργάζονται εύκολα το πρόγραμμα χωρίς να χρειάζονται εκτεταμένες γνώσεις προγραμματισμού σε υπολογιστή.

Τα PLC χρησιμοποιούν έναν κύκλο σάρωσης για την επεξεργασία των πληροφοριών και την εκτέλεση του προγράμματος. Κατά τη διάρκεια του κύκλου σάρωσης, η CPU διαβάζει τις εισόδους από τις μονάδες I/O και ενημερώνει τις τιμές στη μνήμη της. Στη συνέχεια εκτελεί το πρόγραμμα με βάση τις τιμές στη μνήμη, το οποίο μπορεί να περιλαμβάνει την εκτέλεση μαθηματικών υπολογισμών, τη λήψη αποφάσεων με βάση τις τιμές αισθητήρων ή τον έλεγχο ενεργοποιητών (actuators).

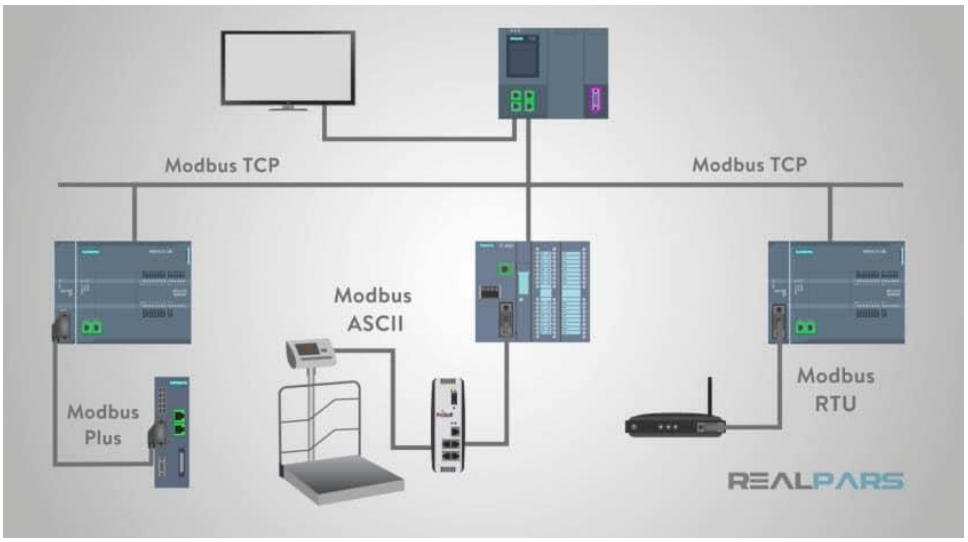
Μόλις εκτελεστεί το πρόγραμμα, η ΚΜΕ ενημερώνει τις τιμές των σημάτων εξόδου στη μνήμη της και τις στέλνει στις κατάλληλες μονάδες Ε/Ε, οι οποίες στη συνέχεια ενεργοποιούν τους αντίστοιχους ενεργοποιητές για τον έλεγχο του ελεγχόμενου εξοπλισμού.

Τα PLC έχουν σχεδιαστεί για να είναι ανθεκτικά και αξιόπιστα, με χαρακτηριστικά όπως η ανίχνευση σφαλμάτων για να εξασφαλίζουν συνεχή λειτουργία. Χρησιμοποιούνται ευρέως στον βιομηχανικό αυτοματισμό λόγω της ικανότητάς τους να παρέχουν ακριβή και αξιόπιστο έλεγχο πολύπλοκου εξοπλισμού. [6]

Modbus

Εισαγωγή

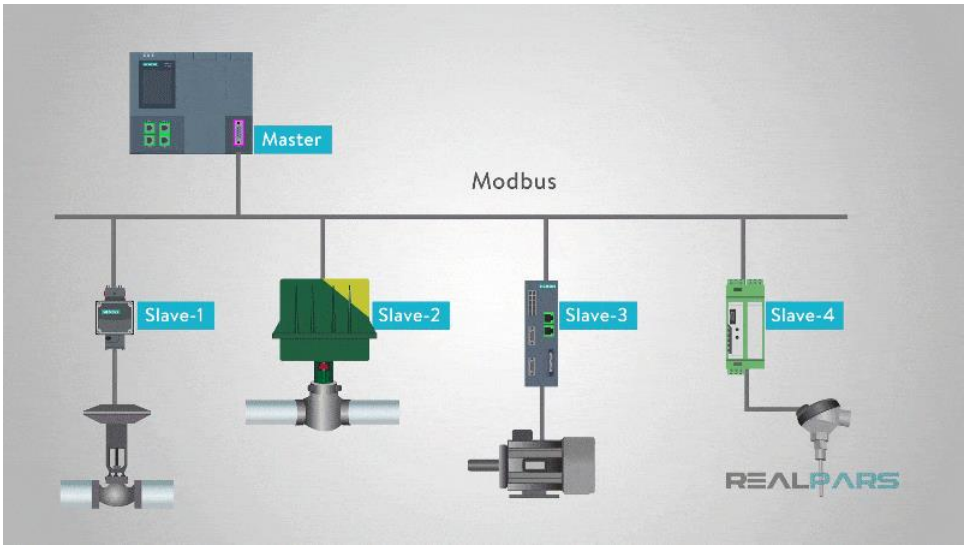
Το Modbus είναι ένα πρωτόκολλο σειριακής επικοινωνίας που αναπτύχθηκε από την εταιρεία Modicon το 1979 για τη διευκόλυνση της επικοινωνίας μεταξύ βιομηχανικών ηλεκτρονικών συσκευών [7]. Πρόκειται για το παλαιότερο και δημοφιλέστερο βιομηχανικό πρωτόκολλο στον τομέα του αυτοματισμού διαδικασιών και του SCADA (εποπτικός έλεγχος και απόκτηση δεδομένων - supervisory control and data acquisition). Την ανάπτυξη και ενημέρωση των πρωτοκόλλων Modbus διαχειρίζεται ο οργανισμός «Modbus». Ο Οργανισμός Modbus είναι μια ένωση χρηστών και προμηθευτών των συσκευών που είναι συμβατές με αυτό. Υπάρχουν διάφορες εκδόσεις του πρωτοκόλλου Modbus . Οι πιο συνηθισμένες είναι το Modbus RTU, το Modbus ASCII, το Modbus TCP και το Modbus Plus, όπως απεικονίζονται στην εικόνα 2-2.



Εικόνα 2-2: Οι πιο συχνά χρησιμοποιούμενες εκδόσεις του πρωτοκόλλου Modbus [8]

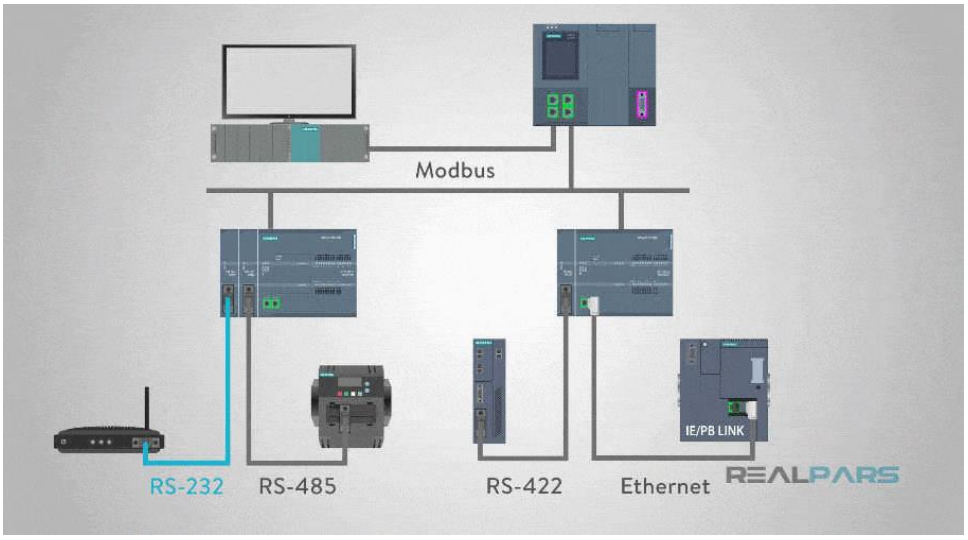
Είναι ένα ανοιχτό πρωτόκολλο, που σημαίνει ότι οι προδιαγραφές δημοσιεύονται και μπορούν να χρησιμοποιηθούν από οποιονδήποτε ελεύθερα ή με άδεια χρήσης, ακόμα για εμπορικούς σκοπούς. [9]

Πρόκειται για πρωτόκολλο master/slave, που σημαίνει ότι μια συσκευή (master) μπορεί να ελέγχει πολλαπλές συσκευές (slaves) μέσω ενός ενιαίου δικτύου (εικόνα 2-3). Θα απευθυνόμαστε από εδώ και στο εξής σε αυτές ως «κύρια» συσκευή (master) και «δευτερεύουσες» συσκευές (slaves).



Εικόνα 2-3: Η αρχιτεκτονική κύριας-δευτερεύσας συσκευής και οι ανομοιογενής πηγές δεδομένων για το Modbus.

Το Modbus χρησιμοποιεί ένα απλό μοντέλο αίτησης/απόκρισης για την ανταλλαγή δεδομένων μεταξύ συσκευών. Είναι εύκολο στην υλοποίηση και υποστηρίζεται από πληθώρα συσκευών. Η επικοινωνία επιτυγχάνεται μέσω διαφόρων τύπων φυσικών μέσων, όπως σειριακά καλώδια RS-232, RS-485, RS-422 και Ethernet (εικόνα 2-4) [10]. Η αρχική διεπαφή Modbus λειτουργούσε με σειριακή επικοινωνία RS-232, αλλά οι περισσότερες από τις μεταγενέστερες υλοποιήσεις Modbus χρησιμοποιούν RS-485, καθώς επιτρέπει την επικοινωνία μεταξύ μεγαλύτερων αποστάσεων, προσφέρει υψηλότερες ταχύτητες μετάδοσης, καθώς και τη δυνατότητα διασύνδεσης πολλών συσκευών σε ένα ενιαίο δίκτυο πολλαπλής διοχέτευσης (multi-drop network).

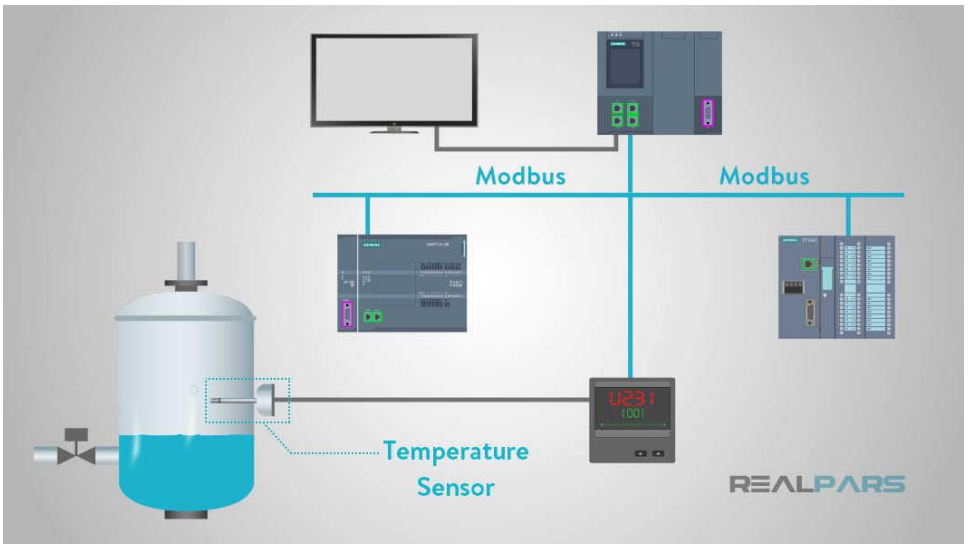


Εικόνα 2-4: Ευρέως διαδεδομένοι τύποι φυσικών μέσων που χρησιμοποιούνται από το πρωτόκολλο Modbus [8]

Το πρωτόκολλο Modbus χρησιμοποιείται σε ένα ευρύ φάσμα εφαρμογών βιομηχανικού αυτοματισμού, όπως η βιομηχανική παραγωγή, ο αυτοματισμός σε εγκαταστάσεις, η HVAC και ο έλεγχος διεργασιών. Χρησιμοποιείται συνήθως για την ανάγνωση και εγγραφή δεδομένων από διάφορους τύπους συσκευών. Ακολουθούν μερικά παραδείγματα χρήσης:

- Σε ένα εργοστάσιο παραγωγής, το σύστημα ελέγχου θα μπορούσε να χρησιμοποιεί το Modbus για να διαβάζει τις τιμές των αισθητήρων θερμοκρασίας, πίεσης και ισχύς και να ελέγχει την ταχύτητα και την κατεύθυνση των κινητήρων και των βαλβίδων.
- Σε ένα σύστημα αυτοματισμού ενός κτιρίου, το σύστημα ελέγχου θα μπορούσε να χρησιμοποιεί το Modbus για να διαβάζει τις τιμές των αισθητήρων θερμοκρασίας, υγρασίας και ποιότητας αέρα και να ελέγχει τη λειτουργία των ανεμιστήρων, των αντλιών και του εξοπλισμού HVAC.
- Σε ένα σύστημα παραγωγής και διανομής ηλεκτρικής ενέργειας, το σύστημα ελέγχου χρησιμοποιείτο Modbus για την ανάγνωση των τιμών των αισθητήρων τάσης, ρεύματος και ισχύος και για τον έλεγχο της λειτουργίας των γεννητριών, των μετασχηματιστών και άλλου ηλεκτρικού εξοπλισμού.
- Σε ένα σύστημα διαχείρισης νερού και λυμάτων, το σύστημα ελέγχου θα μπορούσε να χρησιμοποιεί το Modbus για να διαβάζει τις τιμές των

αισθητήρων pH, ροής και πίεσης και να ελέγχει τη λειτουργία αντλιών, βαλβίδων και λοιπού εξοπλισμού (εικόνα 2-5).



Εικόνα 2-5: Διαχείριση θέρμανσης νερού, με χρήση PLC και του πρωτοκόλλου Modbus [8].

Αποθήκευση δεδομένων

Στο πρωτόκολλο Modbus, τα δεδομένα οργανώνονται σε καταχωρητές και πηνία. Οι καταχωρητές χρησιμοποιούνται για την αποθήκευση αριθμητικών τιμών, όπως θερμοκρασίες, πιέσεις ή άλλους τύπους δεδομένων μέτρησης. Τα πηνία χρησιμοποιούνται για την αποθήκευση δυαδικών τιμών, όπως οι καταστάσεις on/off ή οι ψηφιακές είσοδοι και έξοδοι.

Τα δεδομένα μπορούν να διαβαστούν ή να εγγραφούν σε πηνία (διακριτές έξοδοι), διακριτές εισόδους, καταχωρητές εισόδου και καταχωρητές αποθήκευσης.

Τα πηνία χρησιμοποιούνται για την αναπαράσταση ενός μόνο bit δεδομένων, χρησιμοποιούνται ως διακριτές έξοδοι και μπορούν να χρησιμοποιηθούν για εγγραφή και ανάγνωση δεδομένων. Οι διακριτές είσοδοι είναι, όπως και τα πηνία, καταχωρητές 1-bit με χρήση μόνο ανάγωσης. Στην πράξη, τα πηνία και οι διακριτές είσοδοι χρησιμοποιούνται συχνά για την αναπαράσταση της κατάστασης ψηφιακών εισόδων και εξόδων, όπως η κατάσταση ενεργοποίησης/απενεργοποίησης ενός ρελέ ή η κατάσταση ανοίγματος/κλεισίματος μιας βαλβίδας.

Οι καταχωρητές είναι συνήθως λέξεις των 16 bit, πράγμα που σημαίνει ότι μπορούν να αποθηκεύσουν ακέραιες τιμές που κυμαίνονται από 0 έως 65535. Ορισμένες υλοποιήσεις Modbus υποστηρίζουν επίσης καταχωρητές 32 bit, οι οποίοι μπορούν να αποθηκεύουν μεγαλύτερες ακέραιες τιμές ή τιμές κινητής υποδιαστολής. Οι καταχωρητές εισόδου χρησιμοποιούνται συνήθως για την αναπαράσταση αναλογικών σημάτων εισόδου και έχουν δυνατότητα εγγραφής. Οι καταχωρητές αποθήκευσης έχουν δυνατότητα εγγραφής και ανάγνωσης.

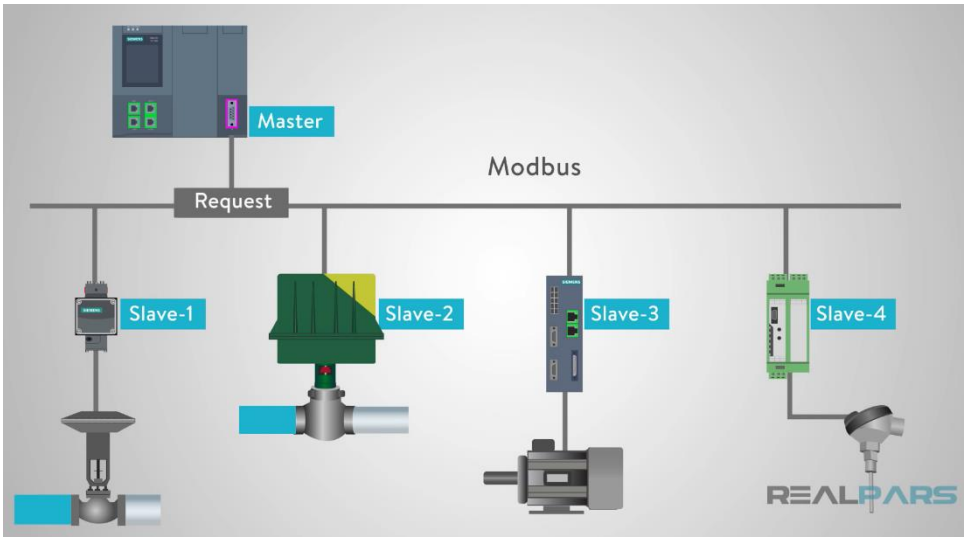
Σημειώνεται πως μια συσκευή μπορεί να μην έχει όλους αυτούς τους τύπους καταχωρητών. Αυτό είναι κάτι που αφορά τη συγκεκριμένη υλοποίηση συσκευής και είναι ευθύνη του κατασκευαστή. Για παράδειγμα, είναι δυνατόν για μια συσκευή να επικοινωνεί με το πρωτόκολλο Modbus, αλλά να χρησιμοποιεί μόνο καταχωρητές αποθήκευσης και καθόλου πηνία, διακριτές εισόδους και καταχωρητές εισόδου.

Η ακριβής σημασία κάθε καταχωρητή και πηνίου εξαρτάται από τη συγκεκριμένη υλοποίηση, αλλά οι βασικές έννοιες είναι ίδιες σε όλα τα συστήματα Modbus.[11]

Επικοινωνία

Το Modbus χρησιμοποιεί δυαδική μορφή για τα μηνύματά του, πράγμα που σημαίνει ότι τα δεδομένα αντιπροσωπεύονται ως μια αλληλουχία μονάδων και μηδενικών. Κάθε bit αποστέλλεται ως τάση. Τα μηδενικά αποστέλλονται ως θετικές τάσεις και οι μονάδες ως αρνητικές. Μια συνηθισμένη ταχύτητα μετάδοσης είναι 9600 baud (bit ανά δευτερόλεπτο).

Η κύρια συσκευή ξεκινά την επικοινωνία με μία δευτερεύουσα στέλνοντας ένα αίτημα (εικόνα 2-6). Στη συνέχεια, η δευτερεύουσα συσκευή απαντά με τα ζητούμενα δεδομένα. Οι δευτερεύουσες συσκευές δεν μπορούν να αρχίσουν την επικοινωνία με την κύρια συσκευή ή με άλλες συσκευές. Μπορούν μόνο να απαντούν στην κύρια συσκευή ή να επιτελούν τη λειτουργία που η κύρια συσκευή τους υποδεικνύει. Η κύρια συσκευή μπορεί να απευθύνεται σε μια συγκεκριμένη δευτερεύουσα ή να στείλει ένα μήνυμα εκπομπής (broadcast message), το οποίο απευθύνεται σε όλες τις δευτερεύουσες συσκευές. Σε περίπτωση μηνύματος εκπομπής, οι δευτερεύουσες συσκευές δεν απαντούν στη μήνυμα.



Εικόνα 2-6: η κύρια συσκευή (master) στέλνει ένα αίτημα (request) σε μια δευτερεύουσα (slave)

Τα αιτήματα στο Modbus αποτελούνται από μια επικεφαλίδα, έναν κωδικό λειτουργίας (function code), ένα πεδίο δεδομένων (data) και το πεδίο CRC, το οποίο θα αναλυθεί παρακάτω. Τα πεδία ενός αιτήματος απεικονίζονται στην εικόνα 2-7.

Η επικεφαλίδα περιέχει τη διεύθυνση της δευτερεύουσας συσκευής. Σε κάθε δευτερεύουσα συσκευή σε ένα δίκτυο εκχωρείται μια μοναδική διεύθυνση, μέγεθος ενός byte, με τιμές από το 1 έως το 247. Όταν η κύρια συσκευή ζητά δεδομένα, το πρώτο byte που στέλνει είναι η διεύθυνση της δευτερεύουσας συσκευής. Με αυτόν τον τρόπο κάθε δευτερεύουσα συσκευή γνωρίζει μετά το πρώτο byte αν πρέπει ή όχι να αγνοήσει το μήνυμα. Σε περίπτωση μηνύματος εκπομπής (broadcast message), το πεδίο της επικεφαλίδας περιέχει μια ειδική διεύθυνση προκειμένου οι δευτερεύουσες συσκευές να αναγνωρίσουν αυτό το ειδικό μήνυμα.

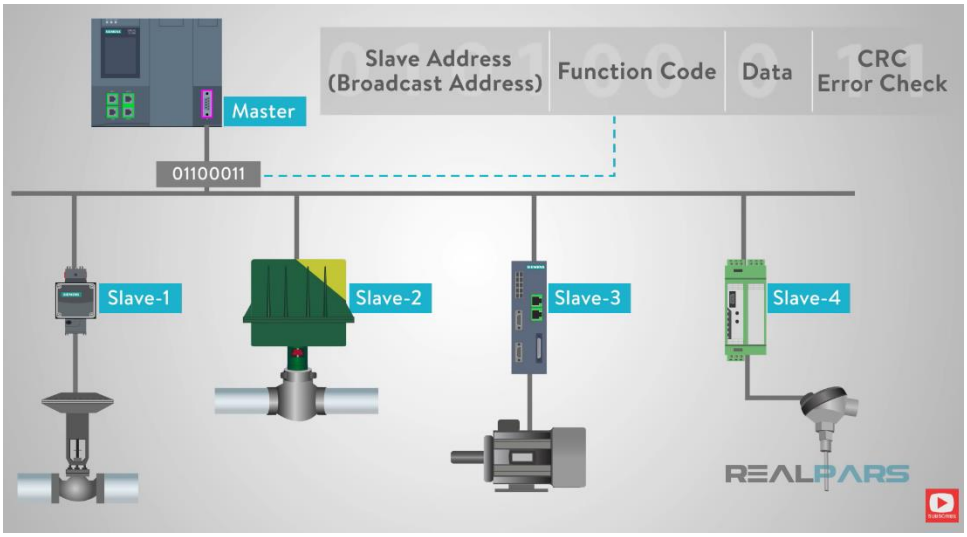
Το δεύτερο byte που στέλνεται είναι ο κωδικός λειτουργίας και καθορίζει τον τύπο του αιτήματος. Καθοδηγεί τη δευτερεύουσα συσκευή αναφορικά με τον πίνακα που πρέπει να προσπελάσει, καθώς και καθορίζει εάν πρέπει να διαβάσει ή να γράψει από αυτόν.

Ακολουθεί ένας πίνακας με τους πιο συνηθισμένους κωδικούς λειτουργίας ενός αιτήματος στο πρωτόκολλο Modbus.

Κωδικός λειτουργίας	Περιγραφή λειτουργίας
1	Ανάγνωση σε ένα ή περισσότερα πηνία
2	Ανάγνωση μίας ή περισσοτέρων διακριτών εισόδων
3	Ανάγνωση σε έναν ή περισσότερους καταχωρητές αποθήκευσης
4	Ανάγνωση σε έναν ή περισσότερους καταχωρητές εισόδου
5	Εγγραφή σε ένα πηνίο
6	Εγγραφή σε έναν καταχωρητή αποθήκευσης
15	Εγγραφή σε πολλαπλά πηνία
16	Εγγραφή σε πολλαπλούς καταχωρητές αποθήκευσης

Πίνακας 1 : Συνήθεις κωδικοί λειτουργίας (function codes) σε ένα αίτημα Modbus [9], [12]

Τέλος, το πεδίο δεδομένων περιέχει τα δεδομένα που θα σταλούν ως απάντηση στην κύρια συσκευή, εάν το αίτημα αφορά λειτουργία ανάγνωσης.



Εικόνα 2-7: τα πεδία ενός αιτήματος από μια κύρια συσκευή σε μια δευτερεύουσα.

Επιπλέον, η κύρια συσκευή μπορεί, πέρα από το να διαβάσει και να γράψει δεδομένα, να εκτελέσει διαγνωστικές λειτουργίες σε μια δευτερεύουσα. Οι

διαγνωστικές λειτουργίες επιτρέπουν στην κύρια συσκευή να ελέγχει την κατάσταση μιας δευτερεύουσας, καθώς και να αναγνωρίζει πιθανά σφάλματα.

Παρακάτω, ακολουθεί μια απλοποιημένη αλληλουχία επικοινωνίας μεταξύ κύριας και δευτερεύουσας συσκευής, προκειμένου να διασαφηνιστεί ο τρόπος όπου αυτή λαμβάνει χώρα.

1. Η κύρια συσκευή στέλνει ένα αίτημα σε μια συγκεκριμένη δευτερεύουσα συσκευή, χρησιμοποιώντας τη διεύθυνση της δευτερεύουσας και έναν κωδικό λειτουργίας που καθορίζει τον τύπο του αιτήματος. Το αίτημα μπορεί να περιέχει δεδομένα, όπως μια εντολή ή μια τιμή που πρέπει να εγγραφεί σε έναν καταχωρητή.

2. Η συσκευή slave λαμβάνει το αίτημα και το επεξεργάζεται. Εάν το αίτημα αφορά λειτουργία ανάγνωσης, η συσκευή slave ανακτά τα ζητούμενα δεδομένα από την εσωτερική της μνήμη. Εάν το αίτημα είναι αίτημα εγγραφής, η συσκευή slave αποθηκεύει τα δεδομένα στον καθορισμένο καταχωρητή ή πηνίο.

3. Η συσκευή slave αποστέλλει μια απάντηση στην κύρια συσκευή, η οποία περιλαμβάνει τα ζητούμενα δεδομένα ή έναν κωδικό κατάστασης που υποδεικνύει την επιτυχία ή την αποτυχία του αιτήματος. Η απάντηση περιλαμβάνει επίσης τη διεύθυνση της slave συσκευής και τον ίδιο κωδικό λειτουργίας με το αίτημα.

4. Η κύρια συσκευή λαμβάνει την απάντηση και την επεξεργάζεται. Εάν η απάντηση περιέχει δεδομένα, η κύρια συσκευή μπορεί να χρησιμοποιήσει τα δεδομένα για να ενημερώσει την εσωτερική της κατάσταση ή για να ελέγξει τη λειτουργία της δευτερεύουσας συσκευής. Εάν η απόκριση περιέχει κωδικό σφάλματος, η κύρια συσκευή μπορεί να προβεί στις κατάλληλες ενέργειες, όπως η επανάληψη της αίτησης ή η εμφάνιση ενός μηνύματος σφάλματος. Οι κωδικοί σφάλματος αναλύονται αναλυτικότερα παρακάτω.

Αποθήκευση δεδομένων

Οι πληροφορίες αποθηκεύονται σε τέσσερις διαφορετικούς πίνακες. Δύο πίνακες αποθηκεύουν διακριτές τιμές (πηνία) και δύο αποθηκεύουν αριθμητικές τιμές (καταχωρητές). Τα πηνία και οι καταχωρητές διαθέτουν από έναν PDU πίνακα μόνο

για ανάγνωση και έναν πίνακα ανάγνωσης-εγγραφής. Κάθε πίνακας έχει 9999 τιμές. Κάθε πηνίο είναι 1 bit και του αποδίδεται μια διεύθυνση δεδομένων μεταξύ 0000 και 270E. Κάθε καταχωρητής είναι 1 λέξη = 16 bits = 2 bytes και έχει επίσης διεύθυνση δεδομένων μεταξύ 0000 και 270E. Τα παραπάνω αναγράφονται και στον πίνακα 2 που ακολουθεί παρακάτω:

Αριθμός Πηνίων/Καταχωρητών	**Διευθύνσεις Μνήμης**	**Λειτουργία**	**Όνομα Πίνακα**
1 – 9999	0000 έως 270E	Ανάγνωσης – Εγγραφής	Πηνία διακριτής εξόδου
10001 – 19999	0000 έως 270E	Ανάγνωσης	Πηνία διακριτής εισόδου
30001 – 39999	0000 έως 270E	Ανάγνωσης	Καταχωρητές αναλογικής εισόδου
40001 – 49999	0000 έως 270E	Ανάγνωσης – Εγγραφής	Καταχωρητές αποθήκευσης αναλογικής εξόδου

Πίνακας 2: Οργάνωση της μνήμης [11], [12]

Μορφές πινάκων αποθήκευσης

Οι πίνακες καταχωρητών έχουν τις κολώνες που απεικονίζονται παρακάτω στον πίνακα 3:

Address	Register	No	RW	X	Unit	Type	Range	Default Value	Svd	Function Code	Applicable Devices	Description

Πίνακας 3: Κολώνες των πινάκων αποθήκευσης καταχωρητών [13]

Στον πίνακα 4 που ακολουθεί, παρατίθεται μια συνοπτική περιγραφή των κολωνών του πίνακα αποθήκευσης και των λειτουργιών τους.

Ονομασία	Περιγραφή
Διεύθυνση	16-bit διεύθυνση καταχωρητή σε δεκαεξαδικό σύστημα. Η διεύθυνση περιέχει τα δεδομένα που χρησιμοποιούνται.
Καταχωρητής	Αριθμός καταχωρητή 16-bit σε δεκαδικό σύστημα. Καταχωρητής = Διεύθυνση + 1
No	Αριθμός καταχωρητών 16-bit που πρέπει να διαβαστούν/εγγραφούν για πρόσβαση στις πλήρεις πληροφορίες.
R/RW	Εάν ο καταχωρητής είναι μόνο για ανάγνωση (R/RW) ή για ανάγνωση-εγγραφή (RW).
X	Συντελεστής κλίμακας: • Κλίμακα 1 σημαίνει ότι η τιμή του καταχωρητή είναι η σωστή με την υποδεικνυόμενη μονάδα. • Κλίμακα 10 σημαίνει ότι ο καταχωρητής περιέχει την τιμή πολλαπλασιασμένη επί 10. Η πραγματική τιμή είναι επομένως η τιμή του καταχωρητή διαιρούμενη με το 10. • Κλίμακα 0,1 σημαίνει ότι ο καταχωρητής περιέχει την τιμή πολλαπλασιασμένη επί 0,1. Η πραγματική τιμή είναι επομένως η τιμή του καταχωρητή πολλαπλασιασμένη επί 10.
Μονάδα	Μονάδα μέτρησης πληροφοριών: • "-": δεν υπάρχει μονάδα που να αντιστοιχεί στην εκφραζόμενη τιμή. • "h": ώρες • "D": η μονάδα εξαρτάται από τη συνδεδεμένη συσκευή.

Τύπος	Τύπος δεδομένων κωδικοποίησης (βλ. παρακάτω τον πίνακα 7: Τύποι δεδομένων).
Εύρος	Εύρος επιτρεπόμενων τιμών για τη μεταβλητή, συνήθως ένα υποσύνολο των τιμών που επιτρέπει η μορφή. Για δεδομένα τύπου BITMAP, το περιεχόμενο αυτού του τομέα είναι “–”.
Προεπιλεγμένη τιμή	Προεπιλεγμένη τιμή για τη μεταβλητή
Svd	Τιμή που αποθηκεύεται όταν απενεργοποιείται η παροχή ρεύματος στην πύλη PowerTag Link: • "Y": η τιμή του καταχωρητή αποθηκεύεται. • "N": η τιμή χάνεται. ΣΗΜΕΙΩΣΗ: Κατά την έναρξη ή την επαναφορά λειτουργίας, ανακτώνται οι διαθέσιμες τιμές.
Κωδικός λειτουργίας	Κωδικός των λειτουργιών που μπορούν να χρησιμοποιηθούν στον καταχωρητή.
Συσκευές	Κωδικός που υποδεικνύει τους τύπους συσκευών για τους οποίους είναι διαθέσιμος ο καταχωτητής.
Περιγραφή	Πληροφορίες σχετικές με τον καταχωρητή και τους ισχύων περιορισμούς.

Πίνακας 4: Περιγραφή κολωνών των πινάκων αποθήκευσης [13]

Εξαιρέσεις

Όταν μία δευτερεύουσα συσκευή Modbus διαπιστώνει πως υπάρχει σφάλμα στο αίτημα που έχει δεχτεί, τότε επιστρέφει μια απάντηση που περιέχει κωδικό εξαίρεσης. Η απάντηση εξαίρεσης αποτελείται από τη διεύθυνση ή τον αριθμό μονάδας της δευτερεύουσας συσκευής, ένα αντίγραφο του κωδικού λειτουργίας με

το υψηλό bit ρυθμισμένο και έναν κωδικό εξαίρεσης [14]. Για παράδειγμα, εάν ο κωδικός λειτουργίας ήταν 3, ο κωδικός λειτουργίας στην απάντηση εξαίρεσης θα είναι 0x83. Οι κωδικοί εξαίρεσης παρατίθενται παρακάτω στον πίνακα 5:

Κωδικός	**Όνομα εξαίρεσης**	**Αιτία Εξαίρεσης**
1	Μη έγκυρη ενέργεια	Ο κωδικός λειτουργίας που λαμβάνεται στο ερώτημα δεν αναγνωρίζεται ή δεν επιτρέπεται από τη δευτερεύουσα συσκευή.
2	Μη έγκυρη διεύθυνση δεδομένων	Η διεύθυνση δεδομένων (αριθμός καταχωρητή) που λαμβάνεται στο ερώτημα δεν είναι μια επιτρεπόμενη διεύθυνση για τη δευτερεύουσα συσκευή, δηλαδή ο καταχωρητής δεν υπάρχει. Εάν ζητήθηκαν πολλαπλοί καταχωρητές, τουλάχιστον ένας δεν ήταν επιτρεπτός.
3	Μη έγκυρη τιμή δεδομένων	Η τιμή που περιέχεται στο πεδίο δεδομένων του ερωτήματος δεν είναι αποδεκτή από τη δευτερεύουσα συσκευή.
4	Αποτυχία δευτερεύουσας συσκευής	Εμφανίστηκε σφάλμα που δεν μπορεί να αποκατασταθεί.
6	Η δευτερεύουσα συσκευή είναι απασχολημένη	Η δευτερεύουσα είναι απασχολημένη με την επεξεργασία μιας εντολής μεγάλης διάρκειας. Η κύρια συσκευή θα πρέπει να προσπαθήσει ξανά αργότερα.
10 (hex 0A)	Διαδρομή πύλης μη-διαθέσιμη	Η επικοινωνία με τη συσκευή-στόχο μέσω της πύλης δεν επιτεύχθηκε.

11 (hex 0B)	Η συσκευή-στόχος της πύλης απέτυχε να ανταποκριθεί	Ειδική χρήση που αφορά επικοινωνία με πύλες. Υποδεικνύει ότι δεν ελήφθη απάντηση από τη συσκευή-στόχο.
17 (hex 11)	Η συσκευή-στόχος της πύλης απέτυχε να ανταποκριθεί	Δεν λήφθηκε καμία απάντηση από τη δευτερεύουσα συσκευή. Ξεπεράστηκε το ανώτατο χρονικό όριο απόκρισης για ένα αίτημα.

Πίνακας 5: Κωδικοί εξαίρεσης [15]

Προκειμένου να χρησιμοποιηθούν οι παραπάνω κωδικοί εξαίρεσης, είναι απαραίτητο να ανιχνευτούν τυχόν σφάλματα. Για το σκοπό αυτό, αποθηκεύονται δύο προστιθέμενα bytes στο τέλος κάθε μηνύματος. Η διαδικασία ανίχνευσης σφαλμάτων χρησιμοποιεί όλα τα bytes του μηνύματος, καθώς επίσης και τα 2 προστιθέμενα bytes. Το πλήρες μήνυμα μαζί με τα προστιθέμενα bytes ονομάζεται CRC (Κυκλικός έλεγχος πλεονασμού - Cyclic Redundancy Check). Η συσκευή που λαμβάνει ένα μήνυμα υπολογίζει τον CRC και τον συγκρίνει με τον CRC από τη συσκευή αποστολής. Εάν έστω και ένα bit του μηνύματος ληφθεί εσφαλμένα, οι CRC θα είναι διαφορετικοί και θα προκύψει σφάλμα.

Τύποι δεδομένων

Το Modbus υποστηρίζει πολλούς διαφορετικούς τύπους δεδομένων, συμπεριλαμβανομένων δυαδικών, ακέραιων και τιμών κινητής υποδιαστολής. Υποστηρίζει επίσης διαφορετικές μορφές δεδομένων, όπως big-endian και small-endian, οι οποίες καθορίζουν τη σειρά με την οποία είναι τακτοποιημένα τα bit στο μήνυμα. Ακολουθούν οι τύποι δεδομένων στο πρωτόκολλο Modbus:

Όνομα	Περιγραφή	Εύρος τιμών
INT16	16-bit προσημασμένος ακέραιος (1 λέξη)	-32768 έως +32767
UINT16	16-bit ακέραιος χωρίς πρόσημο (1 λέξη)	0 έως 65535
INT32	32-bit προσημασμένος ακέραιος (2 λέξεις)	-2.147.483.648 έως +2.147.483.647
UINT32	32-bit ακέραιος χωρίς πρόσημο (2 λέξεις)	0 έως 4.294.967.295
INT64	64-bit προσημασμένος ακέραιος (4 λέξεις)	-9.223.372.036.854.775.808 έως 9.223.372.036.854.775.807
UINT64	64-bit ακέραιος χωρίς πρόσημο (4 λέξεις)	0 έως 18.446.744.073.709.600.000
Float32	32-bit τιμή (2 λέξεις)	-3.4028E+38 έως +3.4028E+38
ASCII	8-bit αλφαριθμητικός χαρακτήρας	Πίνακας χαρακτήρων ASCII
BITMAP	16-bit πεδίο (1 λέξη)	–
DATETIME	Ανατρέξτε παρακάτω στον πίνακα X: DATETIME	–

Πίνακας 6: Τύποι δεδομένων [13]

Σημειώσεις:

- Δεδομένα τύπου Float32: έχουν 8-bit εκθέτη, 23-bit mantissa (θετική, αρνητική, κανονικοποιημένη πραγματική).

- Για δεδομένα τύπου ASCII, η σειρά μετάδοσης των χαρακτήρων σε λέξεις (καταχωρητές 16 bit) έχει ως εξής:
  1. Χαρακτήρας n ως λιγότερο σημαντικός
  2. Χαρακτήρας n + 1 ως πιο σημαντικός
- Όλοι οι καταχωρητές (16-bit ή 2 bytes) μεταδίδονται με κωδικοποίηση Big Endian:
  1. Το πιο σημαντικό byte μεταδίδεται πρώτο.
  2. Το λιγότερο σημαντικό byte μεταδίδεται δεύτερο.
- Οι μεταβλητές 32-bit και 64-bit που αποθηκεύονται σε λέξεις των 16-bit είναι σε μορφή Big Endian: Η πιο σημαντική λέξη μεταδίδεται πρώτη και ύστερα η λιγότερο σημαντική.

DATETIME

Καταχωρητής	**Τύπος**	**Bit**	**Εύρος**	**Περιγραφή**
1	INT16U	0–6	0x00–0x7F	Έτος: 0x00 (00) έως 0x7F (127) αντιστοιχούν στα έτη 2000 έως 2127 Για παράδειγμα, το 0x0D (13) αντιστοιχεί στο έτος 2013.
		7–15	–	Δεσμευμένα
2	INT16U	0–4	0x01–0x1F	Μέρα
		5–7	–	Δεσμευμένα
		8–11	0x00–0x0C	Μήνας
		12–15	–	Δεσμευμένα
3	INT16U	0–5	0x00–0x3B	Λεπτά
		6–7	–	Δεσμευμένα
		8–12	0x00–0x17	Ώρες

		13–15	–	Δεσμευμένα
4	INT16U	0–15	0x0000–0xEA5F	Χιλιοστά του δευτερολέπτου

Πίνακας 7: Τύπος «Datetime» [13]

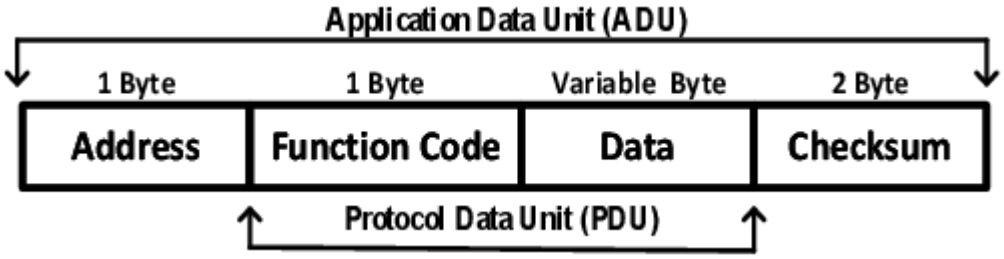
Το DATETIME είναι ένας τύπος δεδομένων που χρησιμοποιείται για την κωδικοποίηση της ημερομηνίας και της ώρας και ορίζεται από το πρότυπο IEC 60870-5. [13]

Εκδοχές του πρωτοκόλλου Modbu

Μια εκδοχή του Modbus αποτελείται από μια Μονάδα Δεδομένων Εφαρμογής (Application Data Unit - ADU), η οποία ενσωματώνει μια Μονάδα Δεδομένων Πρωτοκόλλου (Protocol Data Unit - PDU)

Η ADU αποτελείται από τη διεύθυνση δευτερεύουσας συσκευής, την PDU και το πεδίο ελέγχου σφάλματος.

Η PDU αποτελείται από τον κωδικός λειτουργίας και το πεδίο δεδομένων.



Εικόνα 2-8: Οι μονάδα δεδομένων εφαρμογής (ADU) και η μονάδα δεδομένων προτοκόλλου (PDU).

Όλες οι εκδοχές του πρωτοκόλλου Modbus που παραθέτονται παρακάτω υλοποιούν με διαφορετικό τρόπο τις μονάδες ADU και PDU.

MODBUS RTU

Το Modbus RTU είναι η πιο κοινή υλοποίηση που διατίθεται για το Modbus. Πρόκειται απλά για το Modbus μέσω σειριακών μέσων επικοινωνίας όπως τα RS485, RS422 και RS232.

Στο RTU χρησιμοποιούνται τα περισσότερα από αυτά που αναλύθηκαν διεξοδικά παραπάνω. Οι τυπικές διευθύνσεις κόμβων Modbus RTU είναι 1-255 με το 0 να προορίζεται για μηνύματα εκπομπής (broadcast) και μόνο για εγγραφή.

Το μέγεθος του σειριακού PDU περιορίζεται από τον περιορισμό μεγέθους που κληρονομήθηκε από την πρώτη υλοποίηση του σειριακού δικτύου Modbus των 256 bytes. Οι διευθύνσεις των δευτερευόντων συσκευών περιορίζονται σε 1-255. Οι διευθύνσεις 1-247 είναι διαθέσιμες στο χρήστη και οι διευθύνσεις 248-255 είναι δεσμευμένες. [7], [11], [16]

MODBUS TCP/IP Ή MODBUS TCP

Το Modbus TCP ή TCP/IP είναι ουσιαστικά το Modbus RTU σε ένα πακέτο Ethernet (IEEE 802.3) με τη διεύθυνση προορισμού ως διεύθυνση IP χρησιμοποιώντας το πρωτόκολλο συναλλαγών TCP/IP. Η θύρα TCP 502 προορίζεται για το Modbus, ενώ η νέα ασφάλεια Modbus/TCP χρησιμοποιεί τη θύρα 802. Επίσης γνωστό ως Modbus IP, Modbus Ethernet και Modbus TCP/IP.

Το Modbus TCP εκτελείται σε φυσικό επίπεδο Ethernet. Χρησιμοποιεί μια επικεφαλίδα 6 byte για να επιτρέπει τη δρομολόγηση. Επιτρέπει πολλαπλές δευτερεύον συσκευές και δεν περιορίζεται σε 32, όπως συμβαίνει με το σειριακό μέσο μετάδοσης RS-485. Σε αντίθεση με το σειριακό Modbus RTU, στο οποίο είναι δυνατή η σύνδεση όλων των συσκευών μεταξύ τους, το Modbus TCP (Ethernet) απαιτεί ένα διακόπτη (switch) για τη σύνδεση πολλαπλών συσκευών.

Τα μηνύματα στο Modbus TCP/IP αποτελούνται από τον κωδικό λειτουργίας Modbus και το αίτημα δεδομένων Modbus. Η διεύθυνση της δευτερεύον συσκευής και ο κωδικός σφάλματος (CRC) συνήθως δεν χρειάζονται, καθώς το πακέτο Modbus TCP/IP δρομολογείται από το δίκτυο στην επιθυμητή διεύθυνση IP και ο έλεγχος σφάλματος γίνεται ως μέρος του Ethernet πακέτου. Εξαίρεση αποτελεί η περίπτωση όπου είναι απαραίτητη η σύνδεση και επικοινωνία με σειριακό δίκτυο. Σε αυτήν την περίπτωση είναι απαραίτητη και η μετάδοση των προαναφερθέντων πεδίων.

Στο δίκτυο Modbus TCP/IP χρησιμοποιείται το δίκτυο Modbus Server (διακομιστής) καθώς και Modbus Client (πελάτης). Οποιαδήποτε συσκευή Modbus συνδέεται με διακόπτες (switches) ή σειρά διακοπτών Ethernet.[7], [16]

MODBUS ASCII

Αφορά χρήση σε σειριακή επικοινωνία και χρησιμοποιεί χαρακτήρες ASCII για την επικοινωνία μεταξύ των συσκευών. Πλεονέκτημα, ανάλογα με το σκοπό χρήσης, αποτελεί πως οι χαρακτήρες ASCII μπορούν να διαβαστούν και από ανθρώπου. Τα μηνύματα στο Modbus ASCII πλαισιώνονται από εμπρόσθια άνω και κάτω τελεία (":") και οπίσθια νέα γραμμή (CR/LF). Χρησιμοποιείται σε περιπτώσεις όπου η ακεραιότητα των δεδομένων δεν είναι από τις πρώτες προτεραιότητες.

MODBUS PLUS (MODBUS+, MB+ Ή MBP)

Το Modbus Plus δεν είναι απλώς πρωτόκολλο, πρόκειται για πλήρες σύστημα με προκαθορισμένο μέσο και υλοποίηση φυσικού επιπέδου (επίπεδο 1 OSI). Είναι ένα σύστημα LAN για εφαρμογές βιομηχανικού ελέγχου. Το Modbus Plus χρησιμοποιεί μηχανισμό ελέγχου πρόσβασης στο μέσο με χρήση token, ο οποίος έχει ως αποτέλεσμα ντετερμινιστική λειτουργία, αν και όχι απαραίτητα γρήγορη, συγκριτικά με τα άλλα πρωτόκολλα.

Το πρωτόκολλο ανταλλαγής μηνυμάτων επιπέδου 7 (επίπεδο εφαρμογών στο OSI) του Modbus Plus είναι ουσιαστικά το ίδιο με αυτό που χρησιμοποιείται για το Modbus Serial και το Modbus/TCP. Το φυσικό επίπεδο υλοποιείται με RS-485 και λειτουργεί μέσω θωρακισμένου καλωδίου συνεστραμμένου ζεύγους. Το πρωτόκολλο του επιπέδου σύνδεσης δεδομένων (OSI επίπεδο 2) βασίζεται στο πρωτόκολλο πολλαπλής πτώσης HDLC (High-level Data Link Control) του ISO/IEC 3309:1991, το οποίο χρησιμοποιεί μηχανισμό ελέγχου πρόσβασης στο μέσο με χρήση token και μεταδίδει τα δεδομένα με σύγχρονο τρόπο, σε αντίθεση με την ασύγχρονη μετάδοση του Modbus Serial. Αυτό έχει ως αποτέλεσμα τη μετάδοση δεδομένων με ταχύτητα 1 Mbps. [16]

Profinet

Το Profinet είναι ένα πρωτόκολλο δικτύου που χρησιμοποιείται για τον βιομηχανικό αυτοματισμό. Βασίζεται στην τεχνολογία Ethernet και σήμερα είναι από τα σημαντικότερα πρωτόκολλα Industrial Ethernet. Χρησιμοποιείται για τη σύνδεση βιομηχανικών συσκευών, όπως PLC, αισθητήρες και ηλεκτρονικές μονάδες κίνησης.

Το Profinet είναι ένα πρωτόκολλο που είναι σχεδιασμένο να λειτουργεί σε πραγματικό χρόνο και είναι εξαιρετικά αξιόπιστο λόγω των ενσωματωμένων χαρακτηριστικών ανοχής σφαλμάτων. Χρησιμοποιεί συγχρονισμένα μηνύματα, τα οποία αποστέλλονται μέσω του διαύλου και χρησιμοποιούνται για επικοινωνία σε πραγματικό χρόνο. Υποστηρίζει επίσης ένα ευρύ φάσμα λειτουργιών, όπως απομακρυσμένη διάγνωση συσκευών, παραμετροποίηση και έλεγχο μέσω διαδικτύου, καθώς επίσης και χειρισμό ειδοποιήσεων (alerts).

Το Profinet είναι ένα πρωτόκολλο Ethernet επιπέδου 2 και χρησιμοποιεί συνδεσμολογία αστέρα με κεντρικό switch. Υποστηρίζει καλωδίωση χαλκού ή/και οπτικών ινών και μπορεί να διαμορφωθεί ώστε να υποστηρίζει έως και 64 συσκευές ανά τμηματοποίηση. Είναι σχεδιασμένο για τη διαχείριση δεδομένων υψηλής ταχύτητας και χρησιμοποιεί ένα συνδυασμό κυκλικών συγχρονισμένων υπηρεσιών πραγματικού χρόνου (IRT) και ασύγχρονων υπηρεσιών.

Διαμέσου του Profinet κατασκευάζονται ασφαλή δίκτυα με έλεγχο στις συσκευές που συνδέονται, καθιστώντας το ιδανικό για βιομηχανικές εφαρμογές. Υποστηρίζει επίσης ποικιλία διαγνωστικών και διαχειριστικών λειτουργιών, όπως παρακολούθηση της κατάστασης της συσκευής και του δικτύου, καθώς και απομακρυσμένες ενημερώσεις λογισμικού. Το Profinet αποτελεί ένα αξιόπιστο και ασφαλές πρωτόκολλο που χρησιμοποιείται ευρέως σε εφαρμογές βιομηχανικού αυτοματισμού. Παρέχει τα απαραίτητα χαρακτηριστικά όπου είναι απαραίτητα σε σύγχρονες εφαρμογές στη βιομηχανία. [17]

OPC - UA

Το OPC Unified Architecture (OPC UA) είναι ένα βιομηχανικό πρωτόκολλο επικοινωνίας που σχεδιάστηκε για να αντιμετωπίσει τις ελλείψεις του προκατόχου του, του OPC Classic. Το OPC UA αναπτύχθηκε από τον οργανισμό OPC Foundation, κυκλοφόρησε το 2006 και έχει γίνει το de facto πρότυπο για την ασφαλή, αξιόπιστη και ανεξάρτητη από το υλικό ανταλλαγή δεδομένων σε συστήματα βιομηχανικού αυτοματισμού, παραγωγής και ελέγχου διεργασιών.

Το OPC UA βασίζεται σε ένα ισχυρό και ευέλικτο πλαίσιο που παρέχει απρόσκοπτη επικοινωνία μεταξύ διαφόρων συσκευών και συστημάτων, ανεξάρτητα από την υποκείμενη τεχνολογία, το λειτουργικό σύστημα ή τη γλώσσα προγραμματισμού. Σε αντίθεση με το OPC Classic, το οποίο βασιζόταν στην πλατφόρμα Microsoft Windows και την τεχνολογία COM/DCOM, το OPC UA χρησιμοποιεί binary TCP/IP ή εναλλακτικά SOAP. Αυτό του επιτρέπει να τρέχει σε οποιοδήποτε λειτουργικό σύστημα, ακόμη και σε ενσωματωμένα συστήματα, καθιστώντας το πιο ευέλικτο και προσιτό για ένα ευρύτερο φάσμα εφαρμογών. Επιπλέον, έχει σχεδιαστεί για να κλιμακώνεται από μικρές συσκευές με περιορισμένους πόρους έως μεγάλα, πολύπλοκα βιομηχανικά συστήματα, διασφαλίζοντας ότι το ίδιο πρωτόκολλο μπορεί να χρησιμοποιηθεί σε διάφορα επίπεδα ενός οργανισμού. Από την έκδοση 1.04 (2017), υποστηρίζει επικοινωνία με το μοντέλο δημοσίευσης – συνδρομής (publish – subscribe).

Αρχιτεκτονική

Το πρότυπο OPC UA αποτελείται από επιμέρους προδιαγραφές. Κάθε προδιαγραφή περιγράφει μια μερική λειτουργία και καθορίζει τις διεπαφές διακομιστή και πελάτη που πρέπει να υλοποιηθούν για τη λειτουργία αυτή προκειμένου να υποστηριχθεί. Οι διακομιστές και οι πελάτες OPC δεν χρειάζεται να υποστηρίζουν όλες τις προδιαγραφές. Ανάλογα με την εφαρμογή, συχνά προγραμματίζονται μόνο μεμονωμένες προδιαγραφές. Κατά τη χρήση ενός διακομιστή και πελατών OPC, είναι επομένως σημαντικό να εξετάζεται ποιες

προδιαγραφές απαιτούνται και ποιες υλοποιούνται από τον διακομιστή και τον πελάτη.

Το OPC UA αποτελείται από αυτές τις προδιαγραφές:

1. Έννοιες (concepts)
2. Μοντέλο ασφάλειας (security model)
3. Μοντέλο χώρου διευθύνσεων (address space model)
4. Υπηρεσίες (services)
5. Μοντέλο πληροφοριών (information model)
6. Αντιστοιχίσεις (mappings)
7. Προφίλ (profiles)
8. Πρόσβαση στα δεδομένα (data access)
9. Συναγερμοί και περιπτώσεις (alerts and conditions)
10. Προγράμματα (programs)
11. Ιστορική πρόσβαση (historical access)
12. Ανακάλυψη (discovery)
13. Συγκεντρωτικά στοιχεία (aggregates)
14. PubSub (δημοσίεση – συνδρομή, publish – subscribe)

Για τη χρήση του εν λόγω πρωτοκόλλου, δεν είναι απαραίτητη η λεπτομερή γνώση όλων των προδιαγραφών. Ακολουθούν μερικά δομικά στοιχεία της αρχιτεκτονικής του πρωτοκόλλου, καθώς και μερικές σημαντικές OPC UA προδιαγραφές.

Διακομιστής (Server): Ο διακομιστής OPC είναι η βάση της επικοινωνίας OPC. Πρόκειται για ένα λογισμικό που υλοποιεί το πρότυπο OPC και εκθέτει δεδομένα και υπηρεσίες από συσκευές ή συστήματα με τυποποιημένο τρόπο.

Πελάτης (Client): Ο πελάτης OPC επικοινωνεί με τον διακομιστή και μπορεί να διαβάσει δεδομένα που παρέχονται. Δεδομένου ότι οι διακομιστές υλοποιούν τις προκαθορισμένες διεπαφές του προτύπου OPC, κάθε πελάτης μπορεί να έχει πρόσβαση σε οποιονδήποτε διακομιστή OPC και να ανταλλάσσει δεδομένα με τον διακομιστή με τον ίδιο τρόπο.

Χώρος διευθύνσεων (address space): Ο χώρος διευθύνσεων είναι μια ιεραρχική αναπαράσταση των δεδομένων, των μεταδεδομένων και των υπηρεσιών που εκτίθενται από τον διακομιστή OPC UA.

Μοντέλο πληροφοριών: Το μοντέλο πληροφοριών ορίζει τις δομές δεδομένων και τις σχέσεις μεταξύ τους, επιτρέποντας την τυποποιημένη και επεκτάσιμη αναπαράσταση δεδομένων.

Υπηρεσίες: Το OPC UA παρέχει ένα σύνολο υπηρεσιών για την πρόσβαση και τον χειρισμό δεδομένων, καθώς και για τη διαχείριση συνδρομών (subscriptions), συναγερμών (alerts) και γεγονότων (events).

Πρόσβαση σε δεδομένα (data access): Η προδιαγραφή πρόσβασης σε δεδομένα περιγράφει την κλασική ανταλλαγή τρεχόντων δεδομένων. Το κλασικό πρότυπο OPC ορίζει ήδη ότι η ανταλλαγή δεδομένων είναι προσανατολισμένη στα σημεία δεδομένων (data points). Μια τιμή μπορεί να διαβαστεί και να γραφτεί για κάθε σημείο δεδομένων. Η τιμή ενός σημείου δεδομένων περιγράφεται από την πραγματική τιμή, τη χρονική σφραγίδα κατά την οποία η τιμή ήταν η τρέχουσα και από την ποιότητα, η οποία περιγράφει εάν η τιμή είναι έγκυρη. Αυτή και μόνο η προδιαγραφή καθιστά δυνατή τη λήψη και την επεξεργασία δεδομένων ανεξάρτητα από το εκάστοτε σύστημα. Υπάρχει επιπλέον η δυνατότητα καθορισμού σύνθετων τύπων δεδομένων (structures) και συναρτήσεων.

Ιστορική πρόσβαση (historical access): Χρησιμοποιώντας την προδιαγραφή historical access, είναι δυνατή όχι μόνο η ανάγνωση δεδομένων με την τρέχουσα τιμή, αλλά και η αναζήτηση ιστορικών τιμών. Ένας διακομιστής που εφαρμόζει αυτή την προδιαγραφή πρέπει να διαθέτει εσωτερική μνήμη δεδομένων για να αποθηκεύει τις τιμές των δεδομένων για πιθανές ιστορικές προσβάσεις.

Συναγερμοί και περιπτώσεις (alarms and conditions): Η προδιαγραφή Alarms and Conditions ορίζει ένα τυποποιημένο μοντέλο για μηνύματα συναγερμού. Για τις εφαρμογές πελάτη, αυτό απλοποιεί το έργο της δημιουργίας συναγερμών, επειδή η λογική μπορεί να υλοποιηθεί από τον διακομιστή OPC και όχι από τον κατασκευαστή του λογισμικού του πελάτη.

Ασφάλεια (security): Κατά την ανάπτυξη του προτύπου OPC UA, λήφθηκε εξαρχής υπόψη ο υψηλότερος βαθμός ασφάλειας. Σε αντίθεση με το OPC Classic, το OPC UA αναπτύχθηκε "με τείχος προστασίας" (firewall). Στο επίπεδο της μεταφοράς, μπορεί να χρησιμοποιηθεί απευθείας ένα δυαδικό πρωτόκολλο στο TCP/IP για γρήγορες εφαρμογές ή το cross-platform SOAP με HTTPS.

Για την ασφάλεια των δεδομένων κατά τη μετάδοση χρησιμοποιούνται κρυπτογραφήσεις 128 ή 256 bit, καθώς και υπογραφή μηνυμάτων, αλληλουχία πακέτων και πιστοποίηση ταυτότητας χρήστη (user authentication).

Το OPC UA χρησιμοποιεί μια ανταλλαγή πιστοποιητικών για περαιτέρω ασφάλεια, έτσι ώστε κάθε πελάτης να πρέπει να πιστοποιείται με ένα πιστοποιητικό. Με αυτόν τον τρόπο μπορεί να ελεγχθεί ποιος πελάτης επιτρέπεται να συνδεθεί με τον διακομιστή.

Επίλογος

Το 2011, ο όρος «Industry 4.0» χρησιμοποιήθηκε για πρώτη φορά από μια ομάδα εργασίας του ινστιτούτου Research Union Economy – Science of the BMBF. Το OPC UA είχε ήδη οριστεί ως πρότυπο πολύ νωρίτερα. Παρόλα αυτά, το OPC UA είναι ένα από τα κορυφαία πρωτόκολλα επικοινωνίας για αυτό που ονομάζουμε «Industry 4.0». Η ευφυής δικτύωση των εργοστασίων απαιτεί μια κοινή γλώσσα. Αυτό ακριβώς προσφέρει το OPC UA και, ως εκ τούτου, αποτελεί σημαντικό εργαλείο για την υλοποίηση του Industry 4.0.

PLC4X

Το PLC4X είναι μια βιβλιοθήκη βιομηχανικής επικοινωνίας ανοικτού κώδικα που δημιουργήθηκε από το Ίδρυμα Λογισμικού Apache. Σημαίνει Programmable Logic Controller for Anything (Προγραμματιζόμενος λογικός ελεγκτής για οτιδήποτε). Ο πρωταρχικός στόχος του PLC4X είναι να παρέχει μια ενιαία διεπαφή για την επικοινωνία με διάφορους προγραμματιζόμενους λογικούς ελεγκτές (PLC) και συσκευές βιομηχανικού αυτοματισμού, ανεξάρτητα από τον προμηθευτή ή το πρωτόκολλο που χρησιμοποιείται.

Υπάρχουν πολυάριθμα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται από διάφορους προμηθευτές PLC, όπως τα Modbus, OPC-UA, S7 και EtherNet/IP. Το PLC4X στοχεύει στην απλοποίηση και την τυποποίηση της επικοινωνίας με αυτές τις συσκευές, προσφέροντας ένα ενιαίο API που οι προγραμματιστές μπορούν να χρησιμοποιούν για να αλληλεπιδρούν με διάφορα πρωτόκολλα.

Το PLC4X παρέχει υλοποιήσεις για διάφορα βιομηχανικά πρωτόκολλα επικοινωνίας. Αυτές οι υλοποιήσεις χειρίζονται τις ιδιαιτερότητες της επικοινωνίας για κάθε πρωτόκολλο, όπως η κωδικοποίηση, η αποκωδικοποίηση, η διευθυνσιοδότηση και η συνδεσιμότητα. Με την υποστήριξη πολλαπλών πρωτοκόλλων, το PLC4X δίνει τη δυνατότητα στους προγραμματιστές να εργάζονται με διαφορετικούς τύπους PLC χωρίς να απαιτείται εξειδίκευση σε κάθε πρωτόκολλο.

Το PLC4X παρέχει ένα ενιαίο, ενοποιημένο API που οι προγραμματιστές μπορούν να χρησιμοποιούν για να επικοινωνούν με PLC, ανεξάρτητα από το υποκείμενο πρωτόκολλο. Αυτό το API αφαιρεί την πολυπλοκότητα και τις διαφορές μεταξύ των διαφόρων πρωτοκόλλων, απλοποιώντας τη διαδικασία συγγραφής εφαρμογών που αλληλεπιδρούν με τα PLC. Οι προγραμματιστές χρειάζεται να μάθουν μόνο το API του PLC4X, αντί να χρειάζεται να κατανοήσουν τις ιδιαιτερότητες κάθε μεμονωμένου πρωτοκόλλου.

Χρησιμοποιεί ένα μοντέλο driver-based, το οποίο επιτρέπει την επέκτασή του με πρόσθετες υλοποιήσεις πρωτοκόλλων. Καθώς εισάγονται νέα πρωτόκολλα ή συσκευές, οι προγραμματιστές μπορούν να δημιουργούν και να ενσωματώνουν νέους drivers στο PLC4X, διασφαλίζοντας ότι η βιβλιοθήκη παραμένει ενημερωμένη και ευέλικτη.

Έχει σχεδιαστεί ώστε να μπορεί να χρησιμοποιηθεί από ποικίλλες γλώσσες προγραμματισμού, όπως Java, C++, C# και Python. Αυτή η ευρεία γλωσσική υποστήριξη επιτρέπει στους προγραμματιστές να χρησιμοποιούν το PLC4X με τη γλώσσα προγραμματισμού που προτιμούν, προωθώντας την ευρύτερη υιοθέτηση και ευελιξία.

Χρησιμοποιεί τεχνικές παραγωγής κώδικα (code generation) για τη δημιουργία υλοποιήσεων πρωτοκόλλων. Αυτή η προσέγγιση μειώνει την ποσότητα του απαιτούμενου χειροκίνητου κωδικοποίησης και ελαχιστοποιεί τα ανθρώπινα λάθη. Με τη χρήση ενός συνδυασμού προτύπων και προδιαγραφών πρωτοκόλλου, η διαδικασία δημιουργίας κώδικα συμβάλλει στη διασφάλιση της συνέπειας και της αξιοπιστίας σε διαφορετικές υλοποιήσεις πρωτοκόλλων.

Συνοπτικά, το PLC4X είναι μια βιβλιοθήκη βιομηχανικής επικοινωνίας που απλοποιεί και τυποποιεί την επικοινωνία με προγραμματιζόμενους λογικούς ελεγκτές και συσκευές αυτοματισμού. Παρέχοντας ένα ενοποιημένο API, υποστηρίζοντας πολλαπλά πρωτόκολλα και γλώσσες και χρησιμοποιώντας ένα μοντέλο driver-based, το PLC4X διευκολύνει τους προγραμματιστές να δημιουργούν εφαρμογές που αλληλεπιδρούν με διάφορα PLC χωρίς να χρειάζονται βαθιά γνώση κάθε συγκεκριμένου πρωτοκόλλου. [21]

3 Apache Kafka

Εισαγωγή

Το Apache Kafka είναι ένα κατανεμημένο, κλιμακούμενο, ανθεκτικό και ανεκτικό σε σφάλματα σύστημα ανταλλαγής μηνυμάτων. Ένα σύστημα ανταλλαγής μηνυμάτων είναι υπεύθυνο για τη μεταφορά δεδομένων από τη μία εφαρμογή στην άλλη. Καθώς η ευθύνη της μετάδοσης και ανταλλαγής δεδομένων μετατίθεται στο Kafka, οι εφαρμογές μπορούν να επικεντρωθούν στα δεδομένα και την εργασία που τους έχει ανατεθεί. Αυτό οδηγεί σε μείωση τους κόστους υλοποίησης μιας εφαρμογής, καθώς και αύξηση της αξιοπιστίας της.

Η κατανεμημένη ανταλλαγή μηνυμάτων βασίζεται στην έννοια της αξιόπιστης ουράς μηνυμάτων. Τα μηνύματα τίθενται σε ουρά αναμονής ασύγχρονα μεταξύ των εφαρμογών-πελατών (clients) και του Kafka. Το Kafka ακολουθεί το μοντέλο δημοσίευσης-συνδρομής (publish-subscribe) που επιτρέπει τη μεταφορά δεδομένων γρήγορα και αξιόπιστα μεταξύ κατανεμημένων συστημάτων και εφαρμογών. Εν συντομία, οι εφαρμογές χωρίζονται σε παραγωγούς (producers) και καταναλωτές (consumers). Οι παραγωγοί στέλνουν δεδομένα στο Kafka. Ένας ή περισσότεροι παραγωγοί διαβάζουν αυτά τα δεδομένα από το Kafka και τα επεξεργάζονται/χρησιμοποιούν κατάλληλα. Αυτού του είδους η ασύγχρονη επικοινωνία αφαιρεί ή μετριάζει τις καθυστερήσεις και τις δυσχέρειες που σχετίζονται με την άμεση, σύγχρονη επικοινωνία μεταξύ δύο εφαρμογών.

Το Kafka είναι γραμμένο σε Scala και Java, αναπτύχθηκε αρχικά στο LinkedIn όπου έγινε έργο ανοιχτού κώδικα το 2011. Από το 2012 αποτελεί έργο του οργανισμού Apache. Αποτελεί μια ιδανική λύση για εφαρμογές επεξεργασίας μηνυμάτων μεγάλης κλίμακας. Προκειμένου να συνδεθεί σε εξωτερικά συστήματα για μεταφορά δεδομένων, χρησιμοποιείται το Kafka Connect, ενώ υπάρχει επίσης η βιβλιοθήκη Kafka Streams, που με την οποία δύναται η κατασκευή εφαρμογών

επεξεργασίας δεδομένων σε ζωντανό χρόνο. Ένα σύστημα Kafka συνήθως αποτελείται από διακομιστές (servers) και πελάτες (clients) που επικοινωνούν μέσω ενός πρωτοκόλλου δικτύου TCP υψηλής απόδοσης.

Διακομιστές (servers): Το Kafka εκτελείται ως συστάδα (cluster) ενός ή περισσότερων διακομιστών που μπορεί να καλύπτει πολλαπλά κέντρα δεδομένων ή περιοχές υπολογιστικού νέφους (cloud). Ορισμένοι από αυτούς τους διακομιστές αποτελούν το επίπεδο αποθήκευσης, που ονομάζεται brokers. Άλλοι διακομιστές εκτελούν το Kafka Connect για τη συνεχή εισαγωγή και εξαγωγή δεδομένων ως ροές γεγονότων (event streaming) για την ενσωμάτωση του Kafka με άλλα συστήματά, όπως οι σχεσιακές βάσεις δεδομένων, καθώς και άλλες συστάδες Kafka. Μία συστάδα Kafka είναι εξαιρετικά κλιμακούμενη και ανεκτική σε σφάλματα: αν κάποιος από τους διακομιστές της αποτύχει, οι άλλοι διακομιστές θα αναλάβουν το έργο τους για να διασφαλίσουν συνεχή λειτουργία χωρίς απώλεια δεδομένων.

Πελάτες (Clients): Πρόκειται για προγράμματα που επιτρέπουν την επικοινωνία με το σύστημα Kafka. Ένας πελάτης μπορεί να λειτουργεί είτε ως παραγωγός, είτε ως καταναλωτής δεδομένων. Επιτρέπουν τη δημιουργία κατανεμημένων εφαρμογών με λειτουργίες ανάγνωσης, εγγραφής και επεξεργασίας ροών γεγονότων παράλληλα, με δυνα--τότητες κλιμάκωσης και ανοχής σε σφάλματα, ακόμη και σε περίπτωση προβλημάτων δικτύου ή βλαβών μηχανών. Το Kafka συνοδεύεται από μερικούς τέτοιους πελάτες που περιλαμβάνονται στη βασική έκδοση. Συμπληρωματικά, υπάρχουν δεκάδες πελάτες που παρέχονται και συντηρούνται από την κοινότητα του Kafka. Συνοπτικά, υπάρχουν πελάτες για Java και Scala, για Go, Python, C/C++ και πολλές άλλες γλώσσες προγραμματισμού, καθώς επίσης και REST APIs. [22]

Παραδείγματα χρήσης

Παρακάτω ακολουθούν μερικά συνήθη σενάρια χρήσης για το Apache Kafka.[26]

1. Επεξεργασία δεδομένων σε πραγματικό χρόνο (real-time processing): Ένα από τα πιο συνηθισμένα σενάρια χρήσης του Apache Kafka είναι η επεξεργασία

δεδομένων σε πραγματικό χρόνο. Αυτό περιλαμβάνει τη χρήση του Kafka για τη συλλογή και επεξεργασία μεγάλων ροών δεδομένων σε πραγματικό χρόνο, όπως δεδομένα καταγραφής (log data), δεδομένα αισθητήρων και οικονομικά δεδομένα. Τα δεδομένα αυτά μπορούν στη συνέχεια να χρησιμοποιηθούν για τη δημιουργία αναλύσεων σε πραγματικό χρόνο (real-time analytics), όπως η παρακολούθηση της απόδοσης ενός συστήματος ή ο εντοπισμός τάσεων στη συμπεριφορά των πελατών.

2. Συγκέντρωση αρχείων καταγραφής (log aggregation) : Μια άλλη συνήθης περίπτωση χρήσης του Apache Kafka είναι η συγκέντρωση αρχείων καταγραφής. Αυτό περιλαμβάνει τη χρήση του Kafka για τη συλλογή και συγκέντρωση δεδομένων καταγραφής από πολλαπλές πηγές, όπως διακομιστές, εφαρμογές και συσκευές. Τα συγκεντρωτικά δεδομένα καταγραφής μπορούν στη συνέχεια να χρησιμοποιηθούν για σκοπούς εντοπισμού σφαλμάτων και ελέγχου.
3. Αρχιτεκτονική καθοδηγούμενη από γεγονότα (event-driven architecture): Το Apache Kafka μπορεί να χρησιμοποιηθεί ως η ραχοκοκαλιά μιας αρχιτεκτονικής με γνώμονα τα γεγονότα (events). Αυτό περιλαμβάνει τη χρήση του Kafka για την αποστολή και τη λήψη γεγονότων μεταξύ microservices, επιτρέποντάς τους να επικοινωνούν μεταξύ τους άμεσα. Συνεπώς, δίνεται η δυνατότητα να κατασκευάζονται επεκτάσιμα και ευέλικτα συστήματα που μπορούν να ανταποκρίνονται στις αλλαγές σε πραγματικό χρόνο.
4. Ανάλυση ροών δεδομένων (streaming analytics): Το Apache Kafka μπορεί να χρησιμοποιηθεί για ανάλυση ροών δεδομένων σε πραγματικό χρόνο. Αυτή η δυνατότητα μπορεί να χρησιμοποιηθεί για ευρεία γκάμα εφαρμογών, όπως ο εντοπισμός μοτίβων στη συμπεριφορά των πελατών, η παρακολούθηση της απόδοσης ενός συστήματος ή η ανίχνευση απάτης (fraud detection) σε πραγματικό χρόνο.
5. Παρακολούθηση δραστηριότητας ιστότοπου: Μία ακόμα δυνατότητα που παρέχει το Apache Kafka αποτελεί η παρακολούθηση της δραστηριότητας ιστότοπου σε πραγματικό χρόνο. Περιλαμβάνεται η συλλογή δεδομένων σχετικά με τις αλληλεπιδράσεις των χρηστών, όπως οι προβολές σελίδων και

τα κλικ, και τη χρήση τους για τη δημιουργία αναλύσεων σε πραγματικό χρόνο. Τα δεδομένα μπορούν να χρησιμοποιηθούν για διάφορους σκοπούς, όπως η εξατομίκευση της εμπειρίας του χρήστη, η βελτιστοποίηση της απόδοσης του ιστότοπου ή ο εντοπισμός πιθανών κενών ασφαλείας.

Αρχεία καταγραφής στο Apache Kafka

Η έννοια των αρχείων καταγραφής (log files) είναι κεντρική στην αρχιτεκτονική του Apache Kafka και αποτελεί ένα από τα βασικά χαρακτηριστικά που το καθιστούν ένα δημοφιλές και ισχυρό κατανεμημένο σύστημα ανταλλαγής μηνυμάτων.

Στο Kafka, τα αρχεία καταγραφής χρησιμοποιούνται για την αποθήκευση μηνυμάτων που παράγονται από εκδότες και καταναλώνονται από συνδρομητές. Κάθε αρχείο καταγραφής αντιπροσωπεύει μια ταξινομημένη, αμετάβλητη ακολουθία μηνυμάτων που μπορούν να διαβαστούν και να γραφτούν αποτελεσματικά.

Υπάρχουν διάφοροι λόγοι για τους οποίους η έννοια του αρχείου καταγραφής είναι σημαντική στο Kafka:

- **Ανθεκτικότητα και ανοχή σε σφάλματα:** Επειδή το Kafka γράφει όλα τα μηνύματα στο δίσκο καθώς παράγονται, τα μηνύματα αποθηκεύονται με διάρκεια και μπορούν να ανακτηθούν σε περίπτωση αποτυχίας. Εάν ένας διαμεσολαβητής (Kafka server / broker) αποτύχει, τα αρχεία καταγραφής μπορούν να αναπαραχθούν για να ανακτηθούν τυχόν χαμένα δεδομένα.
- **Υψηλή απόδοση και χαμηλή καθυστέρηση:** Ο σχεδιασμός του Kafka με επίκεντρο το αρχείο καταγραφής του επιτρέπει την επίτευξη επιδόσεων υψηλής απόδοσης και χαμηλής καθυστέρησης. Γράφοντας τα μηνύματα στο δίσκο διαδοχικά, η Kafka μπορεί να βελτιστοποιήσει την είσοδο/έξοδο στο

δίσκο και να ελαχιστοποιήσει τις αναζητήσεις, επιτρέποντάς της να διαχειρίζεται μεγάλο αριθμό μηνυμάτων με χαμηλή καθυστέρηση.

- **Επεκτασιμότητα:** Τα αρχεία καταγραφής του Kafka μπορούν να κατανεμηθούν σε πολλούς διαμεσολαβητές, επιτρέποντάς του να κλιμακώνεται οριζόντια και να διαχειρίζεται μεγάλους όγκους δεδομένων. Με την κατανομή των αρχείων καταγραφής σε πολλαπλούς μεσίτες, η Kafka μπορεί επίσης να διασφαλίσει ότι τα μηνύματα αναπαράγονται και είναι διαθέσιμα ακόμη και αν ένας μεσίτης αποτύχει.
- **Επεξεργασία ροής:** Η έννοια του αρχείου καταγραφής είναι επίσης σημαντική για τις δυνατότητες επεξεργασίας ροής του Kafka. Αντιμετωπίζοντας το αρχείο καταγραφής ως ροή γεγονότων, η Kafka επιτρέπει στους προγραμματιστές να δημιουργούν σωληνώσεις (pipelines) επεξεργασίας που μπορούν να μετασχηματίζουν και να αναλύουν δεδομένα σε πραγματικό χρόνο.

Συνολικά, η έννοια του αρχείου καταγραφής είναι θεμελιώδης για το σχεδιασμό του Kafka και διαδραματίζει κρίσιμο ρόλο στην απόδοση, την επεκτασιμότητα και την ανοχή σε σφάλματα.[24][5]

Έννοιες

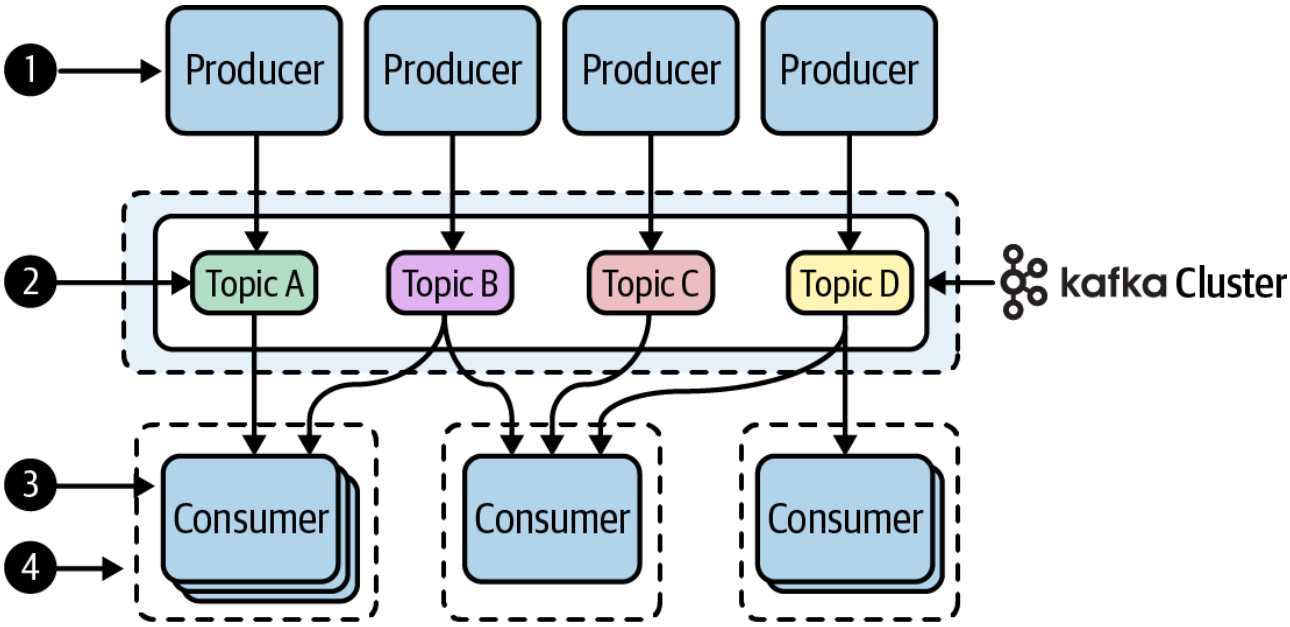
Ένα γεγονός (event) καταγράφει ότι κάτι συνέβη στον κόσμο. Ονομάζεται επίσης εγγραφή (record) ή μήνυμα (message). Τα δεδομένα γράφονται ή διαβάζονται στο Kafka με τη μορφή γεγονότων. Εννοιολογικά, ένα γεγονός έχει κλειδί, τιμή, χρονοσφραγίδα και προαιρετικές επικεφαλίδες (headers) επιπλέον πληροφοριών (metadata) (εικόνα 3-3).

Παραγωγοί (producers) είναι οι εφαρμογές-πελάτες που δημοσιεύουν (γράφουν) γεγονότα στο Kafka και καταναλωτές (consumers) είναι αυτοί που εγγράφονται (διαβάζουν και επεξεργάζονται) σε αυτά τα γεγονότα. Στο Kafka, οι παραγωγοί και οι καταναλωτές είναι πλήρως αποσυνδεδεμένοι και ανεξάρτητοι μεταξύ τους (εικόνα 3-

1). Για παράδειγμα, οι παραγωγοί δεν χρειάζεται ποτέ να περιμένουν τους καταναλωτές. Το Kafka παρέχει διάφορες εγγυήσεις, όπως η δυνατότητα επεξεργασίας γεγονότων ακριβώς μία φορά.

Τα γεγονότα οργανώνονται και αποθηκεύονται σε θέματα (topics). Με απλά λόγια, ένα θέμα θυμίζει μια συρταριέρα αποθήκευσης ρούχων και τα γεγονότα είναι τα ρούχα που τοποθετούμε. Στο 1ο συρτάρι βάζουμε τις κάλτσες, στο 2ο πιτζάμες κ.ο.κ. Με τον ίδιο τρόπο κάθε κατηγορία γεγονότων εισέρχεται σε διαφορετικό θέμα, κάτι που καθορίζεται από του χρήστες. Το κάθε θέμα στο αποθηκεύεται σε ένα αρχείο καταγραφής (log file), το οποίο έχει τα χαρακτηριστικά που αναλύθηκαν στο παραπάνω σχετικό κεφάλαιο.

Τα θέματα είναι πάντα πολλαπλών παραγωγών και πολλαπλών συνδρομητών: ένα θέμα μπορεί να έχει μηδέν, έναν ή πολλούς παραγωγούς που γράφουν γεγονότα σε αυτό, καθώς και μηδέν, έναν ή πολλούς καταναλωτές που εγγράφονται σε αυτά τα γεγονότα.



Εικόνα 3-1: Μοντέλο επικοινωνίας του Apache Kafka. Τα διάφορα συστήματα, αντί να επικοινωνούν απευθείας μεταξύ τους, γράφουν ή διαβάζουν τα δεδομένα τους στο Kafka (1),(3). Συνεπώς, οι παραγωγοί και οι καταναλωτές είναι πλήρως αποσυνδεδεμένοι μεταξύ τους. Τα δεδομέν

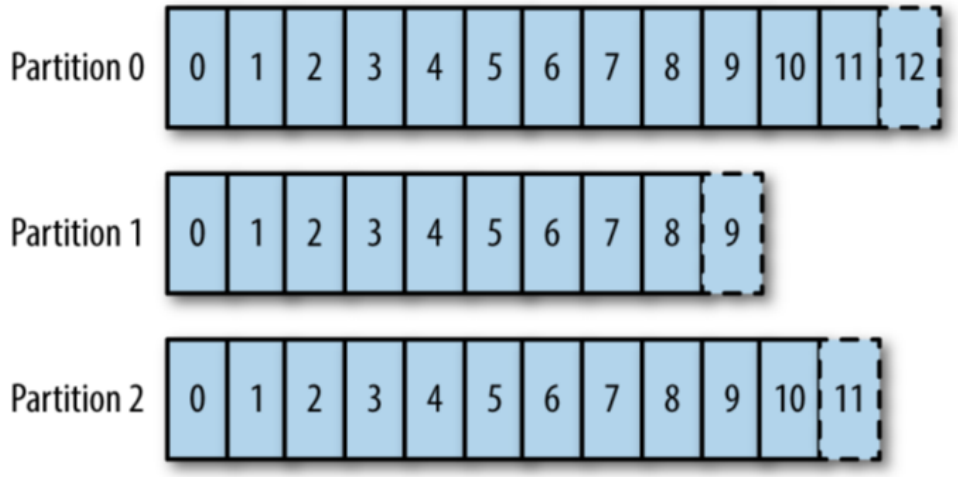
Τα γεγονότα σε ένα θέμα μπορούν να διαβάζονται όσο συχνά χρειάζεται - σε αντίθεση με άλλα συστήματα ανταλλαγής μηνυμάτων, τα γεγονότα δεν

διαγράφονται μετά την κατανάλωση. Αντ' αυτού, ορίζεται για πόσο χρονικό διάστημα το Kafka θα πρέπει να διατηρεί τα γεγονότα, μέσω μιας ρύθμισης ανά θέμα, μετά την οποία οι παλαιότερες εγγραφές θα διαγράφονται (retention). Η απόδοση του Kafka είναι σταθερή σε σχέση με το μέγεθος των δεδομένων, οπότε η αποθήκευση δεδομένων για μεγάλο χρονικό διάστημα είναι αποδεκτή.

Τα θέματα είναι κατανεμημένα, πράγμα που σημαίνει ότι ένα θέμα κατανέμεται σε έναν αριθμό κομματιών που βρίσκονται σε διαφορετικούς διακομιστές του Kafka. Τα κατανεμημένα κομμάτια κάθεται θέματος ονομάζονται κατατμήσεις (partitions). Η κάθε κατάτμηση δεν είναι παρά ένα αρχείο καταγραφής. Αυτή η κατανεμημένη τοποθέτηση των δεδομένων είναι πολύ σημαντική για την επεκτασιμότητα, επειδή επιτρέπει στις εφαρμογές-πελάτες να διαβάζουν και να γράφουν τα δεδομένα από/προς πολλούς brokers ταυτόχρονα. Όταν ένα νέο γεγονός δημοσιεύεται σε ένα θέμα, στην πραγματικότητα προσαρτάται σε μία από τις κατατμήσεις του θέματος. Εάν το κλειδί είναι κενό, τότε η επιλογή της κατάτμησης στην οποία θα γραφτεί το γεγονός γίνεται με χρήση του αλγορίθμου Round-Robin. Αλλιώς, χρησιμοποιείται το υπόλοιπο της ακέραιας διαίρεσης της τιμής της συνάρτησης κατακερματισμού του κλειδιού του γεγονότος με το συνολικό αριθμό των κατατμήσεων ενός θέματος για να αποφανθεί η κατάτμηση στην οποία αυτό θα προσαρτηθεί. Σε ψευδο-κώδικα, είναι η εξής φόρμουλα:

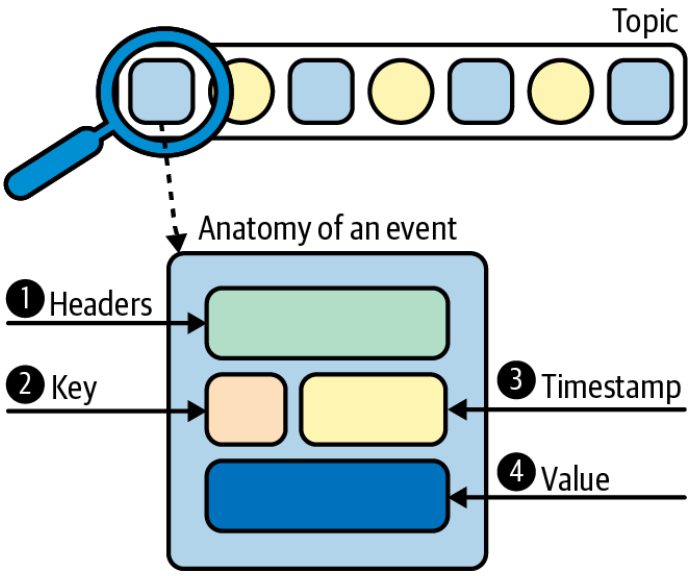
HashCode.(key) % number of partitions [25]

Συνεπώς, τα γεγονότα με το ίδιο κλειδί γεγονότος (π.χ. αναγνωριστικό βιβλίου) εγγράφονται στην ίδια κατάτμηση. Το Kafka εγγυάται ότι οποιοσδήποτε καταναλωτής μιας συγκεκριμένης κατάτμησης ενός θέματος θα διαβάζει πάντα τα γεγονότα αυτής της κατάτμησης με την ίδια ακριβώς σειρά που γράφτηκαν (εικόνα 3-2). [22], [26]



Εικόνα 3-2: Αυτό το παράδειγμα θέματος έχει 3 κατατμήσεις P0-P3. Δύο διαφορετικοί πελάτες-παραγωγοί μπορούν να δημοσιεύουν ανεξάρτητα ο ένας από τον άλλο, νέα γεγονότα στο θέμα γράφοντας μέσω του δικτύου στις κατατμήσεις του θέματος. Γεγονότα με το ίδιο κλειδί γράφονται στην ίδια κατάτμηση. Σημειώστε ότι και οι δύο παραγωγοί μπορούν να γράφουν στην ίδια κατάτμηση, εάν είναι απαραίτητο-.

Κάθε θέμα αντιγράφεται, προκειμένου τα δεδομένα να είναι ανθεκτικά σε σφάλματα και με υψηλή διαθεσιμότητα. Με αυτήν την τεχνική, υπάρχουν πάντα πολλαπλοί διακομιστές που έχουν ένα αντίγραφο των δεδομένων σε περίπτωση που τα πράγματα πάνε στραβά ή χρειάζεται να γίνει συντήρηση στους διακομιστές κ.ο.κ. Μια συνήθης ρύθμιση είναι ένας συντελεστής αντιγραφής 3 (replication strategy), δηλαδή θα υπάρχουν πάντα τρία αντίγραφα των δεδομένων. Αυτή η αντιγραφή πραγματοποιείται σε επίπεδο θέματος-κατατμήσεων (topic-partitions).[19]



Εικόνα 3-3: Τα μέρη που αποτελούν ένα γεγονός στο Apach Kafka. Το 1 αφορά τις επικεφαλίδες, το 2 το κλειδί, το 3 τη χρονοσφραγίδα του γεγονότος, ενώ το 4 την τιμή. Τα γεγονότα γράφονται σε μία κατάτμηση ενός θέματος [27].

Σειριοποίηση - Αποσειριοποίηση

Τα δεδομένα στο Apache Kafka σειριοποιούνται, δηλαδή μετατρέπονται σε δυαδική μορφή, προκειμένου να μεταδοθούν μέσω του δικτύου και να αποθηκευτούν με συμπαγή και αποτελεσματικό τρόπο. Ο λόγος για τη χρήση σειριοποίησης στο Kafka αφορά τη διασφάλιση ότι τα δεδομένα μπορούν να μεταδοθούν (μέσω του δικτύου) καθώς και να αποθηκευτούν αποτελεσματικά. Το Kafka υποστηρίζει διάφορες μορφές σειριοποίησης, όπως η Avro, η JSON και οι Protocol Buffers, μεταξύ άλλων. Η επιλογή της μορφής σειριοποίησης εξαρτάται από την περίπτωση χρήσης και τις απαιτήσεις, όπως ο τύπος των δεδομένων που μεταδίδονται, το μέγεθος των δεδομένων, οι απαιτήσεις επεξεργασίας και η συμβατότητα με άλλα συστήματα. Η σειριοποίηση στην Kafka παίζει καθοριστικό ρόλο στη διασφάλιση της αποτελεσματικής μετάδοσης και αποθήκευσης των δεδομένων. Για παράδειγμα, η σειριοποίηση των δεδομένων σε μια συμπαγή δυαδική μορφή μπορεί να μειώσει την ποσότητα των δεδομένων που πρέπει να μεταδοθούν μέσω του δικτύου, με αποτέλεσμα τη μείωση της καθυστέρησης και τη

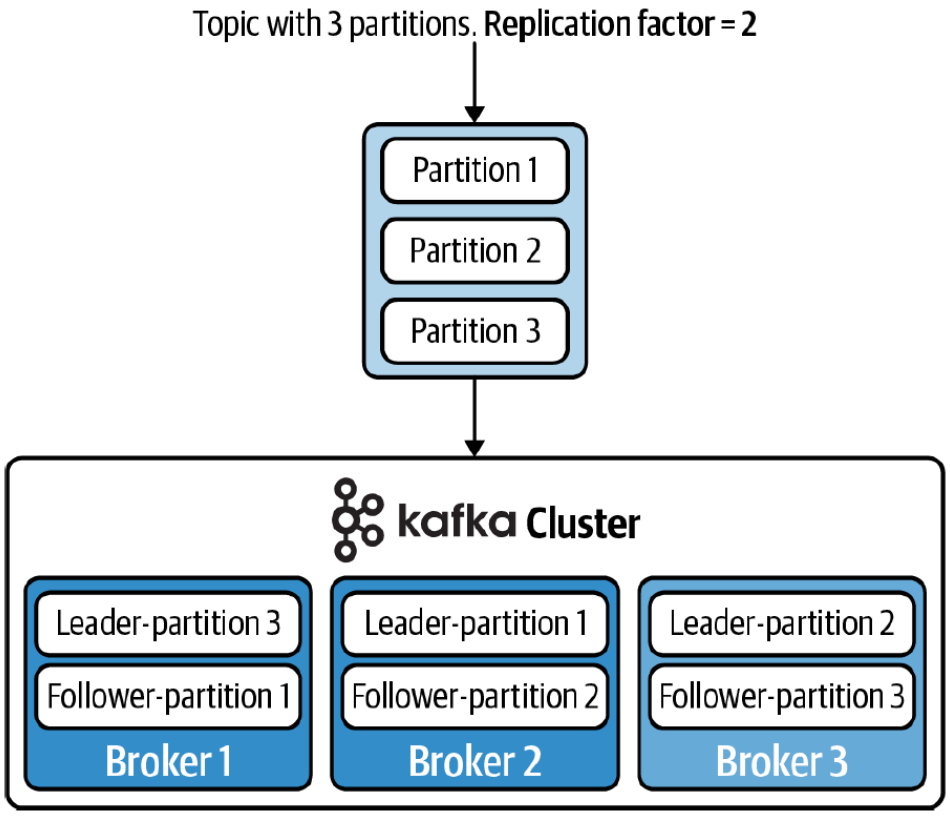
βελτίωση της απόδοσης. Η σειριοποίηση επιτρέπει επίσης την αποθήκευση των δεδομένων σε συμπαγή μορφή, μειώνοντας τον απαιτούμενο αποθηκευτικό χώρο. Εκτός από αυτά τα οφέλη, η σειριοποίηση επιτρέπει επίσης την εύκολη επαναφορά των δεδομένων στην αρχική τους μορφή όταν αυτά ανακτώνται. Αυτό διευκολύνει την επεξεργασία των δεδομένων και την ενσωμάτωσή τους σε άλλα συστήματα.[22]

Αντίγραφα

Η διαχείριση αντιγράφων στο Apache Kafka αναφέρεται στη διαδικασία διατήρησης πολλαπλών αντιγράφων μιας κατάτμησης θέματος σε πολλαπλούς κόμβους διακομιστών για να διασφαλιστεί η αξιοπιστία και η υψηλή διαθεσιμότητα των δεδομένων. Τα αντίγραφα μιας κατάτμησης λειτουργούν ως αντίγραφα ασφαλείας που μπορούν να χρησιμοποιηθούν για την ανάκτηση δεδομένων σε περίπτωση βλάβης του διακομιστή ή άλλων τύπων αστοχιών.

Ο συντελεστής αντιγραφής (replication strategy) είναι μια παράμετρος που καθορίζει τον αριθμό των αντιγράφων που πρέπει να διατηρούνται για κάθε κατάτμηση. Ο συντελεστής αντιγραφής ορίζεται σε επίπεδο θέματος και πρέπει να είναι τουλάχιστον 2 για να διασφαλιστεί ότι υπάρχει τουλάχιστον ένα αντίγραφο διαθέσιμο σε περίπτωση αποτυχίας. Ο συντελεστής αντιγραφής μπορεί να αυξηθεί για την παροχή πρόσθετης ανοχής σφαλμάτων ή για τη βελτίωση των επιδόσεων σε συστήματα μεγάλης κλίμακας.

Σε μια κατάτμηση, ένα από τα αντίγραφα αναλαμβάνει το ρόλο του ηγέτη (leader), ενώ τα υπόλοιπα ορίζονται ως ακόλουθα αντίγραφα. Το αντίγραφο - ηγέτης είναι υπεύθυνο για όλες τις αιτήσεις ανάγνωσης και εγγραφής για την κατάτμηση. Τα ακόλουθα αντίγραφα είναι υπεύθυνα για τη διατήρηση ενός αντιγράφου των δεδομένων της κατάτμησης και τον συγχρονισμό με το αντίγραφο – ηγέτης (εικόνα 3-4).[22]



Εικόνα 3-4: Ένα θέμα Kafka με συντελεστή αντιγραφής (replication factor) ίσο με 2. Για κάθε κατάτμηση (partition) υπάρχει ένα αντίγραφο - ηγέτης (leader partition) και ένα αντίγραφο - ακόλουθος (follower partition) [27]

Τα in-sync replicas είναι τα αντίγραφα που είναι ενημερωμένα με το αντίγραφο – ηγέτης και μπορούν να γίνουν νέοι ηγέτες σε περίπτωση αποτυχίας του τωρινού αντίγραφου - ηγέτη. Τα in-sync αντίγραφα είναι κρίσιμα για τη διατήρηση της διαθεσιμότητας και της αξιοπιστίας των δεδομένων στην κατάτμηση.

Όταν ένας παραγωγός ή καταναλωτής ζητά δεδομένα για μια κατάτμηση, ο διακομιστής που φιλοξενεί το αρχηγικό αντίγραφο για την εν λόγω κατάτμηση είναι υπεύθυνος για την εκπλήρωση του αιτήματος. Η διαδικασία επιλογής διακομιστή είναι διαφανής για τον παραγωγό και τον καταναλωτή και διαχειρίζεται από το σύμπλεγμα διακομιστών Apache Kafka.

Τα αντίγραφα μπορεί να επανατοποθετηθούν από έναν διακομιστή σε έναν άλλο για την εξισορρόπηση του φορτίου, την αύξηση της ανοχής σφαλμάτων ή την εκτέλεση συντήρησης σε έναν διακομιστή. Η επανατοποθέτηση αντιγράφων είναι

μια διαδικασία στο παρασκήνιο που διαχειρίζεται από τη συστάδα διακομιστών Apache Kafka.

Εν κατακλείδι, η διαχείριση αντιγράφων στον Apache Kafka είναι ένα κρίσιμο στοιχείο της πλατφόρμας που εξασφαλίζει την αξιοπιστία και την υψηλή διαθεσιμότητα των δεδομένων. [22], [26]

Παραγωγοί (Producers)

Οι παραγωγοί στέλνουν τα μηνύματα ομαδοποιημένα (batching), προκειμένου να μειωθεί η κίνηση του δικτύου, καθώς και για λόγους που θα συζητηθούν παρακάτω. Για την ομαδοποίηση χρησιμοποιείται μια προσωρινή αποθήκευση (buffer), καθώς και ένας ανώτατος χρόνος αναμονής. Συνεπώς, μια ομάδα μηνυμάτων στέλνεται στον Kafka διακομιστή εάν συμπληρωθεί ο αποθηκευτικός χώρος προσωρινής αποθήκευσης ή ξεπεραστεί ο ανώτατος χρόνος αναμονής (παράμετροι buffer.memory και linger.ms). Συνεπώς, προσθέτωντας λίγη καθυστέρηση στη μετάδοση των μηνυμάτων επιτυγχάνουμε υψηλά μεγαλύτερο ρυθμό μετάδοσης (throughput). Το μέγεθος της ομαδοποίησης μηνυμάτων καθορίζεται από την παράμετρο batch.size.[26]

Για κάθε κατάτμηση ενός θέματος, υπάρχει ένας διακομιστής με το ρόλο του ηγέτη (leader). Οι παραγωγοί στέλνουν μηνύματα στον ηγέτη της κατάτμησης. Επιπλέον, μπορούν να στείλουν τα δεδομένα τοποθετώντας κάποιο κλειδί. Τα μηνύματα με τα ίδια κλειδιά καταλήγουν πάντα στο ίδιο partition. Αυτό επιτρέπει στις εφαρμογές – παραγωγούς να επωφεληθούν από την κατανεμημένη επεξεργασία, τη μείωση της κίνησης του δικτύου, αλλά και την τοπικότητα των δεδομένων. Επιπλέον, μπορούν να εφαρμόζουν λογική στον τρόπο που στέλνουν τα μηνύματα. Σε ένα σύστημα του διαδικτύου των πραγμάτων (Internet of Things), μετρήσεις από πολλαπλές συσκευές θα μπορούσαν να γράφονται στο Kafka με κάποιο μοναδικό αναγνωριστικό συσκευής. Με αυτόν τον τρόπο, κάθε κατάτμηση θα

περιέχει μετρήσεις της ίδιας συσκευής και μπορεί να εφαρμοστεί επεξεργασία ανά συσκευή ή κατηγορία συσκευών. Εάν δεν προσδιοριστεί κλειδί, τότε τα μηνύματα στέλνονται σε κατατμήσεις με κυκλικό τρόπο, με χρήση του αλγορίθμου Round – Robin.

Όταν μια ομάδα μηνυμάτων αποτυγχάνει να φθάσει μέχρι το διακομιστή Kafka, τότε λαμβάνουν χώρα επαναλήψεις της αποστολής της ομάδας. Αυτές καθορίζονται από τον μέγιστο αριθμό επαναλήψεων (παράμετρος retries), καθώς και από το χρόνο αναμονής μεταξύ της επόμενης προσπάθειας αποστολής (παράμετρος *retry.backoff.ms*).[22]

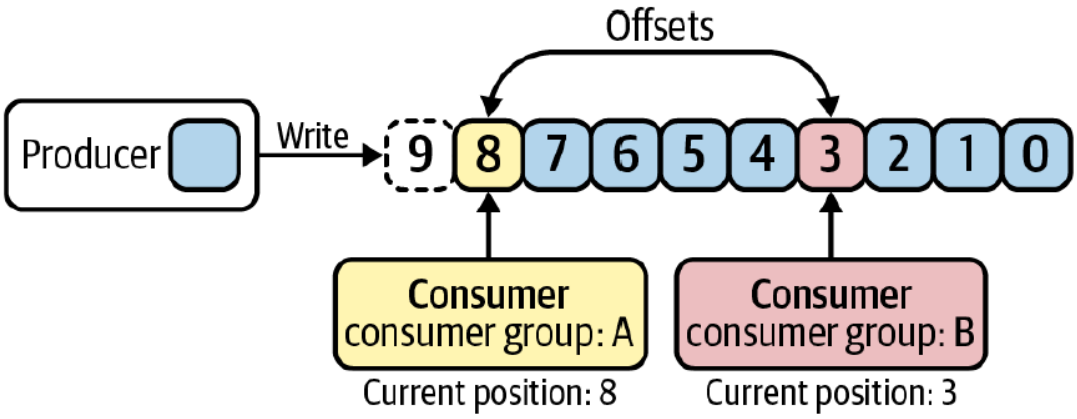
Καταναλωτές (Consumers)

Οι καταναλωτές στο Apache Kafka τραβούν τα δεδομένα (pull) από έναν Kafka διακομιστή. Χρησιμοποιούν και αυτοί ομαδοποίηση των μηνυμάτων και ορίζεται ο ελάχιστος αριθμός από bytes που μπορεί να τραβήξει ένας καταναλωτής (παράμετρος *fetch.min.bytes*). Εάν αυτός ο αριθμός δεν έχει συμπληρωθεί, τότε ο καταναλωτής θα περιμένει.

Οι καταναλωτές στην Kafka ανήκουν σε μια ομάδα καταναλωτών. Μια ομάδα καταναλωτών είναι μια συλλογή καταναλωτών που μοιράζονται το ίδιο αναγνωριστικό ομάδας. Όταν μια ομάδα καταναλωτών επεξεργάζεται ένα θέμα, κάθε κατάτμηση αυτού του θέματος ανατίθεται σε έναν μόνο καταναλωτή της ομάδας. Με αυτόν τον τρόπο, η επεξεργασία των δεδομένων είναι ισορροπημένη μεταξύ των καταναλωτών της ομάδας.

Οι καταναλωτές διατηρούν έναν δρομέα, που ονομάζεται offset, ο οποίος υποδεικνύει τη θέση τους στο θέμα. Το offset αποθηκεύεται σε ένα ειδικό θέμα που ονομάζεται *__consumer_offsets*. Όπως ειπώθηκε και παραπάνω, κάθε κατάτμηση ενός θέματος διαβάζεται από μόνο έναν καταναλωτή μιας ομάδας καταναλωτών. Αυτό σημαίνει πως η κάθε κατάτμηση έχει ένα offset για κάθε ομάδα καταναλωτών που διαβάζουν από αυτό το θέμα. Από κάθε θέμα (και κατά συνέπεια τις κατατμήσεις

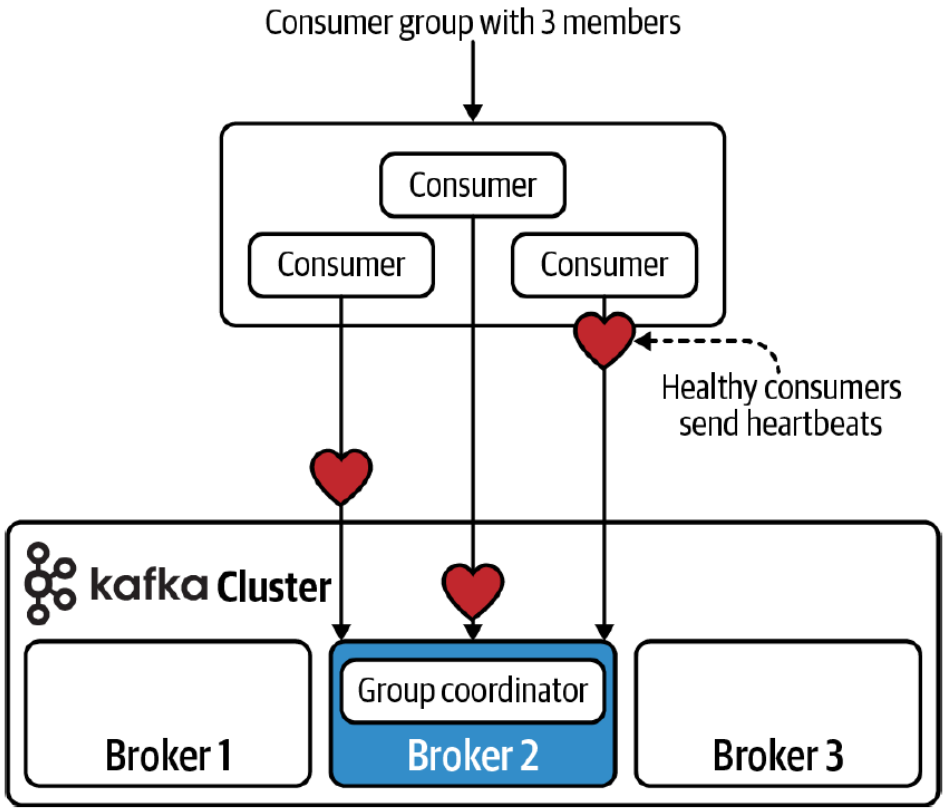
του) μπορούν να διαβάζουν πολλές ομάδες καταναλωτών, όπως φαίνεται και στην εικόνα 3-5. Με αυτό τον τρόπο ελέγχεται η τρέχουσα κατάσταση αναφορικά με το ποιά μηνύματα έχουν καταναλωθεί, ένα πρόβλημα δυσκολότερο στην πράξη από όσο φαίνεται αρχικά. [22]



Εικόνα 3-5: Πολλαπλές ομάδες καταναλωτών μπορούν να διαβάζουν από την ίδια κατάτμηση (partition) ενός θέματος.

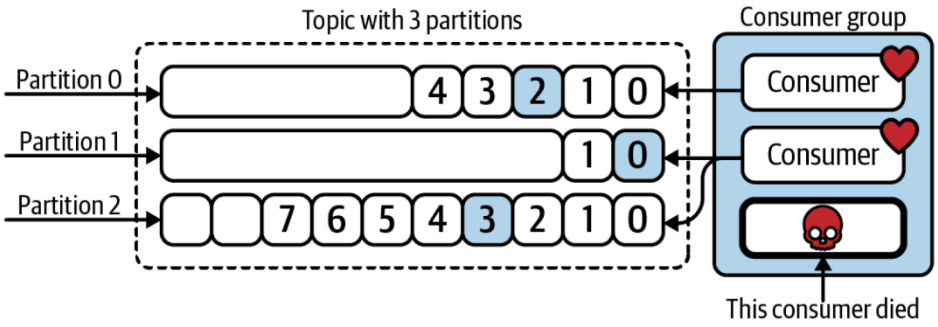
Οι καταναλωτές πραγματοποιούν έλεγχο (polling) στους διακομιστές Kafka για νέα δεδομένα. Το αίτημα ελέγχου αποστέλλεται σε έναν τυχαία επιλεγμένο διακομιστή στη συστάδα υπολογιστών Kafka, ο οποίος ενεργεί ως συντονιστής (coordinator) για την εν λόγω ομάδα καταναλωτών. Στη συνέχεια, ο συντονιστής επιστρέφει τα δεδομένα για τις κατατμήσεις που έχουν ανατεθεί στον καταναλωτή, μαζί με τις τρέχουσες μετατοπίσεις (offsets) τους. Ο καταναλωτής επεξεργάζεται τα δεδομένα και στη συνέχεια ενημερώνει τη μετατόπισή του.[26]

Για να αποφευχθεί η θεώρηση ενός καταναλωτή ως *"νεκρού"* από τον συντονιστή (coordinator), ο καταναλωτής στέλνει περιοδικά σήματα στον συντονιστή, τα οποία ονομάζονται *heartbeats*, όπως φαίνεται και στην εικόνα 3-6. Εάν ο συντονιστής δεν λάβει heartbeat από έναν καταναλωτή για ορισμένο χρονικό διάστημα, θεωρεί τον καταναλωτή νεκρό και αναθέτει εκ νέου τις κατατμήσεις (partitions) του σε άλλους καταναλωτές της ομάδας (εικόνα 3-6).[22]



Εικόνα 3-6: Τρεις καταναλωτές που ανήκουν στην ίδια ομάδα (consumer group). Οι καταναλωτές στέλνουν περιοδικά σήματα στον συντονιστή (coordinator) [27].

Το Kafka χρησιμοποιεί έναν μηχανισμό που ονομάζεται επανεξισορρόπηση (rebalancing) για την κατανομή των κατατμήσεων μεταξύ των καταναλωτών μιας ομάδας. Ο μηχανισμός αυτός ενεργοποιείται κάθε φορά που ένας νέος καταναλωτής προσχωρεί ή αποχωρεί από την ομάδα ή όταν ο συντονιστής αποτυγχάνει και εκλέγεται νέος. Κατά τη διάρκεια μιας επανεξισορρόπησης, οι κατατμήσεις ανακατανέμονται για να εξασφαλιστεί η ομοιόμορφη κατανομή του φόρτου εργασίας (load balancing), όπως φαίνεται και στην εικόνα 3-7.



Εικόνα 3-7: Ένα θέμα Kafka με 3 κατατμήσεις και 3 καταναλωτές. Όταν ένας καταναλωτής αποτυγχάνει, τότε η επεξεργασία της κατάτμησης όπου εκείνος είχε αναλάβει, περνάει στην ευθύνη ενός άλλου καταναλωτή της ίδιας ομάδας καταναλωτών [27].

Οι καταναλωτές μπορούν να επιλέξουν να δημοσιεύουν τις μετατοπίσεις τους περιοδικά ή μόνο μετά την επεξεργασία μιας ομάδας (batch) μηνυμάτων. Αυτό είναι γνωστό ως στρατηγική δημοσιοποίησης (commit strategy). Η περιοδική στρατηγική δημοσιοποίησης είναι χρήσιμη για επεξεργασία με χαμηλότερη καθυστέρηση, ενώ η στρατηγική δημοσιοποίησης ομάδας είναι χρήσιμη για επεξεργασία με υψηλότερη απόδοση (higher-throughput).

Μόλις οι καταναλωτές έχουν τα δεδομένα από τους διακομιστές Kafka, μπορούν να τα επεξεργαστούν σύμφωνα με τις απαιτήσεις τους. Μπορούν να εκτελέσουν λειτουργίες όπως φιλτράρισμα, συνάθροιση ή μετασχηματισμό των δεδομένων προτού τα αποθηκεύσουν σε μια βάση δεδομένων ή τα στείλουν σε άλλο σύστημα.

Εγγυήσεις παράδοσης μηνυμάτων

Η σημασιολογία παράδοσης μηνυμάτων (message delivery semantics) στο Apache Kafka αναφέρεται στις εγγυήσεις που παρέχει το σύστημα για την παράδοση των μηνυμάτων από τους παραγωγούς στους καταναλωτές. Το Kafka παρέχει διάφορα επίπεδα σημασιολογίας παράδοσης μηνυμάτων που έχουν σχεδιαστεί για να καλύπτουν τις ανάγκες διαφορετικών τύπων εφαρμογών.

Ακολουθεί επεξήγηση των σημασιολογιών παράδοσης μηνυμάτων στον Apache Kafka.

Το πολύ μία φορά (0): Αυτή είναι η ασθενέστερη εγγύηση παράδοσης που παρέχει το Kafka. Σε αυτό το σενάριο, τα μηνύματα μπορεί να χαθούν εάν ο καταναλωτής ή ο διακομιστής αποτύχει και μπορεί να προκύψουν διπλότυπα εάν το ίδιο μήνυμα παραδοθεί περισσότερες από μία φορές. Αυτό το επίπεδο σημασιολογίας παράδοσης είναι κατάλληλο για περιπτώσεις χρήσης όπου η απώλεια μεμονωμένων μηνυμάτων δεν είναι κρίσιμη, όπως η συγκέντρωση αρχείων καταγραφής (log aggregation).

Τουλάχιστον μία φορά (1): Σε αυτό το σενάριο, τα μηνύματα εγγυάται ότι θα παραδοθούν στον καταναλωτή τουλάχιστον μία φορά, αλλά ενδέχεται να προκύψουν αντίγραφα εάν το ίδιο μήνυμα παραδοθεί περισσότερες από μία φορές. Αυτό το επίπεδο σημασιολογίας παράδοσης είναι κατάλληλο για περιπτώσεις χρήσης όπου τα διπλότυπα είναι αποδεκτά.

Ακριβώς μία φορά (2): Αυτή είναι η ισχυρότερη εγγύηση παράδοσης που παρέχει το Kafka. Σε αυτό το σενάριο, τα μηνύματα εγγυάται ότι θα παραδοθούν στον καταναλωτή ακριβώς μία φορά και δεν είναι δυνατή η δημιουργία διπλών μηνυμάτων. Αυτό το επίπεδο σημασιολογίας παράδοσης είναι κατάλληλο για περιπτώσεις χρήσης όπου τα διπλότυπα δεν είναι αποδεκτά, όπως οι οικονομικές συναλλαγές.

Για να γίνει καλύτερα αντιληπτός ο τρόπος με τον οποίο υλοποιούνται οι παραπάνω εγγυήσεις παράδοσης, είναι απαραίτητο ο αναγνώστης να έχει μελετήσει τα κεφάλαια «Παραγωγοί» και «Καταναλωτές».

Καθώς το Kafka είναι κατανεμημένο σύστημα, δεν μπορούμε να είμαστε σίγουροι για την παράδοση ενός μηνύματος. Επιπλέον, δε μπορούμε να είμαστε σίγουροι πως ο διακομιστής ή οι πελάτες θα παραμείνουν ενεργοί. Στην πραγματικότητα, μπορεί να αποτύχουν οποιαδήποτε στιγμή.

Από πλευράς παραγωγού, κάθε μήνυμα που στέλνεται μπορεί να αποτύχει. Εάν αποτύχει, τότε ο παραγωγός το ξαναστέλνει. Μπορεί όμως και να αποτύχει να λάβει μια απάντηση. Σε αυτήν την περίπτωση μπορεί το μήνυμα να μην λήφθηκε, μπορεί

όμως να λήφθηκε, να γράφτηκε κανονικά στο αρχείο καταγραφής, όμως να μην έφτασε το μήνυμα αποδοχής πίσω στον παραγωγό. Σε αυτήν την περίπτωση ο παραγωγός πρέπει να ξαναστείλει το μήνυμα. Αυτό αντιστοιχεί στη σημασιολογία τουλάχιστον-ένα (at least once).

Αν ένας καταναλωτής καταρρεύσει, τότε πιθανόν ένας άλλος να πάρει τη θέση του, διαβάζοντας το offset προκειμένου να συνεχίσει. Θα πρέπει να καθοριστεί από ποια θέση πρέπει να ξεκινήσει την επεξεργασία των μηνυμάτων. Οι ενέργειες που μπορεί να λάβει είναι οι εξής:

1. Μπορεί να διαβάσει τα μηνύματα, να αποθηκεύσει το offset του στο αρχείο καταγραφής και, τέλος, να επεξεργαστεί τα μηνύματα. Σε αυτή την περίπτωση μπορεί ο καταναλωτής να καταρρεύσει μετά την αποθήκευση του offset του αλλά πριν επεξεργαστεί τα μηνύματα και σώσει το αποτέλεσμα στο δίσκο. Δηλαδή η διεργασία που ανέλαβε την επεξεργασία θα ξεκινήσει από το συγκεκριμένο offset, παρόλο που μερικά μηνύματα πριν από αυτή τη θέση δεν είχαν υποστεί επεξεργασία. Αυτό αντιστοιχεί στη σημασιολογία *τουλάχιστον-ένα (at-most-once)*, καθώς σε περίπτωση αποτυχίας του καταναλωτή τα μηνύματα μπορεί να μην έχουν επεξεργαστεί.
2. Μπορεί να διαβάσει τα μηνύματα, να επεξεργαστεί τα μηνύματα και τέλος να αποθηκεύσει τον offset του. Σε αυτή την περίπτωση μπορεί ο καταναλωτής να καταρρεύσει μετά την επεξεργασία των μηνυμάτων αλλά πριν αποθηκεύσει τον offset. Σε αυτή την περίπτωση, όταν η νέα διεργασία αναλάβει τη θέση της, τα πρώτα μηνύματα που λαμβάνει θα έχουν ήδη υποστεί επεξεργασία. Αυτό αντιστοιχεί στη σημασιολογία *τουλάχιστον-μία-φορά (at-least-once)* στην περίπτωση αποτυχίας του καταναλωτή. Σε πολλές περιπτώσεις, τα μηνύματα έχουν ένα πρωτεύον κλειδί και έτσι οι ενημερώσεις είναι ιδεοληπτικές (idempotent), δηλαδή η λήψη του ίδιου μηνύματος δύο φορές αντικαθιστά μια εγγραφή με ένα άλλο αντίγραφο του εαυτού της.

Για την επίτευξη της σημασιολογίας παράδοσης ακριβώς μία φορά, η Kafka χρησιμοποιεί έναν συνδυασμό σημασιολογίας ιδεοληπτικού παραγωγού (idempotent producer) και σημασιολογίας συναλλαγής (transactional schematics). Η

σημασιολογία ιδεοληπτικού παραγωγού εξασφαλίζει ότι τα μηνύματα παραδίδονται με τη σειρά με την οποία στάλθηκαν. Σε κάθε παραγωγό ανατίθεται ένα μοναδικό αναγνωριστικό παραγωγού (id), και έτσι εάν ένα μήνυμα έχει σταλθεί πολλές φορές, τότε εγγράφεται μόνο μία στο αρχείο καταγραφής. Ιδεοληπτικός παραγωγός σημαίνει επίσης ότι μπορεί να στείλει μηνύματα σε πολλά θέματα. Η σημασιολογία συναλλαγής εξασφαλίζει ότι τα μηνύματα είτε παραδίδονται όλα είτε κανένα, προσομοιάζοντας συναλλαγές βάσεων δεδομένων. Στη μεριά του καταναλωτή, χρησιμοποιείται το ειδικό θέμα όπου αποθηκεύει τα offset κάθε καταναλωτή. Το offset περιλαμβάνεται στο μήνυμα-συναλλαγής που στέλνει ο ιδεοληπτικός παραγωγός. Εάν το μήνυμα-συναλλαγή απορριφθεί, τότε ο παραγωγός δε μπορεί να διαβάσει το νέο μήνυμα. Με αυτή τη συνεργασία παραγωγού-καταναλωτή επιτυγχάνεται η σημασιολογία *ακριβώς-μία-φορά.*

Κλείνοντας, η επιλογή της σημασιολογίας παράδοσης είναι ιδιαίτερα σημαντική για κάθε εφαρμογή που αλληλεπιδρά με το Kafka και εξαρτάται από την περίπτωση χρήσης και τις απαιτήσεις για παράδοση και χειρισμό διπλών μηνυμάτων. [22], [26], [28]

Αποθήκευση μηνυμάτων

Το Kafka χρησιμοποιεί ως μέσο αποθήκευσης το σκληρό δίσκο ενός υπολογιστή. Στις επόμενες παραγράφους θα καταστούν σαφής οι λόγοι οι οποίοι οδήγησαν σε αυτήν την απόφαση.

Σε σύγχρονους δίσκους τύπου SSD (Solid State Drive – Δίσκος Στερεάς Κατάστασης) οι γραμμικές εγγραφές είναι πολλές τάξεις ταχύτερες από τις τυχαίες εγγραφές (χιλιάδες φορές ταχύτερες). Προκειμένου να αντισταθμιστεί αυτή η δραματική απόκλιση, τα λειτουργικά συστήματα αναθέτουν σε κομμάτια της κύριας μνήμης που είναι άδεια το ρόλο της προσωρινής αποθήκευσης για δεδομένα του δίσκου (pagecache). Με χρήση της προαναφερθείσας τεχνικής, το λειτουργικό

σύστημα καταφέρνει να βελτιώσει την απόδοση των τυχαίων εγγραφών και τυχαίων αναγνώσεων, καθώς πλέον χρησιμοποιείται μεγάλο κομμάτι της μνήμης. Το επίπεδο αυτό προσωρινής αποθήκευσης για δεδομένα του δίσκου γίνεται αυτόματα μέσω του λειτουργικού συστήματος και εγείρει ένα νέο πρόβλημα, τη διπλή αντιγραφή δεδομένων για εφαρμογών που διατηρούν μια δική τους προσωρινή αποθήκευση. Πρακτικά, η κύρια μνήμη θα έχει τα ίδια δεδομένα δύο φορές. Επιπλέον, το Kafka αναπτύσσεται με τη γλώσσα προγραμματισμού Java. Τα προγράμματα σε Java καταναλώνουν αρκετή μνήμη κατά τη διάρκεια της λειτουργίας τους (περισσότερη συγκριτικά με γλώσσες προγραμματισμού χαμηλότερου επιπέδου όπως είναι οι C, C++, Rust, Zig, Nim κ.α.). Αναφέρεται εδώ η μεγαλύτερες απαιτήσεις της Java σε μνήμη για να τονιστεί το πρόβλημα της διπλής αντιγραφής δεδομένων.

Για τους παραπάνω λόγους, το Kafka σχεδιάστηκε προκειμένου να μην χρησιμοποιεί καθόλου προσωρινή αποθήκευση ως εφαρμογή. Όλα τα δεδομένα γράφονται στο σκληρό δίσκο. Το λειτουργικό σύστημα μέσω του συστήματος pagecache ομαδοποιεί τα δεδομένα και τα τοποθετεί σε κομμάτια (blocks) στην κύρια μνήμη. Αυτή η τεχνική οδηγεί σε πλήρη αξιοποίηση της κύριας μνήμης χωρίς την περιττή σπατάλη της αντιγραφής δεδομένων. Το λειτουργικό σύστημα βελτιστοποιεί τα μοτίβα πρόσβασης στη μνήμη (όπου στο Kafka έχουν σχεδιαστεί να είναι κατά βάση γραμμικά και όχι τυχαία), εφαρμόζοντας τεχνικές πρόωρης τοποθέτησης δεδομένων στη μνήμη (pre-fetching).

Τα οφέλη είναι πολλαπλά. Η υλοποίηση του συστήματος Kafka απλοποιείται, καθώς είναι πλέον ευθύνη του λειτουργικού συστήματος η σωστή τοποθέτηση κομματιών στην κύρια μνήμη. Επιπλέον, αξιοποιούνται στο έπακρο οι διαθέσιμοι πόροι του συστήματος, όσον αφορά τη μνήμη. Τέλος, η απόδοση του συστήματος αυξάνεται, καθώς γίνεται κυρίως χρήση της μνήμης για ανάγνωση, με τη μορφή προσωρινής αποθήκευσης.

Εγγραφή και ανάγνωση μηνυμάτων

Το Apache Kafka χρησιμοποιεί απλές αναγνώσεις και προσαρτήσεις σε αρχεία. Αυτή η μεθοδολογία έχει το πλεονέκτημα ότι όλες οι λειτουργίες είναι O(1) και οι αναγνώσεις δεν μπλοκάρουν τις εγγραφές ή η μία την άλλη. Αυτό έχει προφανή πλεονεκτήματα απόδοσης, καθώς η απόδοση είναι πλήρως αποσυνδεδεμένη από το μέγεθος των δεδομένων. Ένας διακομιστής με φτηνό υλικό μπορεί να έχει αξιόλογη απόδοση και να επεξεργάζεται μεγάλο όγκο δεδομένων. Εάν ο όγκος των δεδομένων είναι πολύ μεγάλος, μπορεί απλά να προσαρτηθούν περισσότεροι σκληροί δίσκοι, δίχως να υπάρχει μείωση της απόδοσης.

Επιπροσθέτως, προκειμένου να αυξηθεί περαιτέρω η απόδοση, χρησιμοποιείται η τεχνική της ομαδοποίησης των μηνυμάτων (batching) κατά την επικοινωνία μεταξύ των συστημάτων των παραγωγών, των διακομιστών και των καταναλωτών. Αυτή η τεχνική προσφέρει μια πληθώρα από πλεονεκτήματα. Επιτρέπει στο Kafka να έχει μεγαλύτερες γραμμικές προσβάσεις στο σκληρό δίσκο και στο λειτουργικό σύστημα να ενισχύσει τις τεχνικές βελτιστοποίησης που συζητήθηκαν παραπάνω, όπως είναι η τοποθέτηση δεδομένων στην κύρια μνήμη (caching – pagecache). Επιπλέον η ομαδοποίηση μηνυμάτων μειώνει την κίνηση του δικτύου, πράγμα που είναι από τις σημαντικότερες βελτιστοποιήσεις σε ένα κατανεμημένο σύστημα. Συνδυαστικά με την ομαδοποίηση των δεδομένων, χρησιμοποιείται συμπίεση των δεδομένων, μειώνοντας περαιτέρω τον όγκο των δεδομένων που πρόκειται να αποσταλούν. Η ομαδοποίηση επιτρέπει μεγαλύτερη συμπίεση και εξοικονόμηση υπολογιστικών πόρων του επεξεργαστή.

Για την ταχύτερη αποστολή δεδομένων στο δίκτυο, χρησιμοποιείται επίσης η κλήση συστήματος sendfile(). Πρόκειται για κλήση του λειτουργικού συστήματος Linux. Στο Linux, πριν το λειτουργικό σύστημα στείλει δεδομένα μέσω του δικτύου, λαμβάνει χώρα μια περίπλοκη διαδικασία. Κατά τη διάρκεια αυτής, τα δεδομένα αντιγράφονται τέσσερις φορές, ενώ πραγματοποιούνται δύο κλήσεις συστήματος. Με χρήση της κλήσης sendfile(), παρακάμπτεται αυτή η διαδικασία και τα δεδομένα στέλνονται άμεσα από το χώρο προσωρινής αποθήκευσης δίσκου του λειτουργικού (pagechache) προς το δίκτυο, διχώς περιττές αντιγραφές. Συνεπώς, πρόκειται για μια

σημαντικότατη βελτιστοποίηση που μειώνει το δεσμευμένο χώρο στη μνήμη και το χρόνο αποστολής δεδομένων. Η διαδικασία που περιγράφεται παραπάνω λαμβάνει χώρα σε ταχύτητα τάξεις μεγέθους μικρότερη συγκριτικά με την ταχύτητα μετάδοσης του δικτύου, η οποία αποτελεί και το σημαντικότερο λόγο καθυστέρησης για την ανάγνωση αυτών.

Συμπίεση αρχείων καταγραφής (Log Compaction)

Η συμπίεση αρχείων καταγραφής στον Apache Kafka είναι ένας μηχανισμός για τη μείωση του όγκου των δεδομένων που αποθηκεύονται σε μια κατάτμηση θέματος διατηρώντας μόνο την τελευταία έκδοση κάθε κλειδιού στην κατάτμηση. Ο στόχος της συμπίεσης των αρχείων καταγραφής είναι η ελαχιστοποίηση του όγκου του αποθηκευτικού χώρου που απαιτείται για την αποθήκευση των δεδομένων, διατηρώντας παράλληλα τη δυνατότητα ανάκτησης της πιο πρόσφατης κατάστασης των δεδομένων σε περίπτωση αποτυχίας.

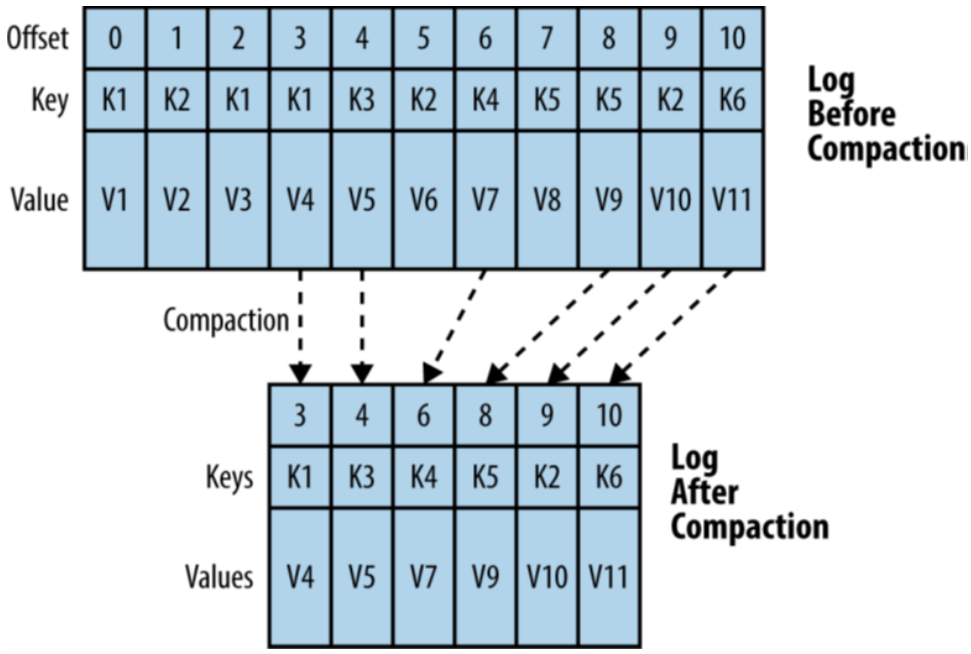
Η συμπίεση των αρχείων καταγραφής λειτουργεί με την παρακολούθηση της πιο πρόσφατης έκδοσης κάθε κλειδιού στην κατάτμηση και την αφαίρεση όλων των προηγούμενων εκδόσεων του κλειδιού. Η πιο πρόσφατη έκδοση του κλειδιού καθορίζεται από τη σειρά με την οποία οι εγγραφές εγγράφονται στην κατάτμηση, με την πιο πρόσφατη έκδοση να αντικαθιστά τις προηγούμενες εκδόσεις του ίδιου κλειδιού. Αυτό σημαίνει ότι οι παλαιότερες τιμές διαγράφονται. Η συμπύκνωση αρχείων καταγραφής ενεργοποιείται σε επίπεδο θέματος θέτοντας την τιμή της παραμέτρου *cleanup.policy* σε compact. Η συχνότητα της συμπίεσης μπορεί να ελεγχθεί με τη ρύθμιση της παραμέτρου *compact.interval.ms*, η οποία καθορίζει πόσο συχνά εκτελείται η διαδικασία συμπίεσης του αρχείου καταγραφής.

Η συμπίεση αρχείων καταγραφής είναι ιδιαίτερα χρήσιμη για περιπτώσεις χρήσης όπου η τελευταία κατάσταση των δεδομένων είναι πιο σημαντική από τα ιστορικά δεδομένα, όπως για την αποθήκευση των πιο πρόσφατων ρυθμίσεων

διαμόρφωσης, προφίλ πελατών ή καταστάσεων συσκευών. Η συμπίεση αρχείων καταγραφής μπορεί επίσης να χρησιμοποιηθεί για την αποθήκευση δεδομένων με υψηλό ρυθμό αλλαγής, όπου τα δεδομένα ενημερώνονται συνεχώς και οι παλαιότερες εκδόσεις δεν έχουν πλέον σημασία.

Η συμπίεση αρχείων καταγραφής μπορεί να έχει αντίκτυπο στην απόδοση, καθώς η διαδικασία συμπίεσης των δεδομένων δεσμεύει πρόσθετους πόρους επεξεργασίας και εισόδου/εξόδου (φορτίο IO). Ωστόσο, ο αντίκτυπος στην απόδοση είναι γενικά μικρός σε σύγκριση με τα οφέλη από τη μείωση του όγκου του αποθηκευτικού χώρου.

Η συμπίεση αρχείων καταγραφής εγγυάται ότι η τελευταία έκδοση κάθε κλειδιού στην κατάτμηση διατηρείται, ενώ όλες οι προηγούμενες εκδόσεις του ίδιου κλειδιού διαγράφονται. Εγγυάται επίσης ότι η σειρά των εγγραφών με το ίδιο κλειδί διατηρείται στην κατάτμηση μετά τη συμπίεση, ότι οποιοσδήποτε καταναλωτής που παραμένει στην κεφαλή του αρχείου καταγραφής θα βλέπει κάθε μήνυμα που γράφεται, ότι η μετατόπιση (offset) ενός μηνύματος δε θα αλλάξει, καθώς και ότι κάθε καταναλωτής που διαβάζει το αρχείο καταγραφής από την αρχή θα δει τουλάχιστον την τελική κατάσταση όλων των εγγραφών με τη σειρά που γράφτηκαν (εικόνα 3-8).



Εικόνα 3-8: Το αρχείο καταγραφής πριν και μετά τη συμπίεση. Διατηρείται μόνο η τελευταία ενημέρωση για κάθε κλειδί.[5]

Συμπερασματικά, η συμπίεση αρχείων καταγραφής στο Apache Kafka είναι ένας μηχανισμός χρήσιμη για περιπτώσεις χρήσης όπου η τελευταία κατάσταση των δεδομένων είναι πιο σημαντική από τα ιστορικά δεδομένα και μπορεί να έχει θετικό αντίκτυπο στις απαιτήσεις απόδοσης και αποθήκευσης των συστημάτων Apache Kafka. [5], [22]

Περιορισμός χρήσης πόρων

Οι περιορισμοί πόρων έχουν σχεδιαστεί για να διασφαλίζουν ότι οι συστάδες Kafka μπορούν να συνεχίσουν να λειτουργούν αποτελεσματικά και χωρίς διακοπές, ακόμη και σε περιπτώσεις όπου ορισμένοι πελάτες παράγουν περισσότερη κίνηση από άλλους. Μπορούν επίσης να βοηθήσουν στην αποτροπή της μονοπώλησης των υπολογιστικών δυνατοτήτων της συστάδας από μεμονωμένους πελάτες, γεγονός που

θα μπορούσε να επηρεάσει την απόδοση και τη διαθεσιμότητα άλλων πελατών. Μπορούν να εφαρμοστούν σε συγκεκριμένους πελάτες ή σε ομάδες πελατών. Εάν υπάρχουν πολλαπλές πολιτικές περιορισμού πόρων, τότε εφαρμόζεται η πιο συγκεκριμένη (αυτή που αφορά έναν συγκεκριμένο πελάτη υπερτερεί έναντι μιας άλλης που αφορά ομάδα πελατών).

Στους παραγωγούς μπορεί να εφαμορστεί μια πολιτική περιορισμού του αριθμού των μηνυμάτων που μπορεί να στείλει ένας παραγωγός σε μια συστάδα Kafka σε μια δεδομένη χρονική περίοδο. Αυτό συμβάλλει στην αποτροπή της υπερφόρτωσης της συστάδας από έναν μεμονωμένο παραγωγό με πάρα πολλά μηνύματα. Αντίστοιχα, στους καταναλωτές μπορεί να εφαρμοστεί μια πολιτική η οποία περιορίζει τον όγκο των δεδομένων που μπορεί να αντλήσει ένας καταναλωτής από μια συστάδα Kafka σε μια δεδομένη χρονική περίοδο. Αυτό συμβάλλει στην αποτροπή ενός μεμονωμένου καταναλωτή από το να καταναλώνει υπερβολικά πολλά δεδομένα.

Περιορισμοί πόρων μπορούν όμως να ρυθμιστούν και για τα αιτήματα που στέλνει ένας πελάτης (παραγωγός ή καταναλωτής). Αυτοί περιορίζουν τον αριθμό των αιτήσεων που μπορεί να κάνει ένας πελάτης σε μια συστάδα Kafka σε μια δεδομένη χρονική περίοδο. Με αυτό τον τρόπο, ενισχύεται η προστασία του συστήματος από υπερβολική κίνηση.

Για κάθε μοναδική ομάδα πελατών, υπάρχουν προεπιλεγμένες ρυθμίσεις σχετικά με τις πολιτικές περιορισμού των πόρων. Οι πολιτικές ρυθμίζονται σε επίπεδο διακομιστή Kafka (broker).

Όταν ξεπεραστούν οι περιορισμοί, τότε ο Kafka διακομιστής σταματά την μεταφορά δεδομένων από ή προς τον πελάτη, εισάγοντας μια καθυστέρηση. Η χρόνος διακοπής εξαρτάται από τις ρυθμίσεις της εκάστοτε πολιτικής περιορισμού.

Kafka Connect

Το Kafka Connect είναι ένα εργαλείο του οικοσυστήματος Apache Kafka που επιτρέπει την εύκολη και επεκτάσιμη ενσωμάτωση δεδομένων από διάφορα εξωτερικά συστήματα. Πρόκειται για ένα κατανεμημένο, κλιμακούμενο και ανεκτικό σε σφάλματα σύστημα σχεδιασμένο για δεδομένα ροής. Το Kafka Connect παρέχει ένα πλαίσιο για τη σύνδεση του Kafka με εξωτερικά συστήματα, όπως βάσεις δεδομένων, ουρές μηνυμάτων, λίμνες δεδομένων (data lakes) και αποθήκες δεδομένων (data warehouse). Μπορεί να χρησιμοποιηθεί για τη μετακίνηση δεδομένων από και προς το Kafka.

Το Kafka Connect αποτελείται από δύο κύρια στοιχεία: τους διαμεσολαβητές (connectors) και τους εργάτες (workers). Οι διαμεσολαβητές είναι οι μονάδες που είναι υπεύθυνες για την εισροή και την εκροή δεδομένων από και προς το Kafka. Είναι συγκεκριμένοι για το σύστημα πηγής ή προορισμού με το οποίο έχουν σχεδιαστεί να συνεργάζονται. Το Kafka Connect περιλαμβάνει έναν αριθμό ενσωματωμένων διαμεσολαβητών για συνήθης πηγές/εκροές δεδομένων, όπως JDBC, Elasticsearch, HDFS, AWS S3 και άλλα. Οι χρήστες μπορούν επίσης να δημιουργήσουν τους δικούς τους προσαρμοσμένους διαμεσολαβητές για την εισαγωγή και εξαγωγή δεδομένων από τα δικά τους συστήματα.

Οι διαμεσολαβητές μπορούν να εκτελούνται είτε σε αυτόνομη είτε σε κατανεμημένη λειτουργία. Στην αυτόνομη λειτουργία, ο σύνδεσμος εκτελείται σε ένα μόνο μηχάνημα ως αυτόνομη διεργασία, ενώ στην κατανεμημένη λειτουργία, ο σύνδεσμος εκτελείται ως μέρος μιας συστάδας (cluster) Kafka Connect με διεργασίες που εκτελούνται παράλληλα. Η κατανεμημένη λειτουργία παρέχει επεκτασιμότητα και ανοχή σε σφάλματα.

Οι εργάτες είναι υπεύθυνοι για τη διανομή και τη διαχείριση των διαμεσολαβητών σε όλη τη συστάδα, το συντονισμό της εισόδου και εξόδου δεδομένων και τη διαχείριση της συνολικής υγείας και απόδοσης του συστήματος. Οι εργάτες τυπικά εκτελούνται ως μέρος μιας συστάδας Kafka Connect, με πολλαπλούς

εργάτες που εκτελούνται παράλληλα, και μπορούν να κλιμακώνονται οριζόντια ανάλογα με τις ανάγκες για την αντιμετώπιση των αυξανόμενων φορτίων.

Το Kafka Connect παρέχει επίσης ένα REST API για τη διαχείριση των διαμεσολαβητών, των εργαζομένων και του συνολικού συστήματος. Οι χρήστες μπορούν να χρησιμοποιούν αυτό το API για την αλληλεπίδραση με το Kafka Connect.

Έχει σχεδιαστεί για να είναι εξαιρετικά επεκτάσιμο και ανεκτικό σε σφάλματα, καθιστώντας το κατάλληλο για χρήση σε περιβάλλοντα επεξεργασίας δεδομένων μεγάλης κλίμακας. Χρησιμοποιείται σε μια ποικιλία εφαρμογών, συμπεριλαμβανομένων της διαχείρισης δεδομένων ροής και των αναλύσεων σε πραγματικό χρόνο.

4 Βασικές αρχές επεξεργασίας ροής

Η *επεξεργασία ροής γεγονότων (Event Streaming Processing, Stream Processing)* είναι η πρακτική της ανάληψης δράσης σε μια συνεχόμενη σειρά δεδομένων που προέρχονται από ένα σύστημα που δημιουργεί δεδομένα ακατάπαυστα. Η *επεξεργασία ροής γεγονότων* είναι απαραίτητη για περιπτώσεις στις οποίες είναι ανάγκη να αναληφθεί δράση άμεσα. Αυτός είναι ο λόγος για τον οποίο η *επεξεργασία ροής γεγονότων* συχνά περιγράφονται ως *επεξεργασία σε πραγματικό χρόνο (real-time processing)*.Ο όρος *γεγονός (event)* αναφέρεται σε κάθε σημείο δεδομένων στο σύστημα και ο όρος *ροή (stream)* αναφέρεται στη συνεχή παράδοση αυτών των γεγονότων. Μια σειρά γεγονότων μπορεί επίσης να αναφέρεται ως *ροή δεδομένων (streaming data, data stream)*. Οι ενέργειες που πραγματοποιούνται σε αυτά τα γεγονότα περιλαμβάνουν:

- Συναθροίσεις (aggregations), όπως για παράδειγμα άθροισμα, υπολογισμός μέσου όρου, ή τυπικής απόκλισης
- Αναλύσεις (analytics), όπως πρόβλεψη ενός μελλοντικού γεγονότος με βάση μοτίβα στα δεδομένα
- Μετασχηματισμούς (transformations), παραδείγματος χάριν αλλαγή ενός αριθμού σε μορφή ημερομηνίας
- Εμπλουτισμό (enrichment), όπως για παράδειγμα συνδυασμός του σημείου δεδομένων με άλλες πηγές δεδομένων για τη δημιουργία περισσότερου πλαισίου και νοήματος)
- Εισαγωγή δεδομένων (ingestion), όπως για παράδειγμα εισαγωγή των δεδομένων σε μια βάση δεδομένων

Η επεξεργασία ροής γεγονότων αντί να αντιμετωπίζει τα δεδομένα ως ένα ολόκληρο σύνολο, επεξεργάζεται ένα μικρό κομμάτι κάθε φορά. Από τα παραπάνω γίνεται σαφές πως τα κλασσικά εργαλεία ανάπτυξης λογισμικού δεν είναι αρκετά και απαιτείται ένα εξειδικευμένο σύνολο τεχνολογιών, ικανό να ανταποκριθεί σε αυτές τις προκλήσεις.[24], [29]

Η έννοια του χρόνου στην επεξεργασία ροής γεγονότων

Σε αυτήν την μορφή επεξεργασίας ροής γεγονότων, ο χρόνος είναι καθοριστικής σημασίας για την επεξεργασία. Αυτό μπορεί να συμβαίνει στις αναλύσεις χρονοσειρών (timeseries), στις αθροίσεις με βάση συγκεκριμένες χρονικές περιόδους (που συνήθως ονομάζονται χρονικά παράθυρα ή σκέτο παράθυρα) ή όταν πραγματοποιείται επεξεργασία γεγονότων όπου ο χρόνος που συνέβη ένα γεγονός είναι σημαντικός.

Όταν αναφέρεται ο χρόνος σε ένα πρόγραμμα ροής (για παράδειγμα για στον ορισμό παραθύρων), γίνεται αναφορά σε διαφορετικές έννοιες του χρόνου, τον χρόνο επεξεργασίας και τον χρόνο γεγονότος.[30], [31]

Χρόνος επεξεργασίας

Ο χρόνος επεξεργασίας (processing time) αναφέρεται στον χρόνο συστήματος του υπολογιστή που εκτελεί την επεξεργασία.

Όταν ένα πρόγραμμα επεξεργασίας ροής γεγονότων εκτελείται με γνώμονα το χρόνο επεξεργασίας, όλες οι λειτουργίες που βασίζονται στον χρόνο (όπως τα χρονικά παράθυρα) θα χρησιμοποιούν το ρολόι συστήματος των υπολογιστών. Ένα ωριαίο χρονικό παράθυρο επεξεργασίας θα περιλαμβάνει όλες τις εγγραφές που έφτασαν σε έναν συγκεκριμένο υπολογιστή μεταξύ του χρονικού διαστήματος όπου το ρολόι του συστήματος ολοκλήρωσε μία πλήρη ώρα. Για παράδειγμα, εάν μια εφαρμογή αρχίζει να εκτελείται στις 8:25 π.μ., το πρώτο ωριαίο χρονικό παράθυρο επεξεργασίας θα περιλαμβάνει γεγονότα που επεξεργάστηκαν μεταξύ 8:25 π.μ. και 9:00 π.μ., το επόμενο παράθυρο θα περιλαμβάνει γεγονότα που επεξεργάστηκαν μεταξύ 9:00 π.μ. και 10:00 π.μ. κ.ο.κ.

Ο χρόνος επεξεργασίας είναι η απλούστερη έννοια του χρόνου και δεν απαιτεί συντονισμό μεταξύ των γεγονότων και των υπολογιστών που πραγματοποιούν την επεξεργασία. Παρέχει την καλύτερη απόδοση και τη χαμηλότερη καθυστέρηση.

Ωστόσο, σε κατανεμημένα και ασύγχρονα περιβάλλοντα ο χρόνος επεξεργασίας δεν παρέχει ντετερμινισμό, επειδή είναι ευαίσθητος στην ταχύτητα με την οποία φτάνουν οι εγγραφές στο σύστημα, στην ταχύτητα με την οποία οι εγγραφές ρέουν μεταξύ των επεξεργαστών μέσα στο σύστημα και στις διακοπές (προγραμματισμένες ή άλλες). [29]

Χρόνος γεγονότος

Ο χρόνος γεγονότος (event time) είναι ο χρόνος που συνέβη κάθε μεμονωμένο γεγονός στη συσκευή παραγωγής του. Αυτή η ώρα συνήθως ενσωματώνεται στις εγγραφές πριν εισέλθουν στο σύστημα επεξεργασίας και αυτή η χρονοσφραγίδα γεγονότος μπορεί να εξαχθεί από κάθε εγγραφή. Στο χρόνο γεγονότος, η πάροδος του χρόνου εξαρτάται από τα δεδομένα και όχι από οποιαδήποτε ρολόγια. Τα προγράμματα που λειτουργούν με γνώμονα το χρόνο γεγονότος πρέπει να καθορίζουν τον τρόπο δημιουργίας υδατοσήμων (watermarks) χρόνου γεγονότος, ο οποίος είναι ο μηχανισμός που σηματοδοτεί την πρόοδο στο χρόνο γεγονότος. Αυτός ο μηχανισμός υδατοσήμανσης (watermarking) περιγράφεται παρακάτω σε επόμενο κεφάλαιο.

Σε έναν τέλειο κόσμο, η επεξεργασία χρόνου γεγονότων θα απέδιδε απολύτως συνεπή και ντετερμινιστικά αποτελέσματα, ανεξάρτητα από το πότε φτάνουν τα γεγονότα ή τη σειρά τους. Ωστόσο, εκτός αν είναι γνωστό ότι τα γεγονότα φθάνουν με τη σειρά τους (βάσει χρονοσφραγίδας), η επεξεργασία βάση του χρόνου γεγονότων συνεπάγεται κάποια καθυστέρηση κατά την αναμονή γεγονότων εκτός σειράς. Καθώς είναι δυνατή η αναμονή μόνο για ένα πεπερασμένο χρονικό διάστημα, αυτό θέτει ένα όριο στο πόσο ντετερμινιστικές μπορούν να είναι οι εφαρμογές χρόνου γεγονότων.

Υποθέτοντας ότι έχουν φτάσει όλα τα δεδομένα, οι λειτουργίες του χρόνου γεγονότος θα συμπεριφέρονται όπως αναμένεται και θα παράγουν σωστά και συνεπή αποτελέσματα ακόμη και όταν εργάζονται με γεγονότα εκτός σειράς ή καθυστερημένα ή όταν επεξεργάζονται εκ νέου ιστορικά δεδομένα. Για παράδειγμα, ένα ωριαίο παράθυρο χρόνου γεγονότος θα περιέχει όλες τις εγγραφές που φέρουν

χρονοσφραγίδα γεγονότος που εμπίπτει στην εν λόγω ώρα, ανεξάρτητα από τη σειρά με την οποία φθάνουν ή από το πότε υποβάλλονται σε επεξεργασία.

Χρόνος γεγονότος και υδατοσήμανση (watermarking)

Ένας επεξεργαστής ροής που υποστηρίζει χρόνο γεγονότος χρειάζεται έναν τρόπο μέτρησης για την πάροδο του χρόνου των γεγονότων. Για παράδειγμα, ένας χειριστής παραθύρων που κατασκευάζει ωριαία παράθυρα πρέπει να ειδοποιείται όταν ο χρόνος γεγονότος έχει περάσει πέρα από το τέλος μιας ώρας, ώστε ο χειριστής να μπορεί να κλείσει το παράθυρο που βρίσκεται σε εξέλιξη.

Ο χρόνος γεγονότος (event time) μπορεί να εξελίσσεται ανεξάρτητα από τον χρόνο επεξεργασίας (processing time). Για παράδειγμα, σε ένα πρόγραμμα, ο τρέχων χρόνος των γεγονότων ενός χειριστή μπορεί να υπολείπεται ελαφρώς του χρόνου επεξεργασίας (λαμβάνοντας υπόψη την καθυστέρηση στη λήψη των γεγονότων), ενώ και οι δύο προχωρούν με την ίδια ταχύτητα. Από την άλλη πλευρά, ένα άλλο πρόγραμμα ροής μπορεί να προχωρήσει σε εβδομάδες χρόνου γεγονότων με λίγα μόνο δευτερόλεπτα επεξεργασίας, με τη γρήγορη προώθηση κάποιων ιστορικών δεδομένων που έχουν ήδη αποθηκευτεί σε ένα θέμα Kafka.

Ένας μηχανισμός για τη μέτρηση της προόδου στο χρόνο γεγονότων είναι η υδατοσήμανση (watermark). Οι υδατοσημάνσεις ρέουν ως μέρος της ροής δεδομένων και φέρουν μια χρονοσφραγίδα t. Μία υδατοσήμανση δηλώνει ότι ο χρόνος γεγονότων έχει φθάσει στο χρόνο t σε αυτή τη ροή, πράγμα που σημαίνει ότι δεν πρέπει να υπάρχουν άλλα στοιχεία από τη ροή με χρονοσφραγίδα t' <= t (δηλαδή γεγονότα με χρονοσφραγίδες παλαιότερες ή ίσες με την υδατοσήμανση). [29]

Η εικόνα 4-1 δείχνει μια ροή γεγονότων με (λογικές) χρονοσφραγίδες και υδατοσημάνσεις που ρέουν σε ευθεία γραμμή. Σε αυτό το παράδειγμα τα γεγονότα είναι με τη σειρά (σε σχέση με τις χρονοσφραγίδες τους), πράγμα που σημαίνει ότι οι υδατοσημάνσεις είναι απλώς περιοδικοί δείκτες στη ροή.

Η υδατοσήμαση είναι ζωτικής σημασίας για ροές δεδομένων εκτός σειράς, όπως φαίνεται παρακάτω, όπου τα γεγονότα δεν είναι ταξινομημένα με βάση τις χρονοσφραγίδες τους. Γενικά, μία υδατοσήμανση είναι μια δήλωση ότι μέχρι το συγκεκριμένο σημείο της ροής, θα πρέπει να έχουν φθάσει όλα τα γεγονότα μέχρι μια συγκεκριμένη χρονοσφραγίδα. Μόλις μία υδατοσήμανση φτάσει σε έναν χειριστή, ο χειριστής μπορεί να προωθήσει το εσωτερικό του χρονόμετρο γεγονότων στην τιμή της υδατοσήμανσης.

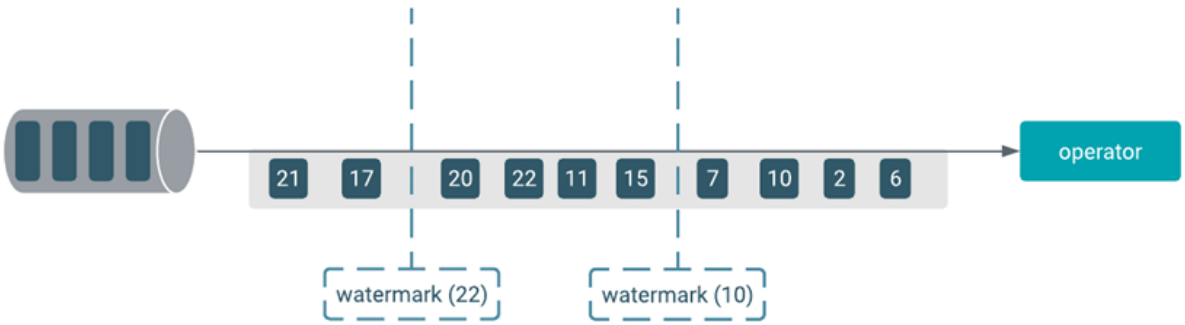
Οι υδατοσημάνσεις επιτρέπουν σε ένα σύστημα να αντιδρά δυναμικά στις μεταβαλλόμενες διαταραχή εισόδου και δεν επιβάλλουν σταθερό κόστος καθυστέρησης.

- Επιτρέπουν τη συνέχιση της σταδιακής επεξεργασίας ακόμη και όταν το σύστημα περιμένει ένα σήμα ότι οι είσοδοι έχουν ολοκληρωθεί.

- Ένα γενικό σύστημα υδατοσήμανσης μπορεί να προσαρμοστεί σε μια ποικιλία διαφορετικών περιπτώσεων χρήσης, ώστε να αξιοποιεί μοναδικά χαρακτηριστικά μιας δεδομένης πηγής εισόδου.

- Οι αδυναμίες, όπως οι καθυστερήσεις, μπορούν συχνά να μετριαστούν με χαλάρωση των περιορισμών πληρότητας που επιβάλλει το σύστημα, ανάλογα με την εκάστοτε περίπτωση.

Για τους λόγους αυτούς οι υδατοσημάνσεις προσφέρουν ένα λογικό συμβιβασμό κόστους/οφέλους σε σχέση με την πληρότητα κατά την επεξεργασία απεριόριστων (unbounded) ροών. [28]

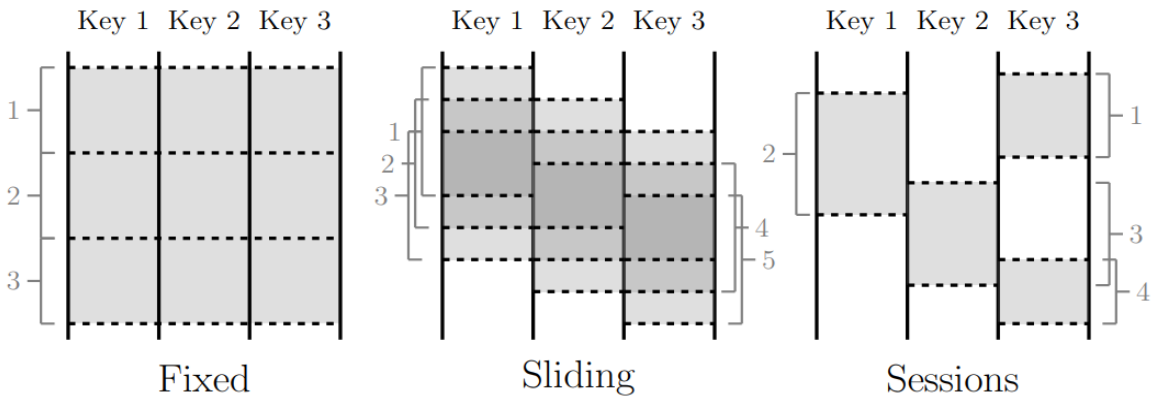


Εικόνα 4-1: Ροή δεδομένων με γεγονότα (μπορεί να φθάνουν εκτός σειράς) και υδατοσήμανση. Ο αριθμός μέσα στο τετραγωνάκι με σκούρο μπλε χρώμα αναφέρεται στη χρονοσφραγίδα του εκάστοτε γεγονότος [30].

Σημειώστε ότι ο χρόνος γεγονότος κληρονομείται από ένα πρόσφατα δημιουργημένο στοιχείο ροής δεδομένων είτε από το γεγονός που τα παρήγαγε είτε από την υδατοσήμανση που προκάλεσε τη δημιουργία αυτών των στοιχείων.

Windowing

Το Windowing τεμαχίζει ένα σύνολο δεδομένων σε πεπερασμένα κομμάτια προκειμένου να επεξεργαστεί ως ομάδα. Όταν πρόκειται για απεριόριστα (unbounded) δεδομένα, η παραθυροποίηση απαιτείται για ορισμένες λειτουργίες (aggregation, outer joins, time-bounded operations κ.λπ.), και περιττή για άλλες (filtering, mapping, inner joins, κ.λπ.). Για τα οριοθετημένα δεδομένα, η παραθυροποίηση είναι ουσιαστικά προαιρετική, αν και εξακολουθεί να είναι σημασιολογικά χρήσιμη έννοια σε πολλές περιπτώσεις (π.χ. backfilling updates μεγάλης κλίμακας). Η παραθυροποίηση είναι πάντα χρονικά συσχετιζόμενη. Τα στοιχεία κατά σειρά έχουν διαδοχικά αυξανόμενες λογικές χρονοσφραγίδες. Τα παράθυρα μπορεί να είναι είτε ευθυγραμμισμένα, δηλαδή να εφαρμόζονται σε όλα τα δεδομένα του παραθύρου του εν λόγω χρονικού διαστήματος, είτε μη ευθυγραμμισμένα, δηλαδή εφαρμόζονται μόνο σε συγκεκριμένα υποσύνολα δεδομένων (π.χ. ανά κλειδί) για το συγκεκριμένο χρονικό παράθυρο. Η εικόνα 4-2 υπογραμμίζει τρεις από τους κύριους τύπους παραθύρων που συναντώνται κατά την επεξεργασία απεριόριστων (unbounded) δεδομένων.



Εικόνα 4-2: Συχνά χρησιμοποιούμενοι τύποι χρονικών παραθύρων

Τα σταθερά (fixed) χρονικά παράθυρα (μερικές φορές αποκαλούμενα και tumbling windows) ορίζονται από ένα στατικό μέγεθος παραθύρου, π.χ. ωριαία ή ημερήσια παράθυρα. Είναι γενικά ευθυγραμμισμένα, δηλαδή κάθε παράθυρο εφαρμόζεται σε όλα τα δεδομένα για την αντίστοιχη περίοδο του χρόνου. Για λόγους διασποράς του φορτίου των παραθύρων ομοιόμορφα στο χρόνο, είναι μερικές φορές μη ευθυγραμμισμένα ανά φάση μετατοπίζοντας τα παράθυρα για κάθε κλειδί κατά κάποια τυχαία τιμή.

Τα παράθυρα ολίσθησης (sliding windows) ορίζονται από ένα μέγεθος παραθύρου και μια ολίσθηση π.χ. ωριαία παράθυρα που ξεκινούν κάθε λεπτό. Η περίοδος μπορεί να είναι μικρότερη από το μέγεθος, πράγμα που σημαίνει ότι τα παράθυρα μπορεί να επικαλύπτονται. Τα παράθυρα ολίσθησης είναι επίσης συνήθως ευθυγραμμισμένα, παρόλο που το διάγραμμα σχεδιάζεται για να δώσει την αίσθηση της κίνησης, και τα πέντε παράθυρα θα εφαρμόζονταν και στα τρία κλειδιά στο διάγραμμα, όχι μόνο στο παράθυρο 3. Τα σταθερά παράθυρα είναι πραγματικά μια ειδική περίπτωση συρόμενων παραθύρων όπου το μέγεθος ισούται με την περίοδο.

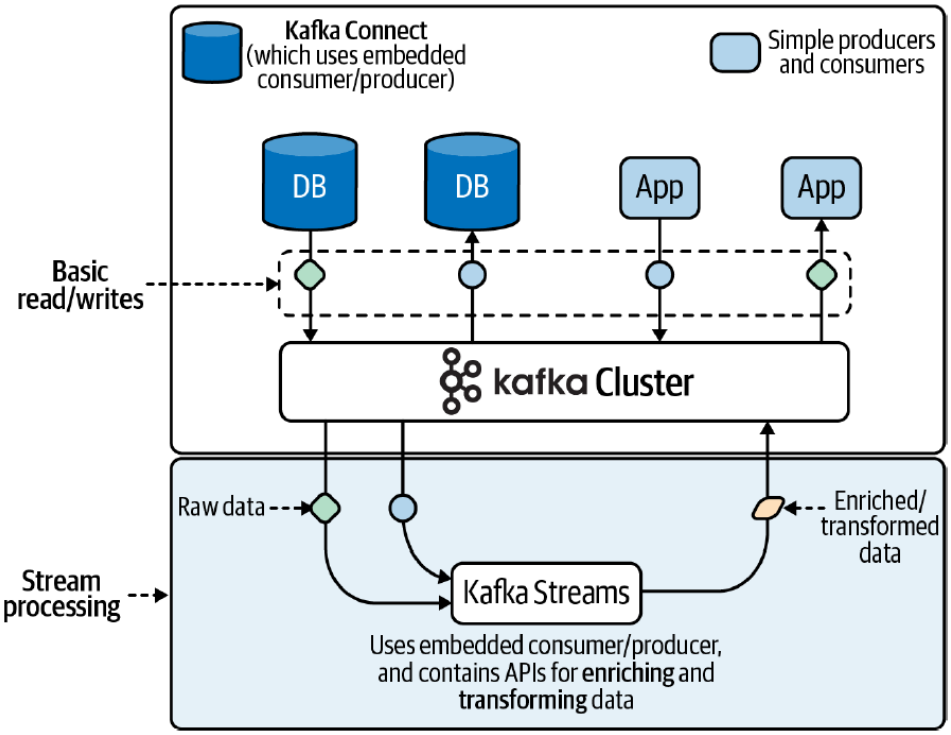
Τα sessions είναι παράθυρα που καταγράφουν κάποια περίοδο δραστηριότητας σε ένα υποσύνολο των δεδομένων. Συνήθως ορίζονται από ένα χρονικό κενό. Οποιαδήποτε γεγονότα που συμβαίνουν μέσα σε χρονικό διάστημα μικρότερο του χρονικού ορίου ομαδοποιούνται μαζί ως session. Τα sessions είναι μη ευθυγραμμισμένα παράθυρα. Για το παράδειγμα, το παράθυρο 2 ισχύει μόνο για το

κλειδί 1, το παράθυρο 3 για το κλειδί 2 μόνο, και τα παράθυρα 1 και 4 μόνο στο κλειδί 3. [31]

5 Kafka Streams

Εισαγωγή

Το Kafka Streams είναι μια Java βιβλιοθήκη που λειτουργεί ως πελάτης (client) στο σύστημα Apache Kafka (εικόνα 5-1). Έχει σχεδιαστεί για την επεξεργασία και ανάλυση δεδομένων που είναι αποθηκευμένα στο Kafka. Επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές που μπορούν να επεξεργάζονται μεγάλο όγκο δεδομένων σε πραγματικό χρόνο αποτελεσματικά και με χαμηλή καθυστέρηση [28]. Το Kafka Streams παρέχει μια υψηλού επιπέδου γλώσσα Streams DSL (Domain Specific Language) για κοινές λειτουργίες μετασχηματισμού δεδομένων και ένα χαμηλού επιπέδου Processor API για προσαρμοσμένους επεξεργαστές και αλληλεπιδράσεις με τα state stores. Τα state stores είναι μια έννοια που θα αναλυθεί παρακάτω.



Εικόνα 5-1: Το Kafka Streams καταναλώνει δεδομένα από ένα Kafka Cluster, τα επεξεργάζεται, και δύναται η εκ νέου εγγραφή εμπλουτισμένων/μετασχηματισμένων δεδομένων πίσω σε ένα θέμα Kafka [27] .

Βασικά χαρακτηριστικά

Το Kafka Streams βασίζεται σε βασικές έννοιες επεξεργασίας ροών, όπως η διάκριση μεταξύ του χρόνου γεγονότων και του χρόνου επεξεργασίας, η υποστήριξη χρονικών παραθύρων (time windows) και η αποδοτική διαχείριση και αναζήτηση της κατάστασης της εφαρμογής σε πραγματικό χρόνο. Έχει σχεδιαστεί ως μια ελαφριά βιβλιοθήκη-πελάτης χωρίς εξωτερικές εξαρτήσεις, πέρα από το ίδιο το Apache Kafka.

Ορισμένα βασικά χαρακτηριστικά του Kafka Streams περιλαμβάνουν:

- Επεκτασιμότητα: Το Kafka Streams αξιοποιεί το μοντέλο κατάτμησης του Kafka για την οριζόντια κλιμάκωση της επεξεργασίας, διατηρώντας παράλληλα ισχυρές εγγυήσεις για τη σειρά με την οποία τα μηνύματα επεξεργάζονται.
- Ανοχή σφαλμάτων: Υποστηρίζει ανεκτική σε σφάλματα τοπική κατάσταση, επιτρέποντας γρήγορες και αποδοτικές λειτουργίες σε κατάσταση, όπως οι παραθυρικές ενώσεις (windowed joins) και οι συναθροίσεις (aggregations).
- Σημασιολογία επεξεργασίας Exactly-once: Το Kafka Streams εγγυάται ότι κάθε εγγραφή επεξεργάζεται μία και μόνο μία φορά, ακόμη και σε περίπτωση αποτυχίας είτε στους πελάτες Streams είτε στους διαμεσολαβητές Kafka κατά τη διάρκεια της επεξεργασίας.
- Χαμηλή καθυστέρηση: Η επεξεργασία μιας εγγραφής τη φορά βοηθά στην επίτευξη καθυστέρησης επεξεργασίας χιλιοστών του δευτερολέπτου και υποστηρίζει παραθυρικές λειτουργίες βασισμένες σε χρόνο γεγονότος με άφιξη εγγραφών εκτός σειράς.

Η βασική οντότητα στο Kafka Streams είναι μια ροή (stream), η οποία αναπαριστά ένα απεριόριστο σύνολο δεδομένων που ανανεώνεται διαρκώς. Μια ροή είναι μια διατεταγμένη, αναπαραγώγιμη (reproducible) και ανεκτική σε σφάλματα (fault tolerant) ακολουθία αμετάβλητων εγγραφών, όπου μια εγγραφή

είναι ένα ζεύγος κλειδιού-τιμής. Οι εφαρμογές επεξεργασίας ροών χρησιμοποιούν τη βιβλιοθήκη Kafka Streams για να ορίσουν την υπολογιστική τους λογική μέσω μίας ή περισσότερων processor topologies. Μια processor topology είναι ένας γράφος από stream processors (κόμβοι - nodes) που συνδέονται με ροές (ακμές).

Υπάρχουν δύο ειδικοί επεξεργαστές στην τοπολογία: Source processor και Sink processor. Ο Source Processor παράγει ροές εισόδου από ένα ή περισσότερα θέματα Kafka, ενώ ο Sink Processor στέλνει εγγραφές σε ένα συγκεκριμένο θέμα Kafka. Οι κόμβοι του κανονικού επεξεργαστή μπορούν να έχουν πρόσβαση σε άλλα απομακρυσμένα συστήματα κατά τη διάρκεια της επεξεργασίας, επιτρέποντας τη ροή των αποτελεσμάτων πίσω στην Kafka ή την εγγραφή σε ένα εξωτερικό σύστημα.

Το Kafka Streams δίνει έμφαση στη σημασία του χρόνου στην επεξεργασία ροής. Χρησιμοποιεί το inferface TimestampExtractor για την ανάθεση χρονοσφραγίδων σε εγγραφές δεδομένων, οι οποίες οδηγούν σε λειτουργίες που εξαρτώνται από το χρόνο, όπως ο χειρισμός των χρονικών παραθύρων (windowing). Οι προγραμματιστές μπορούν να επιβάλλουν διαφορετικές έννοιες του χρόνου, όπως ο χρόνος γεγονότος (event time), ο χρόνος επεξεργασίας (processing time) ή ο χρόνος εισαγωγής (ingestion time), ανάλογα με τις ανάγκες τους.

Ροές – Πίνακες

Μια θεμελιώδης έννοια στο Kafka Streams είναι η δυαδικότητα ροής-πίνακα. Αυτή η δυαδικότητα υπογραμμίζει ότι μια ροή μπορεί να θεωρηθεί ως πίνακας και ένας πίνακας μπορεί να θεωρηθεί ως ροή.

- Ροή ως πίνακας: Μια ροή μπορεί να θεωρηθεί ένα changelog ενός πίνακα, όπου κάθε εγγραφή δεδομένων στη ροή αποτυπώνει μια αλλαγή κατάστασης του πίνακα. Με την αναπαραγωγή του changelog από την αρχή έως το τέλος, είναι δυνατή η ανακατασκευή τον πίνακα. Σε γενικές γραμμές, η συγκέντρωση εγγραφών δεδομένων σε μια ροή, όπως ο υπολογισμός του συνολικού αριθμού προβολών σελίδων ανά χρήστη από μια ροή γεγονότων προβολής σελίδων, οδηγεί σε έναν πίνακα.
- Πίνακας ως ροή: Ένας πίνακας μπορεί να θεωρηθεί ένα στιγμιότυπο, σε μια χρονική στιγμή, της τελευταίας τιμής για κάθε κλειδί σε μια ροή (οι εγγραφές δεδομένων μιας ροής είναι ζεύγη κλειδιών-τιμών). Κάνοντας επανάληψη σε κάθε εγγραφή κλειδιού-τιμής στον πίνακα, ο πίνακας μπορεί να μετατραπεί σε ροή δεδομένων.

Το Kafka Streams χρησιμοποιεί αυτή τη δυαδικότητα για διάφορους σκοπούς, όπως η ελαστικοποίηση των εφαρμογών, η υποστήριξη ανεκτικής σε σφάλματα stateful επεξεργασίας και η εκτέλεση διαδραστικών ερωτημάτων έναντι των τελευταίων αποτελεσμάτων επεξεργασίας. Οι προγραμματιστές μπορούν επίσης να αξιοποιήσουν τη δυαδικότητα ροής-πίνακα στις δικές τους εφαρμογές χρησιμοποιώντας τα βασικά interfaces του Kafka Streams: KStream, KTable και GlobalKTable [33].

Το KStream αναπαριστά μια ροή εγγραφών, όπου κάθε εγγραφή δεδομένων είναι ένα ζεύγος κλειδιού-τιμής. Το KTable αναπαριστά μια ροή αρχείων αλλαγών, η οποία, όταν υλοποιηθεί, παρέχει μια προβολή των συγκεντρωτικών εγγραφών σε μορφή πίνακα. Το GlobalKTable αναπαριστά έναν πλήρως αναπαραγόμενο, μη κατανεμημένο πίνακα, παρέχοντας μια συνεπή προβολή των δεδομένων σε όλες τις περιπτώσεις ροών Kafka σε μια εφαρμογή.

Αποθήκες κατάστασης (State Stores)

Οι αποθήκες κατάστασης στις ροές Apache Kafka είναι τοπικές μονάδες αποθήκευσης που χρησιμοποιούνται για τη διατήρηση πληροφοριών κατάστασης κατά τη διάρκεια της επεξεργασίας ροής. Επιτρέπουν στις εφαρμογές να αποθηκεύουν και να αναζητούν αποτελεσματικά τα δεδομένα που είναι απαραίτητα για διάφορες λειτουργίες με κατάσταση, όπως αθροίσεις (aggregations), ενώσεις (joins) και παραθυροποίηση (windowing). Οι αποθήκες κατάστσης αποτελούν βασικό στοιχείο για την επίτευξη υψηλών επιδόσεων και ανοχής σφαλμάτων στις εφαρμογές Kafka Streams.

Μπορούν να υποστηρίζονται από διαφορετικά συστήματα αποθήκευσης, όπως key-value stores στη μνήμη ή RocksDB, ανάλογα με την περίπτωση χρήσης και τις απαιτήσεις. Το Kafka Streams παρέχει ενσωματωμένες υλοποιήσεις state store, αλλά οι προγραμματιστές μπορούν επίσης να δημιουργήσουν προσαρμοσμένες υλοποιήσεις state store, αν χρειαστεί.

Υπάρχουν δύο κύριοι τύποι state store στο Kafka Streams:

1. KeyValueStore: Ένα key-value store κρατά δεδομένα ως ζεύγη κλειδιών-τιμών, επιτρέποντας στους προγραμματιστές να εκτελούν λειτουργίες όπως put, get και delete με βάση τα κλειδιά. Αυτός ο τύπος αποθήκευσης είναι κατάλληλος για λειτουργίες όπως οι αθροίσεις και οι ενώσεις που απαιτούν γρήγορη πρόσβαση σε μεμονωμένες εγγραφές.

2. WindowStore: Ένας χώρος αποθήκευσης που οργανώνει τα δεδομένα σε παράθυρα με βάση το χρόνο, επιτρέποντας στους προγραμματιστές να εκτελούν χρονικές λειτουργίες, όπως αθροίσεις και ενώσεις σε κάθε παράθυρο. Υποστηρίζει λειτουργίες όπως put, fetch και delete με βάση κλειδιά και χρονικά διαστήματα.

Οι αποθήκες κατάστασης στο Kafka Streams είναι τοπικές σε κάθε περίπτωση της εφαρμογής επεξεργασίας ροής, εξασφαλίζοντας γρήγορη πρόσβαση στα αποθηκευμένα δεδομένα. Ωστόσο, αυτή η τοπική κατάσταση μπορεί να χαθεί εάν μια περίπτωση εφαρμογής αποτύχει. Για την αντιμετώπιση αυτού του προβλήματος, το Kafka Streams αναπαράγει τα δεδομένα του state store σε ένα θέμα changelog στο Kafka, παρέχοντας τη δυνατότητα ανάκτησης (recovery) με ανοχή σε σφάλματα.

Όταν μια εφαρμογή Kafka Streams ξεκινά ή ανακάμπτει από μια αποτυχία, επαναφέρει τα τοπικά αποθέματα κατάστασης από το θέμα changelog πριν συνεχίσει την επεξεργασία. Αυτός ο μηχανισμός αντιγραφής διασφαλίζει ότι η κατάσταση της εφαρμογής δεν χάνεται ακόμη και σε περίπτωση αποτυχίας και παρέχει ένα συνεπές και ανεκτικό σε σφάλματα περιβάλλον επεξεργασίας.[34]

Δεδομένα εκτός σειράς

Τα δεδομένα εκτός σειράς αναφέρονται σε εγγραφές δεδομένων σε μια ροή που φτάνουν με μια σειρά που δεν ακολουθεί την αναμενόμενη σειρά με βάση τις χρονοσφραγίδες τους. Με άλλα λόγια, εγγραφές με μεταγενέστερες χρονοσφραγίδες μπορεί να φτάσουν και να υποβληθούν σε επεξεργασία πριν από εγγραφές με προγενέστερες χρονοσφραγίδες. Αυτό μπορεί να συμβεί για διάφορους λόγους, όπως καθυστερήσεις δικτύου, βλάβες συστήματος ή καθυστερήσεις στην επεξεργασία δεδομένων.

Οι εκτός σειράς εγγραφές δημιουργούν προκλήσεις σε εφαρμογές επεξεργασίας ροής, καθώς μπορεί να επηρεάσουν την ορθότητα της λογικής επεξεργασίας, ειδικά για λειτουργίες με κατάσταση (stateful operations), όπως οι αθροίσεις (aggregations) και οι συνενώσεις (joins). Ο χειρισμός των εκτός σειράς εγγραφών συχνά απαιτεί συμβιβασμούς μεταξύ καθυστέρησης, κόστους και ακρίβειας και μπορεί να περιλαμβάνει τη δημιουργία ενδιάμεσων buffered δεδομένων, τη χρήση τεχνικών παραθύρου ή την αξιοποίηση άλλων στρατηγικών για να ληφθεί υπόψη η εκτός σειράς άφιξη των μηνυμάτων.

Αρχικά, παρουσιάζονται οι δύο κύριες αιτίες των δεδομένων εκτός σειράς στο Kafka Streams:

- Μέσα σε ένα topic-partition, η χρονοσφραγίδα μιας εγγραφής μπορεί να μην είναι αυξανόμενη μαζί με τις μετατοπίσεις τους. Το Kafka Streams επεξεργάζεται τις εγγραφές μέσα σε ένα topic-partition με βάση τη σειρά μετατόπισής τους, οπότε οι εγγραφές με μεγαλύτερες χρονοσφραγίδες αλλά μικρότερες μετατοπίσεις μπορεί να επεξεργαστούν νωρίτερα από τις εγγραφές με μικρότερες χρονοσφραγίδες αλλά μεγαλύτερες μετατοπίσεις.
- Μέσα σε μια εργασία ροής που επεξεργάζεται πολλαπλά topic-partitions, δύναται οι χρήστες να ρυθμίσουν την εφαρμογή ώστε να μην περιμένει να περιέχουν όλα τα partitions κάποια αποθηκευμένα δεδομένα και να επιλέγει από το partition με τη μικρότερη χρονοσφραγίδα. Οι εγγραφές που λαμβάνονται αργότερα από άλλα topic-partitions ενδέχεται να έχουν μικρότερες χρονοσφραγίδες από εκείνες που έχουν ήδη επεξεργαστεί.

Το Kafka Streams χειρίζεται τα μηνύματα εκτός σειράς χρησιμοποιώντας επεξεργασία σε χρόνο γεγονότος, η οποία του επιτρέπει να διατηρεί τη σωστή σημασιολογία επεξεργασίας. Αυτό είναι ιδιαίτερα σημαντικό σε κατανεμημένα συστήματα, όπου διάφοροι παράγοντες όπως οι καθυστερήσεις δικτύου, η μετατόπιση του ρολογιού προκαλούν την επεξεργασία μηνυμάτων εκτός σειράς. Ακολουθούν τεχνικές με χρήση των οποίων το Kafka Streams χειρίζεται τα μηνύματα εκτός σειράς:

- **Χρονοσφραγίδες (timestamps) ανά εγγραφή**: Το Kafka Streams βασίζεται στις χρονοσφραγίδες που ενσωματώνονται σε κάθε μήνυμα για τον προσδιορισμό του χρόνου του γεγονότος. Αυτές οι χρονοσφραγίδες είτε ορίζονται από τον παραγωγό είτε εξάγονται από το Kafka Streams χρησιμοποιώντας το interface TimestampExtractor. Χρησιμοποιώντας αυτές τις χρονοσφραγίδες ανά εγγραφή, το Kafka Streams είναι σε θέση να αιτιολογήσει τις χρονικές σχέσεις μεταξύ των μηνυμάτων, ανεξάρτητα από τη σειρά με την οποία φθάνουν.
- **Windowing**: Για τον χειρισμό μηνυμάτων εκτός σειράς, το Kafka Streams χρησιμοποιεί παράθυρα με βάση το χρόνο γεγονότων. Αυτά τα παράθυρα ορίζονται από τις χρονοσφραγίδες των μηνυμάτων και όχι από τον χρόνο επεξεργασίας. Αυτό σημαίνει ότι ακόμη και αν τα μηνύματα φτάσουν εκτός σειράς, θα τοποθετηθούν στο σωστό παράθυρο με βάση τις ενσωματωμένες χρονοσφραγίδες τους. Το Kafka Streams υποστηρίζει διάφορους τύπους παραθύρων, όπως tumbling, hopping και sliding windows. Επιπλέον, το Kafka Streams παρέχει επίσης υποστήριξη για session windows που ομαδοποιούν γεγονότα με βάση περιόδους αδράνειας, κάτι που μπορεί να είναι χρήσιμο για το χειρισμό μηνυμάτων εκτός σειράς σε σενάρια όπου εμπλέκονται user sessions.
- **Grace period**: Κατά την εκτέλεση παραθυρικών λειτουργιών σε δεδομένα εκτός σειράς, είναι δυνατόν τα καθυστερημένα μηνύματα να απορρίπτονται εάν το παράθυρο στο οποίο ανήκουν έχει ήδη κλείσει. Για την αποφυγή αυτού του προβλήματος, το Kafka Streams επιτρέπει να οριστεί μια περίοδο χάριτος (grace period) για τις λειτουργίες σε χρονικά παράθυρα. Η περίοδος χάριτος καθορίζει για πόσο χρονικό διάστημα θα παραμείνει ανοιχτό ένα παράθυρο για τα καθυστερημένα μηνύματα μετά την παρέλευση του χρόνου λήξης του παραθύρου. Κατά τη διάρκεια αυτής της περιόδου χάριτος, τυχόν μηνύματα εκτός σειράς που φτάνουν θα εξακολουθήσουν να υποβάλλονται σε επεξεργασία και να περιλαμβάνονται στα αποτελέσματα του παραθύρου.
- **Υδατοσημάνσεις (Watermarks)**: Αν και το Kafka Streams δεν χρησιμοποιεί ρητά υδατοσημάνσεις, η έννοια της υδατοσήμανσης μπορεί να εφαρμοστεί για τον έλεγχο της προόδου της επεξεργασίας σε χρόνο γεγονότων. Μία

υδατοσήμανση αντιπροσωπεύει το σημείο στο χρόνο γεγονότος μέχρι το οποίο όλα τα μηνύματα θεωρείται ότι έχουν ληφθεί. Εφαρμόζοντας προσαρμοσμένη λογική με ένα TimestampExtractor, μπορούν να δημιουργηθούν υδατοσημάνσεις για τον έλεγχο της προόδου του χρόνου γεγονότος και την επεξεργασία των μηνυμάτων που φθάνουν με καθυστέρηση (late events).

- **Stateful processing**: Το Kafka Streams διατηρεί τοπικές αποθήκες κατάστασης για stateful λειτουργίες, όπως αθροίσεις, ενώσεις και παραθυρικές λειτουργίες. Αυτές οι αποθήκες κατάστασης μπορούν να χειριστούν μηνύματα εκτός σειράς ενημερώνοντας την κατάσταση με βάση τη χρονοσφραγίδα του μηνύματος, διασφαλίζοντας ότι τα αποτελέσματα της επεξεργασίας είναι συνεπή με τη σημασιολογία του χρόνου γεγονότος.

Επιπλέον, είναι δυνατός ο χειρισμός μηνυμάτων ενός χρονικού παραθύρου που φθάνουν μετά τη λήξη του grace period. Απαιτείται προσαρμοσμένη λογική στην εφαρμογή Kafka Streams για τη διαχείριση της προσωρινής αποθήκευσης και της επεξεργασίας των καθυστερημένων εγγραφών. Η βασική ιδέα είναι πως οι εγγραφές θα αποθηκεύονται για ένα εκτεταμένο χρονικό διάστημα και αργότερα θα επεξεργάζονται με τη σειρά βάσει των χρονοσφραγίδων τους.

Ακολουθεί μια περιγραφή υψηλού επιπέδου για τον τρόπο με τον οποίο μπορούν να χειριστούν οι εγγραφές που φτάνουν μετά την περίοδο χάριτος.

1. Δημιουργία ενός state store στην εφαρμογή Kafka Streams για την αποθήκευση των εισερχόμενων εγγραφών. Το state store θα λειτουργεί ως προσωρινός αποθηκευτικός χώρος για να κρατάει τις εγγραφές που φτάνουν εκτός σειράς ή μετά την περίοδο χάριτος.
2. Για κάθε εισερχόμενη εγγραφή, θα ελέγχεται εάν είναι εντός της περιόδου χάριτος ή όχι. Εάν είναι εντός της περιόδου χάριτος, θα επεξεργάζεται

αμέσως. Αν όχι, θα αποθηκεύεται στην αποθήκη κατάστασης με κλειδί τη χρονοσφραγίδα της.

3. Σε τακτά χρονικά διαστήματα (π.χ. χρησιμοποιώντας μια προγραμματισμένη εργασία), οι εγγραφές που είναι αποθηκευμένες στο state store θα ταξινομούνται με βάση τη χρονοσφραγίδα τους και επεξεργάζονται με τη σειρά. Μετά την επεξεργασία, θα αφαιρούνται από το state store.

Για την αποφυγή της επ' αόριστον αύξηση του state store, μπορεί να εφαρμοστεί μια πολιτική εκδίωξης (retention), όπως η αφαίρεση εγγραφών που είναι παλαιότερες από ένα ορισμένο όριο.

Είναι σημαντικό να σημειωθεί ότι ο χειρισμός των εγγραφών που φτάνουν μετά την περίοδο χάριτος ενδέχεται να μην είναι πάντα εφικτός, ειδικά όταν η επεξεργασία αφορά μεγάλο όγκο δεδομένων ή σε περιπτώσεις όπου απαιτείται αυστηρή σειρά επεξεργασίας. Σε τέτοια σενάρια, ενδέχεται να χρειαστεί να επαναξιολογηθεί η διοχέτευση δεδομένων για να διασφαλιστεί η έγκαιρη παράδοση των εγγραφών ή να προσαρμοστούν ανάλογα οι απαιτήσεις.

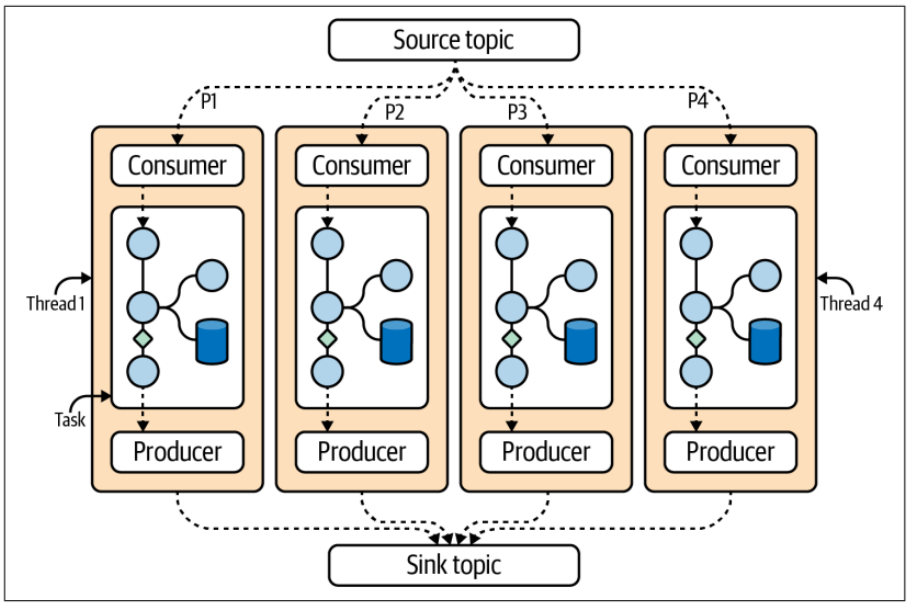
Κατάτμηση σε ροές, εργασίες και νήματα

Το επίπεδο ανταλλαγής μηνυμάτων της Kafka και οι ροές Kafka Streams βασίζονται και οι δύο στην κατάτμηση των δεδομένων για αποθήκευση, μεταφορά και επεξεργασία. Η κατάτμηση επιτρέπει την τοπικότητα των δεδομένων, την ελαστικότητα, την επεκτασιμότητα, την υψηλή απόδοση και την ανοχή σε σφάλματα. Το Kafka Streams χρησιμοποιεί τις έννοιες των κατατμήσεων και των εργασιών ως λογικές μονάδες παραλληλισμού, οι οποίες είναι στενά συνδεδεμένες με τις κατατμήσεις θεμάτων του Kafka:

1. Κάθε κατάτμηση ροής αντιστοιχεί σε μια πλήρως διατεταγμένη ακολουθία εγγραφών δεδομένων και αντιστοιχίζεται σε μια κατάτμηση θέματος της Kafka.
2. Μια εγγραφή δεδομένων στη ροή συνδέεται με ένα μήνυμα Kafka από το topic.
3. Τα κλειδιά των εγγραφών δεδομένων καθορίζουν την κατάτμηση των δεδομένων τόσο στην Kafka όσο και στις ροές Kafka, υπαγορεύοντας τον τρόπο δρομολόγησης των δεδομένων σε συγκεκριμένα διαμερίσματα εντός των topics.

Η τοπολογία του επεξεργαστή μιας εφαρμογής κλιμακώνεται με τον διαχωρισμό της σε πολλαπλές εργασίες. Το Kafka Streams δημιουργεί έναν σταθερό αριθμό εργασιών με βάση τις κατατμήσεις των ροών εισόδου για την εφαρμογή, αναθέτοντας σε κάθε εργασία έναν κατάλογο κατατμήσεων από τις ροές εισόδου (δηλ. τα topics του Kafka). Η ανάθεση κατατμήσεων σε εργασίες παραμένει σταθερή, καθιστώντας κάθε εργασία μια σταθερή μονάδα παραλληλισμού. Στη συνέχεια, οι εργασίες μπορούν να δημιουργήσουν τη δική τους τοπολογία επεξεργαστών με βάση τις κατατμήσεις που τους έχουν ανατεθεί και να διατηρούν έναν buffer για κάθε κατατμήση που τους έχει ανατεθεί, επεξεργαζόμενες τα μηνύματα ένα προς ένα από αυτούς τους buffers. Ως αποτέλεσμα, οι εργασίες ροής μπορούν να επεξεργάζονται ανεξάρτητα και παράλληλα χωρίς χειροκίνητη παρέμβαση.

Με απλούστερους όρους, ο μέγιστος παραλληλισμός για μια εφαρμογή περιορίζεται από τον μέγιστο αριθμό εργασιών ροής, ο οποίος καθορίζεται από τον μέγιστο αριθμό κατατμήσεων (partitions) του θέματος εισόδου (ή των θεμάτων εισόδου) από το οποίο διαβάζει η εφαρμογή. Για παράδειγμα, εάν το θέμα εισόδου έχει πέντε κατατμήσεις, μπορούν να εκτελεστούν έως και πέντε στιγμιόπα της εφαρμογής. Αυτά τα στιγμιότυπα θα επεξεργάζονται συνεργατικά τα δεδομένα του θέματος. Εάν ο αριθμός των στιγμιοτύπων της εφαρμογής είναι μεγαλύτερος από τον αριθμό των κατατμήσεων του θέματος εισόδου, τα επιπλέον στιγμιότυπα της εφαρμογής θα εκκινηθούν αλλά θα παραμείνουν σε αδράνεια. Ωστόσο, εάν ένα από τα ενεργά στιγμιότυπα αποτύχει, ένα αδρανές στιγμιότυπο θα αναλάβει την εργασία της.



Εικόνα 5-2: Απεικόνιση μια Kafka Stream τοπολογίας. Οι καταναλωτές P1-P4 διαβάζουν από το ίδιο θέμα, επεξεργάζονται τα δεδομένα και στη συνέχεια τα γράφουν σε ένα άλλο θέμα Kafka. Κάθε καταναλωτής αναλαμβάνει να φέρει εις πέρας μία εργασία (task), η οποία τρέχει σε ξεχωριστό νήμα (thread) [27].

Το Kafka Streams δεν είναι ένας διαχειριστής πόρων αλλά μια βιβλιοθήκη που λειτουργεί οπουδήποτε εκτελείται η εφαρμογή επεξεργασίας ροής του. Πολλαπλά στιγμιότυπα της εφαρμογής μπορούν να εκτελούνται στο ίδιο μηχάνημα ή να κατανέμονται σε πολλά μηχανήματα, με τις εργασίες να κατανέμονται αυτόματα από τη βιβλιοθήκη σε αυτά τα στιγμιότυπα της εφαρμογής που εκτελούνται. Η ανάθεση κατατμήσεων σε εργασίες (tasks) παραμένει σταθερή. Εάν ένα στιγμιότυπο της εφαρμογής αποτύχει, όλες οι εργασίες που του έχουν ανατεθεί θα επανεκκινηθούν αυτόματα σε άλλα στιγμιότυπα και θα συνεχίσουν να καταναλώνουν από τις ίδιες κατατμήσεις ροής.

Το Kafka Streams δίνει τη δυνατότητα στους χρήστες να ορίσουν τον αριθμό των νημάτων που χρησιμοποιεί η βιβλιοθήκη για παράλληλη επεξεργασία σε ένα στιγμιότυπο της εφαρμογής. Κάθε νήμα μπορεί να εκτελεί ανεξάρτητα μία ή περισσότερες εργασίες μαζί με τις τοπολογίες των επεξεργαστών τους.

Η έναρξη περισσότερων νημάτων ροής ή πρόσθετων περιπτώσεων της εφαρμογής περιλαμβάνει απλώς την αντιγραφή της τοπολογίας και την ανάθεσή της

για την επεξεργασία ενός ξεχωριστού υποσυνόλου των κατατμήσεων του θέματος, επιτρέποντας ουσιαστικά την παράλληλη επεξεργασία. Είναι σημαντικό να σημειωθεί ότι δεν υπάρχει κοινή κατάσταση μεταξύ των νημάτων, εξαλείφοντας την ανάγκη συντονισμού μεταξύ των νημάτων. Αυτή η απλότητα επιτρέπει την παράλληλη εκτέλεση των τοπολογιών σε όλες τις περιπτώσεις εφαρμογών και νήματα. Η κατανομή των κατατμήσεων ενός θέματος μεταξύ των διαφόρων νημάτων ροής διαχειρίζεται από το Kafka Streams, αξιοποιώντας τα χαρακτηριστικά συντονισμού του Kafka.

Όπως αναφέρθηκε προηγουμένως, η κλιμάκωση μιας εφαρμογής επεξεργασίας ροής με χρήση του Kafka Streams είναι απλή. Χρειάζεται να εκκινηθούν πρόσθετα στιγμιότυπα μιας εφαρμογής και το Kafka Streams θα χειριστεί την κατανομή των κατατμήσεων μεταξύ των εργασιών που εκτελούνται στα στιγμιότυπα της εφαρμογής. Μπορούν να εκκινηθούν τόσα νήματα μιας εφαρμογής όσες είναι και οι κατατμήσεις ενός θέματος, διασφαλίζοντας ότι σε όλα τα ενεργά στιγμιότυπα μιας εφαρμογής, κάθε νήμα (ή ακριβέστερα, οι εργασίες που εκτελεί) επεξεργάζεται τουλάχιστον μία κατάτμηση (εικόνα 5-2).

6 InfluxDB

Εισαγωγή

Στην εποχή των μεγάλων δεδομένων, η ανάγκη για αποτελεσματική διαχείριση και ανάλυση δεδομένων χρονοσειρών έχει γίνει όλο και πιο σημαντική. Τα δεδομένα χρονοσειρών είναι μια ακολουθία σημείων δεδομένων, που καταγράφονται σε τακτά ή ακανόνιστα χρονικά διαστήματα. Διαδραματίζουν κρίσιμο ρόλο σε διάφορες εφαρμογές, όπως το Διαδίκτυο των πραγμάτων (IoT), η ανάλυση της χρηματοπιστωτικής αγοράς, η παρακολούθηση της απόδοσης εφαρμογών και η διαχείριση υποδομών. Η ραγδαία αύξηση του όγκου και της ποικιλίας των δεδομένων χρονοσειρών οδήγησε στην ανάπτυξη εξειδικευμένων βάσεων δεδομένων, οι οποίες είναι βελτιστοποιημένες για να χειρίζονται τις μοναδικές προκλήσεις που θέτει αυτός ο τύπος δεδομένων [35].

Μια τέτοια βάση δεδομένων χρονοσειρών είναι η InfluxDB, μια κατανεμημένη βάση δεδομένων ανοικτού κώδικα, υψηλής απόδοσης, ειδικά σχεδιασμένη για τη διαχείριση μεγάλων όγκων δεδομένων με χρονοσειρές και υψηλό φόρτο εγγραφής και ερωτημάτων. Η InfluxDB αναπτύχθηκε από την InfluxData και αποτελεί βασικό συστατικό της στοίβας TICK, η οποία περιλαμβάνει το Telegraf (συλλογή και προώθηση δεδομένων), την InfluxDB (βάση δεδομένων χρονοσειρών), το Chronograf (οπτικοποίηση και dashboarding) και το Kapacitor (επεξεργασία δεδομένων σε πραγματικό χρόνο και alerting). Η στοίβα TICK είναι μια ολοκληρωμένη πλατφόρμα που απλοποιεί τη διαδικασία συλλογής, αποθήκευσης, οπτικοποίησης και ανάληψης δράσης σε δεδομένα χρονοσειρών.

Αρχιτεκτονική της InfluxDB

Γενική περιγραφή

Η InfluxDB ακολουθεί μια modular αρχιτεκτονική, με διάφορα στοιχεία που συνεργάζονται για να χειριστούν την εισαγωγή δεδομένων, την αποθήκευση, την ευρετηρίαση (indexing) και την αναζήτηση (querying). Στον πυρήνα της αρχιτεκτονικής της InfluxDB βρίσκεται η μηχανή αποθήκευσης, η οποία έχει σχεδιαστεί για να παρέχει υψηλές επιδόσεις και αποδοτική διαχείρηση πόρων. Η μηχανή αποθήκευσης συμπληρώνεται από ένα ευέλικτο μοντέλο δεδομένων και έναν μηχανισμό indexing που επιτρέπει τη γρήγορη και αποτελεσματική αναζήτηση (querying) δεδομένων χρονοσειρών.

Μηχανή αποθήκευσης δεδομένων

Η InfluxDB χρησιμοποιεί μια προσαρμοσμένη μηχανή αποθήκευσης που βασίζεται σε έναν συνδυασμό αλγορίθμων LSM (Log-Structured Merge-tree) και B+ tree. Αυτή η υβριδική προσέγγιση επιτρέπει στην InfluxDB να χειρίζεται αποτελεσματικά τόσο βαρύ φόρτο εργασίας εγγραφής όσο και βαρύ φόρτο εργασίας ερωτημάτων (queries), καθώς και τα δυο είναι συνηθισμένα στις εφαρμογές δεδομένων χρονοσειρών. [36]

Η μηχανή αποθήκευσης οργανώνει τα δεδομένα σε χρονικά ταξινομημένα shards, όπου κάθε shard περιέχει ένα συγκεκριμένο χρονικό εύρος δεδομένων. Τα shards χωρίζονται περαιτέρω σε αρχεία TSM (Time-Structured Merge), τα οποία αποθηκεύουν συμπιεσμένα και ευρετηριασμένα (indexed) σημεία δεδομένων. Τα αρχεία TSM συμπιέζονται περιοδικά για τη μείωση του χώρου αποθήκευσης και τη βελτίωση της απόδοσης των ερωτημάτων. [37]

Μοντέλο δεδομένων

Η InfluxDB οργανώνει τα δεδομένα χρησιμοποιώντας ένα ιεραρχικό μοντέλο δεδομένων που αποτελείται από μετρήσεις (measurements), πεδία (fields) και ετικέτες (tags). Οι μετρήσεις είναι παρόμοιες με τους πίνακες σε μια σχεσιακή βάση δεδομένων, ενώ τα πεδία και οι ετικέτες αντιπροσωπεύουν τα σημεία δεδομένων και τα μεταδεδομένα, αντίστοιχα.

Μετρήσεις (measurements): Μια μέτρηση είναι ένας περιέκτης για δεδομένα χρονοσειρών και αντιστοιχεί σε μια συγκεκριμένη κατηγορία ή τύπο δεδομένων, όπως ενδείξεις θερμοκρασίας, χρήση CPU ή τιμές μετοχών. Κάθε μέτρηση αποτελείται από ένα σύνολο σημείων δεδομένων με χρονική σήμανση, τα οποία αντιπροσωπεύονται από πεδία και ετικέτες.

Πεδία (fields): Τα πεδία είναι οι πραγματικές τιμές δεδομένων σε μια μέτρηση και συνδέονται πάντα με μια συγκεκριμένη χρονοσφραγίδα (timestamp). Κάθε πεδίο έχει ένα όνομα και μια τιμή και οι τιμές των πεδίων μπορούν να είναι διαφορετικών τύπων δεδομένων, όπως π.χ. float, integer, string ή boolean.

Ετικέτες (tags): Οι ετικέτες είναι ζεύγη κλειδιών-τιμών που παρέχουν μεταδεδομένα για κάθε σημείο δεδομένων σε μια μέτρηση. Βοηθούν στην αποτελεσματική ευρετηρίαση (indexing) και αναζήτηση (quering) δεδομένων, επιτρέποντας στους χρήστες να φιλτράρουν και να συγκεντρώνουν σημεία δεδομένων με βάση τις τιμές των ετικετών τους. Σε αντίθεση με τα πεδία, οι τιμές των ετικετών είναι πάντα τύπου συμβολοσειράς και δεν υπόκεινται σε μαθηματικές πράξεις κατά τη διάρκεια των ερωτημάτων.

Χρησιμοποιώντας τις ετικέτες για το φιλτράρισμα και τη συγκέντρωση (aggregation) των δεδομένων, η InfluxDB μπορεί αποδοτικά να ανακτά υποσύνολα δεδομένων. Η χρήση indexes στις ετικέτες και η οργάνωση των δεδομένων σε μετρήσεις, βοηθούν περαιτέρω τη γρήγορη αναζήτηση.

Clustering και αντιγραφή

Στην έκδοση για επιχειρήσεις της InfluxDB (Enterprise edition), είναι δυνατή η λειτουργία σε cluster mode. Αυτή η λειτουργία προσφέρει αυξημένη ανοχή σε σφάλματα (fault tolerance) και οριζόντια επεκτασιμότητα (horizontal scalability). Σε μια εγκατάσταση σε cluster, τα δεδομένα αναπαράγονται σε πολλαπλούς κόμβους, εξασφαλίζοντας υψηλή διαθεσιμότητα (high availability). Η λειτουργία σε cluster mode επιτρέπει επίσης την εξισορρόπηση φορτίου, με τις λειτουργίες ανάγνωσης και εγγραφής να κατανέμονται στους κόμβους για τη διατήρηση της βέλτιστης απόδοσης.

Αναφορικά με την αντιγραφή, χρησιμοποιείται ένα consensus πρωτόκολλο, το οποίο διασφαλίζει τη συνέπεια (consistency) και τη διάρκεια των δεδομένων (durability) στους κόμβους του cluster. Οι χρήστες μπορούν να ρυθμίζουν το συντελεστή αντιγραφής (replication factor) και τα shard για να ελέγχουν τον τρόπο με τον οποίο τα δεδομένα διανέμονται και διατηρούνται εντός του cluster, ανάλογα με τις συγκεκριμένες απαιτήσεις και τους πόρους τους. [38]

Σχεδιασμός χωρίς σχήμα (Schema-less Design)

Η InfluxDB υιοθετεί σχεδιασμό χωρίς σχήμα, πράγμα που σημαίνει ότι δεν επιβάλλει μια σταθερή δομή για τα δεδομένα που αποθηκεύει. Αυτή η ευέλικτη προσέγγιση επιτρέπει στους χρήστες να αποθηκεύουν δεδομένα με ποικίλες δομές και να προσθέτουν ή να αφαιρούν δυναμικά πεδία και ετικέτες ανάλογα με τις ανάγκες. Ο σχεδιασμός χωρίς σχήμα είναι ιδιαίτερα επωφελής για δεδομένα χρονοσειρών, καθώς το σύνολο των πεδίων και των τιμών μπορεί να αλλάζει με την πάροδο του χρόνου λόγω εξελισσόμενων απαιτήσεων ή αλλαγών στο σύστημα. [39]

Εισαγωγή δεδομένων (Data ingestion)

Η InfluxDB παρέχει επίσημες βιβλιοθήκες-πελάτες (clients) για διάφορες γλώσσες προγραμματισμού, συμπεριλαμβανομένων των Go, Java, JavaScript, Python και άλλων. Αυτές οι βιβλιοθήκες απλοποιούν τη διαδικασία εισαγωγής δεδομένων

στην InfluxDB παρέχοντας βολικά API για τη δημιουργία, τη σειριοποίηση και την αποστολή σημείων δεδομένων στη βάση δεδομένων.

Εκτός από τις βιβλιοθήκες-πελάτες, η InfluxDB παρέχει επίσης ένα RESTful HTTP API για την εισαγωγή δεδομένων. Το API επιτρέπει να στέλνετε σημεία δεδομένων στη μορφή InfluxDB Line Protocol, η οποία είναι μια απλή μορφή βασισμένη σε κείμενο που αναπαριστά σημεία δεδομένων χρονοσειρών και τα σχετικά μεταδεδομένα τους.

Σημεία δεδομένων μπορούν να στέλνονται μεμονωμένα ή σε πακέτα (batches), ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής.[40]

Ενσωμάτωση με το Telegraf και άλλες πηγές δεδομένων

Το Telegraf είναι ένα λογισμικό συλλογής και προώθησης δεδομένων ανοικτού κώδικα που αποτελεί μέρος της στοίβας TICK. Μπορεί να χρησιμοποιηθεί για τη συλλογή μετρήσεων και γεγονότων από διάφορες πηγές, όπως μετρήσεις απόδοσης συστήματος, αρχεία καταγραφής εφαρμογών ή συσκευές IoT, και να τα προωθήσει στην InfluxDB για αποθήκευση.

Το Telegraf παρέχει ένα ευρύ φάσμα από πρόσθετα (plugins) που υποστηρίζουν τη συλλογή ή/και εισροή δεδομένων από διαφορετικές πηγές. Ορισμένα δημοφιλή πρόσθετα εισόδου που χρησιμοποιούνται στο Βιομηχανικό διαδίκτυο των πραγμάτων (Industrial Internet of Thing - IIoT) είναι:

- **AMQP Telegraf Plugin:** Το AMQP Consumer Telegraf Plugin παρέχει στους καταναλωτές τη δυνατότητανα λαμβάνουν δεδομένα ροής μέσω ενός διαμεσολαβητή συμβατού με AMQP 0-9-1, όπως ο RabbitMQ. Οι μετρήσεις διαβάζονται από μια ανταλλαγή θεμάτων χρησιμοποιώντας το κλειδί binding_key της ουράς.

- **ModBus Telegraf Plugin:** Το Modbus Telegraf Plugin συλλέγει διακριτές εισόδους, πηνία, καταχωρητές εισόδου και καταχωρητές αποθήκευσης μέσω Modbus TCP ή Modbus RTU/ASCII. Στις ρυθμίσεις, μπορούν να προσδιοριστούν οι διευθύνσεις της συσκευής Modbus στο δίαυλο, το εύρος, χρονικά όρια, επαναλήψεις,

κ.λπ. για να συλλεχθούν οι μετρήσεις απευθείας από τον εξοπλισμό, από το SCADA ή από τα συστήματα αυτοματισμού.

- **MQTT Telegraf Plugin:** Το MQTT αντλεί όλα τα δεδομένα χρονοσειρών (μετρήσεις και γεγονότα) από τις εφαρμογές, την υποδομή, ακόμη και τους αισθητήρες, καθιστώντας τη χρήση τους ευκολότερη. Το MQTT Consumer Telegraf Input Plugin διαβάζει από καθορισμένα θέματα MQTT και προσθέτει μηνύματα στην InfluxDB. Τα μηνύματα είναι στις μορφές δεδομένων εισόδου του Telegraf. Είναι δυνατή η συλλογή και η παρουσίαση σε γραφική παράσταση μετρήσεων από συσκευές IoT με το πρωτόκολλο Message Queue Telemetry Transport (MQTT) - ένα απλό και ελαφρύ πρωτόκολλο ανταλλαγής μηνυμάτων, ιδανικό για συσκευές IoT.

- **OPC-UA Telegraf Plugin:** Το OPC Unified Architecture (OPC UA) είναι ένα πρωτόκολλο επικοινωνίας για βιομηχανικό αυτοματισμό που αναπτύχθηκε από το OPC Foundation και κυκλοφόρησε το 2006. Η τρέχον έκδοση είναι η 1.04 (κυκλοφόρησε το 2017) και περιλαμβάνει publish/subscribe και client/server δομή επικοινωνίας. Το πρόσθετο OPC-UA Telegraf βοηθά στη συλλογή μετρήσεων από συσκευές που χρησιμοποιούν το πρωτόκολλο OPC-UA.

- **Sensor Telegraf Plugin:** Το Sensor Telegraf Plugin μπορεί βοηθά στη συλλογή μετρικών από αισθητήρες χρησιμοποιώντας το πακέτο *Linux-monitoring (lm-sensor)*. Το *lm_sensors* είναι ένα ελεύθερο ανοικτό εργαλείο λογισμικού για Linux που παρέχει εργαλεία και προγράμματα οδήγησης για την παρακολούθηση θερμοκρασιών, τάσης, υγρασίας κ.α. Χρησιμοποιώντας αυτό το πρόσθετο αισθητήρα Telegraf - σε συνδυασμό με τα πολλά άλλα Telegraf plugins, όπως ModBus, MQTT και OPC UA - μπορεί να βοηθήσει στη συλλογή μετρήσεων από πολλούς τύπους αισθητήρων που βρίσκονται σε σπίτια, εργοστάσια, χώρους παραγωγής ενέργειας κ.λπ. Συλλέγοντας όλα αυτά τα δεδομένα σε μία μόνο βάση δεδομένων χρονοσειρών, όπως η InfluxDB, μπορεί να βοηθήσει στην απόκτηση πληροφοριών σχετικά με την απόδοση, την πρόβλεψη των κύκλων συντήρησης, καθώς και στην εύρεση ευκαιριών βελτιστοποίησης. [41]

Αναζήτηση (Quering)

Παρέχονται 2 γλώσσες όπου ο χρήστης μπορεί να χρησιμοποιήσει για αναζήτηση (quering) δεδομένων: η InfluxQL και η Flux.

Η InfluxQL είναι μια γλώσσα ερωτημάτων που μοιάζει με SQL και έχει σχεδιαστεί ειδικά για την αναζήτηση δεδομένων χρονοσειρών στην InfluxDB. Παρέχει οικεία σύνταξη και λειτουργίες που μοιάζουν με SQL, διευκολύνοντας τους χρήστες που έχουν υπόβαθρο σε SQL να ξεκινήσουν ομαλά με τη χρήση της InfluxDB. Η InfluxQL υποστηρίζει διάφορους τύπους ερωτημάτων, συμπεριλαμβανομένης της επιλογής δεδομένων, του φιλτραρίσματος, της συνάθροισης και των μετασχηματισμών (select, filter, aggregate, transform).

Η Flux είναι μια ισχυρή, λειτουργική γλώσσα σεναρίων δεδομένων που εισήχθη στην InfluxDB 2.0 και παρέχει μεγαλύτερη ευελιξία και δυνατότητες για την υποβολή ερωτημάτων και την επεξεργασία δεδομένων χρονοσειρών. Η Flux επιτρέπει στους χρήστες να εκτελούν σύνθετες εργασίες επεξεργασίας δεδομένων, όπως η ένωση (join) πολλαπλών πηγών δεδομένων, η παραθυροποίηση (windowing) και οι προσαρμοσμένοι μετασχηματισμοί, χρησιμοποιώντας μια αλυσίδα συναρτησιακών τελεστών. [42]

Διαχείριση του συστήματος

Πολιτικές διατήρησης δεδομένων

Οι πολιτικές διατήρησης (retention policies) αποτελούν βασικό χαρακτηριστικό της InfluxDB για τη διαχείριση της αποθήκευσης και της διάρκειας ζωής των δεδομένων. Μια πολιτική διατήρησης ορίζει πόσο καιρό αποθηκεύονται τα δεδομένα στη βάση δεδομένων και πότε θα πρέπει να αφαιρούνται αυτόματα. Με τον καθορισμό κατάλληλων πολιτικών διατήρησης, οι χρήστες μπορούν να διασφαλίσουν ότι τα παλαιότερα δεδομένα, τα οποία ενδέχεται να μην είναι πλέον σχετικά ή χρήσιμα, αφαιρούνται από το σύστημα, απελευθερώνοντας χώρο αποθήκευσης και διατηρώντας τη βέλτιστη απόδοση.

Στην InfluxDB 1.x, οι πολιτικές διατήρησης ορίζονται ανά βάση δεδομένων και κάθε βάση δεδομένων μπορεί να έχει πολλαπλές πολιτικές διατήρησης. Στην InfluxDB 2.0, οι πολιτικές διατήρησης ορίζονται ανά κάδο (bucket) και κάθε κάδος έχει μία μόνο πολιτική διατήρησης.

Συνεχή ερωτήματα (Continuous queries)

Τα συνεχή ερωτήματα είναι ένα ισχυρό χαρακτηριστικό της InfluxDB που επιτρέπει στους χρήστες να υπολογίζουν εκ των προτέρων και να αποθηκεύουν τα αποτελέσματα συχνά εκτελούμενων και υπολογιστικά δαπανηρών ερωτημάτων. Με την περιοδική εκτέλεση αυτών των ερωτημάτων στο παρασκήνιο, τα συνεχή ερωτήματα μπορούν να συμβάλουν στη μείωση της καθυστέρησης των ερωτημάτων και στη βελτίωση της απόδοσης του συστήματος.

Τα συνεχή ερωτήματα μπορούν να είναι ιδιαίτερα χρήσιμα για την υποδειγματοληψία (downsampling) δεδομένων, η οποία είναι η διαδικασία συγκέντρωσης σημείων δεδομένων υψηλής ανάλυσης σε αναπαραστάσεις χαμηλότερης ανάλυσης. Με την υποδειγματοληψία δεδομένων, οι χρήστες μπορούν

να μειώσουν τις απαιτήσεις αποθηκευτικού χώρου και να βελτιστοποιήσουν την απόδοση των ερωτημάτων που δεν απαιτούν δεδομένα υψηλής ανάλυσης. Στην InfluxDB 2.0, τα συνεχή ερωτήματα αντικαθίστανται από εργασίες (tasks), οι οποίες είναι Flux scripts που εκτελούνται περιοδικά για να εκτελούν λειτουργίες επεξεργασίας δεδομένων, όπως η υποδειγματοληψία. [43]

Διαχείριση shards

Η InfluxDB χωρίζει αυτόματα τα δεδομένα σε shards με βάση τα χρονικά διαστήματα. Τα shards είναι η βασική μονάδα αποθήκευσης και διαχείρισης δεδομένων στην InfluxDB. Από προεπιλογή, τα shards δημιουργούνται με την ίδια διάρκεια με τη σχετική πολιτική διατήρησης, αλλά αυτό μπορεί να προσαρμοστεί αν χρειαστεί. Η σωστή διαχείριση των shards μπορεί να συμβάλει στη βελτιστοποίηση της αποθήκευσης και της απόδοσης των ερωτημάτων, ειδικά όταν πρόκειται για μεγάλους όγκους δεδομένων χρονοσειρών.[36]

Παρακολούθηση, οπτικοποίηση και alerting

Για την παρακολούθηση, οπτικοποίηση και alerting μπορεί να χρησιμοποιηθεί το εξωτερικό σύστημα Grafana, με το οποίο η Influx παρέχει έτοιμα εργαλεία επικοινωνίας.

Στην InfluxDB έκδοση τουλάχιστον 2.0, έχουν εισαχθεί ενσωματωμένες δυνατότητες παρακολούθησης και ειδοποίησης μέσω της χρήσης εργασιών (tasks), ελέγχων (checks) και κανόνων ειδοποίησης (notification rules). Αυτές οι δυνατότητες επιτρέπουν στους χρήστες να παρακολουθούν τα δεδομένα χρονοσειρών τους, να εντοπίζουν ανωμαλίες ή προβλήματα και να λαμβάνουν ειδοποιήσεις βάσει συνθηκών προσαρμοσμένων από τους ίδιους. [44][39]

Εργασίες (tasks)

Οι εργασίες είναι Flux scripts που εκτελούνται περιοδικά για την εκτέλεση λειτουργιών επεξεργασίας δεδομένων, όπως ο υπολογισμός μετρικών ή η υποδειγματοληψία δεδομένων.

Έλεγχοι (checks)

Οι έλεγχοι είναι ένα χαρακτηριστικό στην InfluxDB 2.0 που επιτρέπει τον καθορισμό προσαρμοσμένων συνθηκών για την παρακολούθηση των δεδομένων. Όταν εκτελείται ένας έλεγχος, αξιολογεί την καθορισμένη συνθήκη και παράγει μια κατάσταση (OK, Info, Warn ή Crit) για κάθε σημείο δεδομένων. Αυτές οι πληροφορίες κατάστασης μπορούν να χρησιμοποιηθούν για την οπτικοποίηση της υγείας του συστήματός ή για την ενεργοποίηση ειδοποιήσεων με βάση τη σοβαρότητα του προβλήματος.

Κανόνες ειδοποίησης

Οι κανόνες ειδοποίησης (notification rules) καθορίζουν τον τρόπο και τον χρόνο αποστολής ειδοποιήσεων με βάση την κατάσταση που παράγεται από τους ελέγχους. Μπορούν να ρυθμιστούν κανόνες ειδοποίησης για την αποστολή ειδοποιήσεων σε διάφορους προορισμούς, όπως το ηλεκτρονικό ταχυδρομείο, το Slack ή το PagerDuty, με βάση διαφορετικά κριτήρια όπως η σοβαρότητα, η συχνότητα ή η ώρα της ημέρας.

Παραδείγματα χρήσης σε εφαρμογές

IoT και δεδομένα αισθητήρων

Μία από τις πιο συνηθισμένες περιπτώσεις χρήσης της InfluxDB είναι η αποθήκευση και ανάλυση δεδομένων IoT και δεδομένων αισθητήρων. Καθώς ο αριθμός των συνδεδεμένων συσκευών συνεχίζει να αυξάνεται, αυξάνεται και ο όγκος των δεδομένων χρονοσειρών που παράγονται από αυτές τις συσκευές. Η InfluxDB μπορεί να αποθηκεύσει αποτελεσματικά και να υποβάλει ερωτήματα σε δεδομένα αισθητήρων από διάφορες πηγές, επιτρέποντας στους χρήστες να παρακολουθούν την απόδοση των συσκευών, να εντοπίζουν ανωμαλίες και να προβλέπουν τις ανάγκες συντήρησης.[41]

Παρακολούθηση επιδόσεων εφαρμογών (APM)

Η InfluxDB χρησιμοποιείται ευρέως για την παρακολούθηση της απόδοσης εφαρμογών (APM), όπου αποθηκεύει και αναλύει μετρήσεις που σχετίζονται με την απόδοση και την υγεία των εφαρμογών λογισμικού. Συλλέγοντας μετρικές όπως χρόνους απόκρισης, ποσοστά σφαλμάτων και χρήση πόρων. Οι προγραμματιστές και οι ομάδες λειτουργίας μπορούν να χρησιμοποιούν την InfluxDB για να εντοπίζουν τα σημεία συμφόρησης (bottleneck) της απόδοσης, να αποσφαλματώνουν προβλήματα (debugging) και να βελτιστοποιούν τις εφαρμογές τους για καλύτερη εμπειρία των χρηστών [45].

Παρακολούθηση δικτύου

Η παρακολούθηση δικτύου είναι ένας άλλος τομέας στον οποίο η InfluxDB βρίσκει εφαρμογή. Μπορεί να αποθηκεύσει και να επεξεργαστεί μεγάλους όγκους δεδομένων τηλεμετρίας δικτύου, όπως ρυθμούς κίνησης, καθυστέρηση και απώλεια πακέτων, για να βοηθήσει τους διαχειριστές δικτύου να εντοπίσουν προβλήματα, να βελτιστοποιήσουν την απόδοση του δικτύου και να σχεδιάσουν αναβαθμίσεις χωρητικότητας[45], [46].

7 Μελέτη Περίπτωσης

Στο παρών κεφάλαιο αναλύεται η ανάπτυξη ενός συστήματος/εφαρμογής που αντιμετωπίζει πληθώρα προκλήσεων του βιομηχανικού διαδικτύου των πραγμάτων (IIoT). Το σύστημα επεξεργάζεται μετρήσεις που παράγονται από συσκευές PLC. Η εφαρμογή θα πρέπει να μπορεί να λειτουργήσει ανεξαρτήτως από το είδος των μετρήσεων (πχ μετρήσεις που προέρχονται από σύστημα επιτήρησης ρομποτικών βραχιόνων σε εργοστάσιο ή μετρήσεις από αισθητήρες σε σύστημα ύδρευσης). Για να λάβει αυτές τις μετρήσεις το σύστημα επεξεργασίας, είναι απαραίτητη η επικοινωνία μέσω κάποιου βιομηχανικού πρωτοκόλλου επικοινωνίας (βλ. κεφάλαιο 2). Η εφαρμογή που παρουσιάζεται παρακάτω αποτελεί μια πρόταση-πρωτότυπο.

Προδιαγραφές - Στόχοι

Το σύστημα έχει ως στόχους:

1. Τη λειτουργία του συστήματος ανεξαρτήτως του πρωτοκόλλου επικοινωνίας που χρησιμοποιεί το σύστημα που παράγει τις μετρήσεις.
2. Τη δυνατότητα επεξεργασίας και αποθήκευσης μεγάλου όγκου δεδομένων σε ζωντανό χρόνο με αποδοτικό τρόπο.
3. Την επεξεργασία και αποθήκευση ανομοιογενών δεδομένων (πχ πίεση, θερμοκρασία, γωνία ενός ρομποτικού βραχίονα) από πολλαπλές πηγές.
4. Την ανοχή σε σφάλματα και καταστροφές. Παράδειγμα, εάν κάποιο από τα φυσικά μηχανήματα που συμμετέχουν στο σύστημα χαλάσει ή μια πηγή δεδομένων σταματήσει να λειτουργεί ή αρχίσει να παράγει λάθος δεδομένα, η λειτουργία του συστήματος θα πρέπει να συνεχιστεί απρόσκοπτα.

5. Την εύκολη παρακολούθηση της λειτουργίας του συστήματος (monitoring) και των πόρων που χρησιμοποιεί και η αποστολή ειδοποιήσεων σε έκτατες περιπτώσεις (alerting).
6. Την αποστολή μηνύματος για ανάληψη δράσης σε περίπτωση κινδύνου στο σύστημα που παράγει τις μετρήσεις. Για παράδειγμα, εάν η θερμοκρασία μιας συσκευής είναι υπερβολικά υψηλή, να στέλνεται ένα μήνυμα για να απενεργοποιηθεί η λειτουργία της.
7. Την εξαγωγή πληροφοριών χρήσιμων για την εκάστοτε εταιρεία/οργανισμό που κατέχει τα δεδομένα (business value). Για την ευκολότερη πρόσβαση σε αυτές τις πληροφορίες είναι απαραίτητη η κατασκευή διαγραμμάτων, πινάκων κτλπ που να ανανεώνονται σε ζωντανό χρόνο.

Περιορισμοί

Ένας βασικός περιορισμός αφορά την απουσία πραγματικών δεδομένων από συσκευές PLC. Οι μετρήσεις από αισθητήρες καθώς και η επικοινωνία με τις συσκευές PLC προσομοιώθηκαν με χρήση κατάλληλων εργαλείων. Έτσι, είναι δύσκολος ο διαχωρισμός μεταξύ πληροφοριών μείζονος και ελάσσονος σημασίας. Για παράδειγμα, σε ένα σύστημα μέτρησης της βροχόπτωσης πιθανώς να είναι προτιμότερο να αθροίζεται ο συνολικός όγκο νερού ανά ημέρα ενώ σε ένα σύστημα εποπτικού ελέγχου σε μια βιοτεχνία να ελέγχονται οι τιμές των συσκευών κάθε δευτερόλεπτο προκειμένου να αναληφθεί δράση εάν χρειαστεί. Η απουσία πραγματικού σεναρίου χρήσης οδήγησε στην λήψη αυθαίρετων αποφάσεων πάνω στο ποιές πληροφορίες είναι χρήσιμο να εξαχθούν και να απεικονιστούν.

Επιπλέον, λόγω περιορισμένων πόρων το σύστημα υλοποιήθηκε σε έναν υπολογιστή, ενώ οι τεχνολογίες που χρησιμοποιούνται αφορούν κατά βάση κατανεμημένα συστήματα. Οι λειτουργίες που αφορούν πολλαπλούς υπολογιστές προσομοιώθηκαν χρησιμοποιώντας Docker containers.

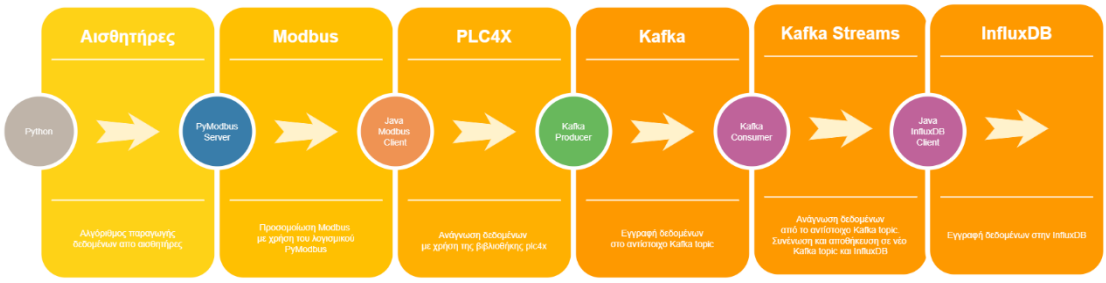
Πλαίσιο

Σε αυτήν την εργασία έγινε η υπόθεση πως υπάρχει μια βιομηχανική μονάδα παραγωγής με πολλαπλούς αισθητήρες που παράγουν μετρήσεις 3 ειδών: θερμοκρασία, πίεση και ισχύς. Ο κάθε αισθητήρας λαμβάνει μετρήσεις σε ζωντανό χρόνο από μια συσκευή που υπάρχει στη μονάδα παραγωγής. Συγκεκριμένα, έχουμε 4 αισθητήρες θερμοκρασίας και από 3 για την πίεση και την ισχύς. Η εργασία έχει κατασκευαστεί με τρόπο ο οποίος επιτρέπει την προσθήκη αισθητήρων στο σύστημα, συνεπώς ο αριθμός τους μπορεί να ποικίλλει έως από μερικές δεκάδες, έως και δεκάδες χιλιάδες. Περισσότερα για τον αριθμό των αισθητήρων θα αναλυθούν στη συνέχεια. Θεωρήθηκε πως κάθε αισθητήρας παράγει 1 μέτρηση κάθε δευτερόλεπτο. Συνεπώς, κάθε δευτερόλεπτο έχουμε τιμές ίσες με τον αριθμό των αισθητήρων.

Για την επικοινωνία μεταξύ των συσκευών, αλλά και εξωτερικών συστημάτων, χρησιμοποιείται το πρωτόκολλο Modbus. Το συγκεκριμένο πρωτόκολλο επιλέχθηκε λόγω της ευρείας αποδοχής που λαμβάνει από τη βιομηχανία. Παρόλα αυτά, θα μπορούσε να είχε επιλεχθεί οποιοδήποτε άλλο πρωτόκολλο χωρίς να αλλάξει η υλοποίηση του συστήματος στο ελάχιστο. Αυτό το πλεονέκτημα μας παρέχει η χρήση της βιβλιοθήκης PLC4X, η οποία προσφέρει ένα κοινό τρόπο επικοινωνίας, ανεξαρτήτως του βιομηχανικού πρωτοκόλλου που χρησιμοποιείται ή του τύπου του PLC [47].

Αρχιτεκτονική

Μια γενική εικόνα της αρχιτεκτονικής του συστήματος είναι η εξής: οι μετρήσεις παράγονται από Python κώδικα, ο οποίος προσομοιώνει τη λειτουργία των αισθητήρων. Οι μετρήσεις γράφονται σε έναν Modbus slave, ο οποίος προσομοιώνεται με χρήση του λογισμικού PyModbus. Στη συνέχεια, οι μετρήσεις διαβάζονται από Java κώδικα με χρήση του λογισμικού PLC4X και εισέρχονται στο Kafka. Από εκεί, το Kafka Streams διαβάζει τις τιμές για κάθε αισθητήρα, τις συνενώνει με βάση χρονικό παράθυρο (πχ τις μαζεύει κάθε 10 δευτερόλεπτα), και τις γράφει πίσω στο Kafka και στην InfluxDB. Στη συνέχεια κατασκευάζονται γραφήματα από τα δεδομένα. Για τα γραφήματα χρησιμοποιούνται τα εργαλεία οπτικοποίησης που παρέχει η InfluxDB, καθώς και το σύστημα Grafana. Επιπλέον, εάν οι τιμές ξεπεράσουν κάποιο όριο στέλνονται ειδοποιήσεις. Εάν η τιμή κάποιας μέτρησης είναι υπερβολικά υψηλή, τότε αυτόματα το Kafka Streams στέλνει σήμα και η συσκευή που παράγει τις μετρήσεις σταματά τη λειτουργία της. Επιπλέον, στέλνονται και ειδοποιήσεις σχετικές με την υγιή κατάσταση του συστήματος επεξεργασίας των μετρήσεων. Αυτές περιλαμβάνουν ειδοποιήσεις σχετικά με την κατάσταση των εξωτερικών συστημάτων, την υπερβολική χρήση υπολογιστικών πόρων, χώρου αποθήκευσης κτλπ. Στην εικόνα 7-1 βλέπουμε τη ροή των δεδομένων, περνώντας μέσα από τα διαφορετικά στάδια της εφαρμογής.



Εικόνα 7-1: Η ροή των δεδομένων

Ανάπτυξη με χρήση Docker containers

Όλα τα συστήματα γίνονται deploy με χρήση docker containers [48]. Τα πλεονεκτήματα είναι η ευκολία εγκατάστασης όλων των διαφορετικών συστημάτων, η ευκολία στο ξεκίνημα, τερματισμό και έλεγχο αυτών την συστημάτων, οδηγώντας σε γρηγορότερη ανάπτυξη. Επιπλέον, η εφαρμογή θα συμπεριφέρεται με τον ίδιο τρόπο σε όλα τα διαφορετικά περιβάλλοντα.

Προσομοίωση αισθητήρων

Ο αλγόριθμος που παράγει τα τυχαία δεδομένα λειτουργεί ως εξής:

- Αρχικά ορίζεται η μέση τιμή καθώς και η τυπική απόκλιση για κάθε αισθητήρα, όπως παρακάτω:

```
# Define the mean power and standard deviation
# for 3 power sensors
mean_power = [1000, 3000, 1700]
std_dev_power = [300, 1600, 500]
```

- Ορίζουμε μια παράμετρο «daily_seasonality», η οποία είναι μια ημιτονοειδής συνάρτηση με εύρος το χρονικό διάστημα που θέλουμε να προσομοιώσουμε. Προεπιλεγμένο εύρος είναι δεδομένα μιας ημέρας.

```
daily_seasonality_pressure = 2 * np.sin(2 * np.pi * np.arange(n)
/ 60 * 60 * 24)
```

Η τιμή 60 * 60 * 24 (86400) είναι ίση με τα δευτερόλεπτα που έχει μια μέρα, εφόσον θα παράξουμε 1 τιμή ανα δευτερόλεπτο για 1 ημέρα.

- Οι τιμές παράγονται τελικά από το συνδυασμό:
  Μέση τιμή + τυπική απόκλιση + τυχαίος θόρυβος + daily_seasonality.
  Επίσης προστίθεται μια τυχαία τιμή (θετική ή αρνητική) με πιθανότητα 1/10, για να προσομοιώσει εξωγενής παράγοντες.

Στην εικόνα 7-2 παρουσιάζονται τα αποτελέσματα από ένα τρέξιμο του αλγορίθμου.

Εικόνα 7-2: Οι τιμές που δημιουργεί ο αλγόριθμος για χρονικό εύρος 3 ημερών.

Ο παραπάνω αλγόριθμος έχει υλοποιηθεί σε Python script και τοποθετήθηκε σε ένα docker container με custom docker image. Κατά την εκκίνηση του container, οι τιμές ξεκινούν αυτόματα να παράγονται.

Modbus Client

Αφού δημιουργηθούν οι τιμές των μετρήσεων, στη συνέχεια γράφονται στο Modbus slave. Χρησιμοποιούνται οι καταχωρητές αποθήκευσης (holding registers), οι οποίοι περιγράφονται στο κεφάλαιο 2.2.1. Η τιμή κάθε αισθητήρα εισάγεται σε έναν καταχωρητή. Για να γνωρίζουμε τι τύπου τιμές αντιπροσωπεύει κάθε καταχωρητής (πιεση,θερμοκρασία,ισχύς), αποθηκεύονται πληροφορίες σε μια σχεσιακή βάση δεδομένων (Postgres), όπως αυτό περιγράφεται παρακάτω. Για χάριν ευκολίας της παρούσας υλοποίησης, οι μετρητές κάθε κατηγορίας μέτρησης είναι ομαδοποιημένοι. Έτσι, οι καταχωρητές με διεύθυνση 1-3 αφορούν μετρήσεις πίεσης, οι διευθύνσεις 4-7 θερμοκρασία, ενώ οι διευθύνσεις 8-10 ισχύς. Οι μονάδες μέτρησης είναι PSI, Celsius(°C), Watts αντίστοιχα. Για την προσθήκη περισσότερων αισθητήρων θα γίνει λόγος παρακάτω. Αυτός ο τρόπος αποθήκευσης επιτρέπει την προσθήκη έως και 2^{16} (65536) αισθητήρων, με την προσθήκη φυσικά των ανάλογων υπολογιστικών πόρων για να υποστηρίξουν αυτό τον όγκο δεδομένων.

Προσομοίωση Modbus Slave

Για την προσομοίωση του Modbus Slave χρησιμοποιείται το λογισμικό PyModbus. Συγκεκριμένα, έγινε χρήση της docker εικόνας *ghcr.io/pymodbus-dev/pymodbus:dev*.

Το PyModbus παρέχει τα απαραίτητα εργαλεία για τη δημιουργία Modbus clients και servers. Υποστηρίζει διαφορετικές παραλλαγές του πρωτοκόλλου, όπως Modbus RTU, Modbus ASCII και Modbus TCP/IP. Η βιβλιοθήκη έχει σχεδιαστεί ώστε να είναι εύκολη στη χρήση. Επιπλέον, είναι καλά τεκμηριωμένη και συμβατή με ένα ευρύ φάσμα εκδόσεων της Python.

Plc4x Modbus Client – Kafka Producer

Λαμβάνουμε από μια σχεσιακή βάση δεδομένων (Postgres) μια λίστα με όλους τους αισθητήρες. Σε αυτή τη λίστα περιλαμβάνονται και οι διευθύνσεις όπου βρίσκεται η τιμή τους. Όλες οι διευθύνσεις διαβάζονται και λαμβάνουμε τις τιμές για όλους τους μετρητές. Οι τιμές γράφονται σε ένα Kafka topic, ανάλογα με την κατηγορία μέτρησής τους (θερμοκρασία, πίεση, ισχύς). Συνεπώς, οι μετρήσεις από όλους τους αισθητήρες θερμοκρασίας καταλήγουν στο topic *streaming.input.temperatureMeasurements*, οι μετρήσεις των αισθητήρων πίεσης στο *streaming.input.pressureMeasurements* κ.ο.κ. Τα Kafka topics έχουν ρυθμιστεί ώστε να έχουν αριθμό partitions ίσο με το συνολικό αριθμό των αισθητήρων κάθε μέτρησης.

Το κλειδί της κάθε μέτρησης είναι ένας ακέραιος αριθμός που αντικατοπτρίζει τον σειριακό αριθμό του αισθητήρα κάθε κατηγορίας. Πχ για την κατηγορία «πίεση» έχουμε τους μετρητές «pressure-0», «pressure-1», «pressure-2». Συνεπώς κλειδί θα είναι ένας αριθμός εκ των {0,1,2}. Αντίστοιχα και για τις άλλες κατηγορίες. Με αυτόν τον τρόπο οι μετρήσεις κάθε αισθητήρα γράφονται σε διαφορετικό partition.

Kafka

Καθώς έχουμε δεδομένα από πολλαπλές πηγές (αισθητήρες) με ανομοιογενή σχήμα και μεγάλο όγκο πληροφορίας, επιλέχθηκε το Apache Kafka. Τα πλεονεκτήματα της ανάπτυξης με το Kafka αναλύονται διεξοδικά στο κεφάλαιο 3.

Ως διανομή για το Apache Kafka επιλέχθηκε η διανομή της εταιρείας Confluent.

Η διανομή της Confluent για το Apache Kafka είναι μια πλατφόρμα έτοιμη για επιχειρήσεις. Επεκτείνει το Apache Kafka με πρόσθετα στοιχεία και εργαλεία, επιτρέποντας την καλύτερη διαχείριση, παρακολούθηση και ασφάλεια. Βασικά συστατικά της διανομής της Confluent είναι τα εξής:

Server: Ο Confluent Server είναι μια βελτιωμένη έκδοση του Apache Kafka. Προσφέρει χαρακτηριστικά όπως έλεγχο πρόσβασης βάσει ρόλων, κλιμακωτή αποθήκευση και cluster που εξισορροπούνται αυτόματα.

Schema Registry: Βοηθά στη διαχείριση και αποθήκευση των σχημάτων που χρησιμοποιούνται από τους παραγωγούς και τους καταναλωτές, εξασφαλίζοντας τη συμβατότητα των δεδομένων και μειώνοντας τον κίνδυνο από τις αλλαγές σχήματος. Υποστηρίζει μορφές σειριοποίησης Avro, JSON Schema και Protocol Buffers.

Κέντρο ελέγχου: Εργαλείο διαχείρισης και παρακολούθησης. Το Κέντρο Ελέγχου επιτρέπει στους χρήστες να διαχειρίζονται και να παρακολουθούν τους Kafka Clusters, να ρυθμίζουν ειδοποιήσεις και να απεικονίζουν βασικές μετρήσεις απόδοσης.

ksqlDB: Η ksqlDB είναι μια κατανεμημένη βάση δεδομένων ροής γεγονότων που έχει δημιουργηθεί πάνω στο Kafka. Επιτρέπει στους χρήστες να εκτελούν επεξεργασία δεδομένων σε πραγματικό χρόνο χρησιμοποιώντας ερωτήματα που μοιάζουν με SQL.

Connectors: Διατίθεται μια σειρά από προκατασκευασμένους connectors για την ενσωμάτωση του Kafka με διάφορες πηγές δεδομένων, όπως βάσεις δεδομένων, υπηρεσίες cloud και άλλα συστήματα ανταλλαγής μηνυμάτων.

REST Proxy: Το REST Proxy παρέχει μια διεπαφή RESTful για την αλληλεπίδραση με τους Kafka Clusters, καθιστώντας εύκολη την ανταλλαγή μηνυμάτων από πελάτες που δεν χρησιμοποιούν Java.

Για την εγκατάσταση της διανομής, χρησιμοποιήθαν τα docker images που προτείνει η Confluent και συγκεκριμένα ένα docker-compose.yml αρχείο που περιέχει όλες τις εικόνες απαραίτητες για τη διανομή (https://github.com/confluentinc/cp-all-in-one).

Κατά τη διάρκεια της ανάπτυξης χρησιμοποιήθηκε 1 Kafka Broker, λόγω περιορισμένων υπολογιστών πόρων. Έγιναν όμως και πειραματισμοί με μεγαλύτερο αριθμό brokers. Για την αύξηση των Kafka Brokers στο περιβάλλον ανάπτυξης προστέθηκαν περισσότεροι containers από την docker εικόνα του broker και ρυθμίστηκαν όλες οι υπηρεσίες ώστε να επικοινωνούν με αυτές τους νέους brokers (παράμετρος bootstrap-servers). Σε ένα παραγωγικό περιβάλλον είναι απαραίτητη η αύξηση των Kafka Brokers σε τουλάχιστον τρεις, προκειμένου το σύστημα να είναι ανθεκτικό. Επίσης, συστήνεται η χρήση σύγχρονων σκληρών δίσκων τύπου SSD. Αυτά αναλύονται διεξοδικά στο κεφάλαιο 3.3.

Περαιτέρω παραμετροποίηση για το Apache Kafka γίνεται καθώς η εφαρμογή επεξεργασίας (αναλύεται παρακάτω) ξεκινά. Κατά το ξεκίνημα, ορίζονται ποικίλλες παράμετροι, χρησιμοποιώντας το Kafka Admin Java API. Αυτές οι παράμετροι είναι εύκολο να ρυθμιστούν, καθώς πρόκειται για key-value pairs σε ένα yaml configuration αρχείο της εφαρμογής (εικόνα 7-3).

Εικόνα 7-3: Ρυθμίσεις για το Apache Kafka

Καθώς η docker εικόνα για το Kafka Connect δεν περιλαμβάνει connectors, υλοποιήθηκε μια εικόνα ειδικά για τις ανάγκες της εφαρμογής, προκειμένου να περιλαμβάνεται connector για τη βάση InfluxDB. Το Dockerfile που παράγει αυτήν την εικόνα είναι το εξής:

```
1 FROM confluentinc/cp-kafka-connect:7.3.0
2 ENV CONNECT_PLUGIN_PATH: "/usr/share/java,/usr/share/confluent-hub-
components"
3 RUN confluent-hub install --no-prompt confluentinc/kafka-connect-
influxdb:1.2.6
```

Επεξεργασία μηνυμάτων

Ανάπτυξη με Spring Boot

Η επεξεργασία των μηνυμάτων γίνεται με χρήση του Kafka Streams, μέσα όμως από ένα Spring Boot application που χρησιμοποιεί τη βιβλιοθήκη Kafka Streams και τη βιβλιοθήκη Spring Kafka. Αυτός ο τρόπος ανάπτυξης προσθέτει ένα ακόμα επίπεδο πολυπλοκότητας στον κώδικα, παρέχει όμως ισχυρά πλεονεκτήματα που κρίθηκαν σημαντικά. Αυτά είναι τα εξής:

Configuration: Το Spring Boot παρέχει αυτόματο configuration, βοηθώντας στη διαμόρφωση ροών Kafka Streams μέσα από configuration αρχεία. Παρέχεται επίσης υποστήριξη για διαφορετικά περιβάλλοντα όπως ανάπτυξης(dev, prod κ.α.).

Οικοσύστημα Spring: Το Spring Boot και το Spring Kafka μπορούν εύκολα να ενσωματωθούν με άλλα έργα Spring. Συγκεκριμένα έχει χρησιμοποιηθεί επίσης το Spring Data, το Spring Security και το Spring Boot Admin.

Testing: Το Spring Boot προσφέρει ενσωματωμένη υποστήριξη για testing σε Kafka και Kafka Streams εφαρμογές.

Δηλωτικό μοντέλο προγραμματισμού: Το Spring Kafka παρέχει ένα υψηλού επιπέδου, δηλωτικό μοντέλο προγραμματισμού που αφαιρεί την πολυπλοκότητα.

Ενισχυμένος χειρισμός σφαλμάτων: Το Spring Kafka προσφέρει βελτιωμένες δυνατότητες χειρισμού σφαλμάτων, όπως retries, error handling strategies, and dead letter queues.

Παρακολούθηση και διαχείριση: Το Spring Boot παρέχει το Actuator, το οποίο δημιουργεί endpoints για την παρακολούθηση και τη διαχείριση της εφαρμογής.

Ανάλυση επεξεργασίας μηνυμάτων

Κάθε δευτερόλεπτο παράγεται 1 τιμή για κάθε μετρητή. Στο Kafka Streams, ρυθμίστηκε η συγκέντρωση (aggregation) των μηνυμάτων με βάση ένα χρονικό παράθυρο. Προεπιλεγμένο εύρος του χρονικού παραθύρου ορίστηκαν τα 10 δευτερόλεπτα.

Η κάθε κατηγορία μέτρησης απαιτεί ξεχωριστό χειρισμό. Για την θερμοκρασία και την πίεση κρίθηκε λογικό να υπολογίζεται η μέση τιμή (mean), ενώ για την ισχύς προστίθενται όλες οι τιμές (sum) εντός του εύρους του χρονικού παραθύρου. Για το λόγω αυτό, εκκινούνται 3 στιγμιότυπα του Kafka Streams, 1 για κάθε κατηγορία μέτρησης. Το κάθε στιγμιότυπο διαβάζει μηνύματα από το αντίστοιχο topic. Καθώς το κάθε στιγμιότυπο Kafka Streams είναι πλήρως ανεξάρτητο από τα υπόλοιπα, αυτό επιτρέπει να εφαρμόζεται οποιαδήποτε λογική στην επεξεργασία, ανάλογα με τις ανάγκες. Η ίδια μεθοδολογία μπορεί να ακολουθηθεί για την περίπτωση όπου έχουμε περισσότερους τύπους μετρήσεων, προσθέτωντας απλώς περισσότερα Kafka Streams στιγμιότυπα, topics και υλικό (εάν χρειάζεται).

Τα topics από τα οποία διαβάζει το Kafka Streams είναι:

1. streaming.input.temperatureMeasurements
2. streaming.input.pressureMeasurements
3. streaming.input.powerMeasurements

Κάθε στιγμιότυπο του Kafka streams λαμβάνει ζεύγη κλειδιών-τιμών, όπου κλειδί είναι το μοναδικό αναγνωριστικό ενός αισθητήρα μιας κατηγορίας. Πχ στην κατηγορία «temperature» λαμβάνονται τιμές: {0: 57.7}, {1: 22.1}, {0: 58.1}, {2: 102.8}, {0: 58.2}

Οι τιμές ομαδοποιούνται (group by) ανά κλειδί. Στη συνέχεια παράγεται το άθροισμα και ο συνολικός αριθμός, ανά κλειδί, για τις κατηγορίες θερμοκρασίας και πίεσης, ενώ για την κατηγορία ισχύς μόνον το άθροισμα ανά κλειδί.

Τέλος, οι επεξεργασμένες τιμές αποθηκεύονται σε νέα Kafka topic, καθώς και στην InfluxDB.

Τα topics στα οποία το Kafka Streams γράφει είναι:

1. streaming.output.temperatureMeasurements
2. streaming. output.pressureMeasurements
3. streaming. output.powerMeasurements

Κατά τη διάρκεια της ανάπτυξης ο συντελεστής αντιγραφής (replication factor) για όλα τα topic ρυθμίστηκε ίσος με 1. Σε ένα παραγωγικό περιβάλλον αυτός ο συντελεστής θα πρέπει να είναι τουλάχιστον 3, όπως γίνεται σαφές από το κεφάλαιο 3.3

Η InfluxDB παρέχει ενσωματωμένα εργαλεία για απεικόνιση δεδομένων. Κατασκευάστηκε το dashboard της εικόνας 7-4. Περιέχει γραφήματα από τα δεδομένα που εγγράφονται στην Influx, καθώς και 3 μετρητές. Τα γραφήματα απεικονίζουν τις τιμές μιας κατηγορίας μέτρησης για όλους τους αντίστοιχους αισθητήρες. Οι μετρητές για τις κατηγορίες πίεσης και θερμοκρασίας μας δείχνουν την μέση τιμή για την τελευταία 1 ώρα, ενώ για την κατηγορία της ισχύς μας δείχνει το συνολικό άθροισμα της τελευταίας 1 ώρας.

Εικόνα 7-4: Τα δεδομένα που εγγράφονται στην InfluxDB

Εάν κάποιο μήνυμα, για οποιοδήποτε λόγο, δε μπορεί να επεξεργαστεί από το Kafka Streams (πχ τα δεδομένα έφτασαν αλλιωμένα), τότε αυτό το μήνυμα αποθηκεύεται σε ένα ειδικό θέμα Kafka, για περαιτέρω χειρισμό ή επεξεργασία. Αυτή η μεθοδολογία χειρισμού λανθασμένων μηνυμάτων ονομάζεται «Dead Letter Queue». Επίσης, η πιο πρόσφατη τιμή για κάθε αισθητήρα εγγράφεται επίσης στο σύστημα Redis, για λόγους βελτιστοποίησης της απόδοσης.

Logging

Τα logs της εφαρμογής αποθηκεύονται σε αρχεία. Χρησιμοποιείται το εργαλείο logback, το οποίο είναι ενσωματωμένο στο Spring Boot. Για τη ρύθμιση των logs χρησιμοποιείται το αρχείο logback.xml, στο οποίο καθορίζεται η μορφή των logs. Επίσης καθορίζεται το όνομα του latest log αρχείου, καθώς και το όνομα που φακέλου που αυτό θα αποθηκεύεται («./logs»).

Κάθε μέρα το log αρχείο συμπιέζεται και μεταφέρεται στον φάκελο «/logs/archived", αφού του προστεθεί το επίθεμα {dd-MM-yyyy}, δηλαδή τη σημερινή ημερομηνία. Επίσης, εάν ξεπεράσει το όριο των 10MB που έχει τεθεί ως ανώτατο, τότε ακολουθείται η ίδια διαδικασία. Εάν πολλαπλά αρχεία ξεπεράσουν το όριο μεγέθους, τότε τους προστίθεται ως επίθεμα ένας αύξων αριθμός (1,2,3…).

Οι ρυθμίσεις που περιγράφονται παραπάνω είναι οι εξής:

```
1       <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
2           <fileNamePattern>${LOG_PATH}/archived/${log.name}_%d{dd-MM-yyyy}-
%i.log.gz</fileNamePattern> <!-- daily rollover -->
3           <maxFileSize>10MB</maxFileSize>
4           <maxHistory>10</maxHistory> <!-- keep 10 days of logs -->
5           <totalSizeCap>100MB</totalSizeCap> <!-- 100MB -->
6       </rollingPolicy>
```

Επίσης, μέσα στο αρχείο logback.xml έχουν δηλωθεί διαφορετικές ρυθμίσεις για περιβάλλον ανάπτυξης και διαφορετικές για το παραγωγικό περιβάλλον. Το περιβάλλον ρυθμίζεται χρησιμοποιώντας τα spring profiles. Οι επιλογές είναι: «dev» και «prod».

Εικόνα 7-5: Κατά την εκκίνηση της εφαρμογής, εκτυπώνεται το μήνυμα της εικόνας

Παρακολούθηση απόδοσης (Monitoring)

Η παρακολούθηση απόδοσης μπορεί να γίνει σε 2 επίπεδα: στο (εξωγενές) σύστημα που παρέχει τις μετρήσεις, καθώς και στο παρόν σύστημα επεξεργασίας των δεδομένων. Στο σύστημα που παρέχει τις μετρήσεις (αισθητήρες) παρακολουθούνται τα δεδομένα και η ποιότητα αυτών. Στο σύστημα επεξεργασίας παρακολουθούνται μετρικές όπως η κίνηση του δικτύου, ο διαθέσιμος αποθηκευτικός χώρος, η μνήμη που δεσμεύει το σύστημα, η επεξεργαστική ισχύς που καταναλώνεται, η καθυστέρηση (εάν υπάρχει) στην επεξεργασία των δεδομένων από και προς το Kafka, τα logs της εφαρμογής. Επίσης παρακολουθείται η κατάσταση των συσχετιζόμενων συστημάτων, όπως Kafka, InfluxDB κ.ο.κ.

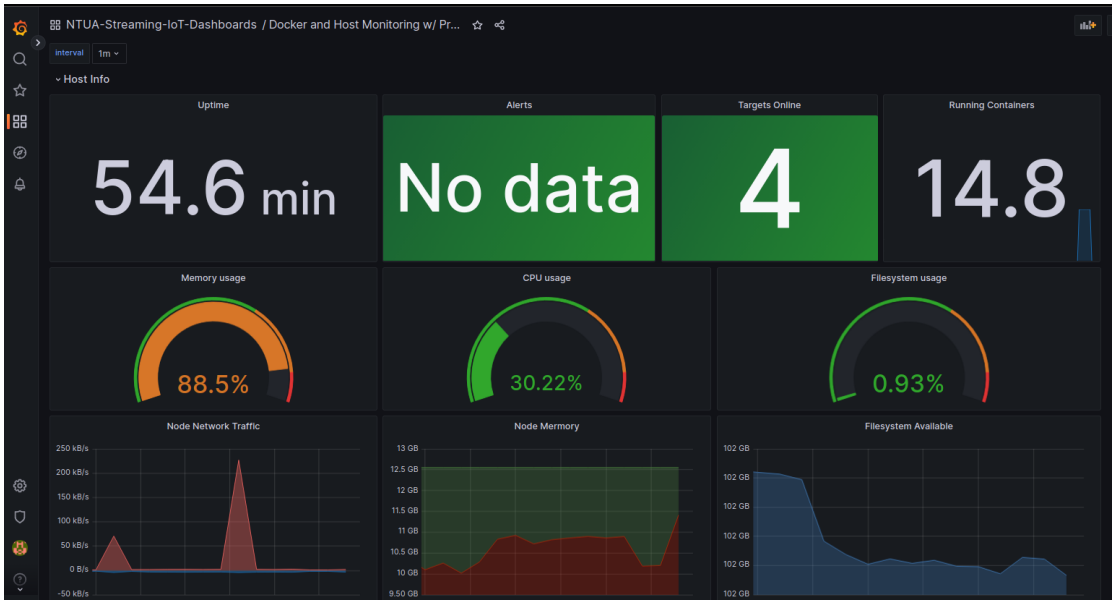
Για την παρακολούθηση της απόδοσης χρησιμοποιούνται τα εξωτερικά συστήματα Prometheus, cAdvisor, Node Exporter, Loki και Grafana. Το Prometheus είναι ένα εργαλείο παρακολούθησης και ειδοποιήσεων ανοιχτού κώδικα. Μπορεί να συγκεντρώνει μετρικές από άλλα συστήματα. Το cAdvisor (πλήρες όνομα: Container Advisor) συγκεντρώνει μετρικές από containers. Οι containers μπορεί να είναι Docker containers ή και άλλου τύπου. Το Node Exporter συγκεντρώνει μετρικές από Linux συστήματα (cpu, ram, disk κ.α.). Το Loki χρησιμοποιείται για να συγκεντρώσει logs από 1 ή περισσότερες εφαρμογές. Τέλος, το Grafana είναι ένα εργαλείο οπτικοποίησης.

Επιπλέον, χρησιμοποιείται το Actuator, ενσωματωμένο εργαλείο του Spring Boot. Το actuator μαζεύει μετρικές από/για την εφαρμογή και τις διαθέτει μέσω ενός endpoint.

Το Prometheus συγκεντρώνει μετρικές από όλα τα προαναφερθέντα συστήματα και στη συνέχεια το Grafana τις παίρνει προκειμένου να κατασκευάσουμε γραφήματα σε ζωντανό χρόνο.

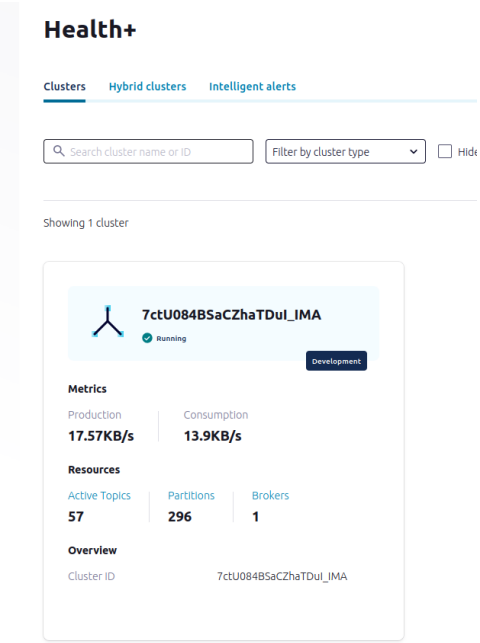
Κατασκευάστηκαν τα εξής 2 dashboard για την παρακολούθηση των μετρικών από τους docker containers (εικόνες 7-6, 7-7).

Εικόνα 7-6: Dashboard για παρακολούθηση Docker containers

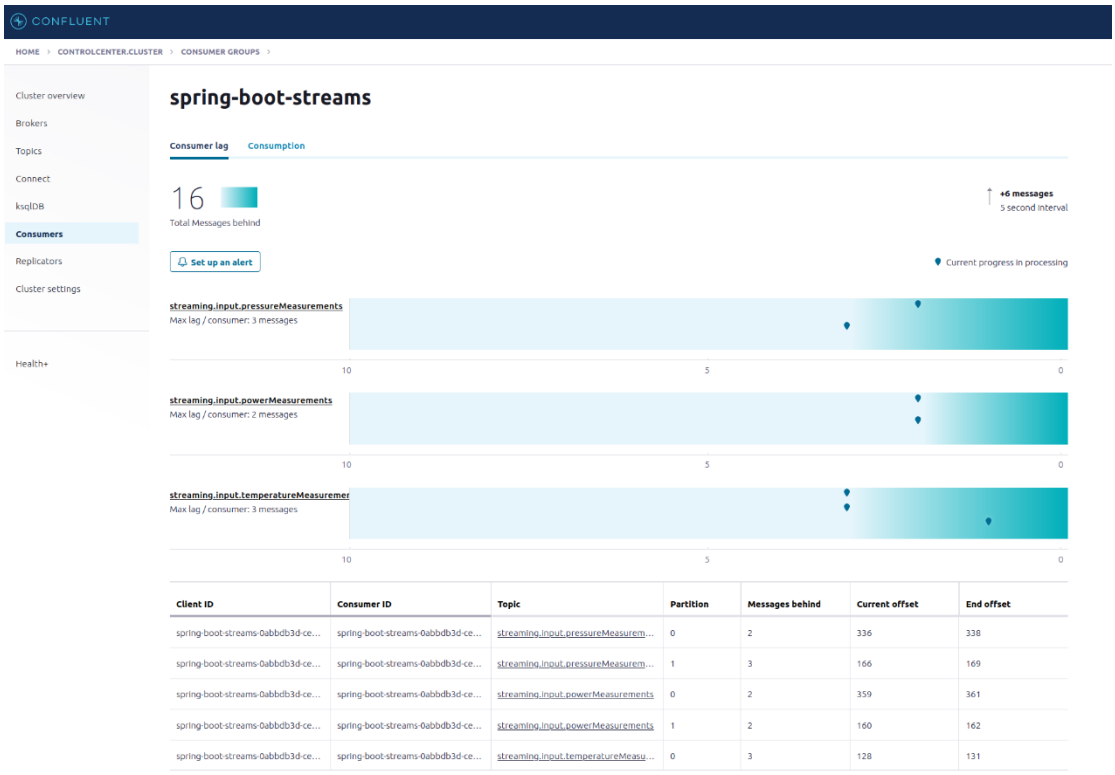


Εικόνα 7-7: Dashboard για παρακολούθηση Docker containers. Η επιγραφή «No data» αναφέρεται σε απουσία πρόσφατων ειδοποιήσεων (alerts), και όχι σε κάποιο πρόβλημα με το σύστημα.

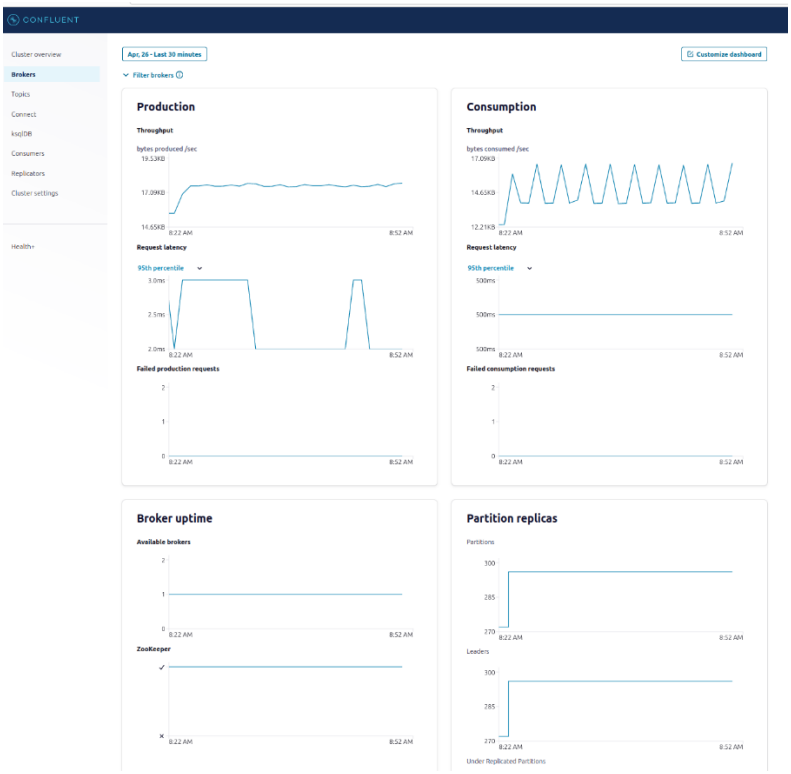
Για την παρακολούθηση της απόδοσης στο Apache Kafka χρησιμοποιήθηκε το σύστημα Control Center της διανομής Confluent. Το σύστημα αυτό περιλαμβάνει ενσωματωμένα εργαλεία παρακολούθησης της απόδοσης. Αυτά παρουσιάζονται στις εικόνες 7-8, 7-9, 7-10 και 7-11.



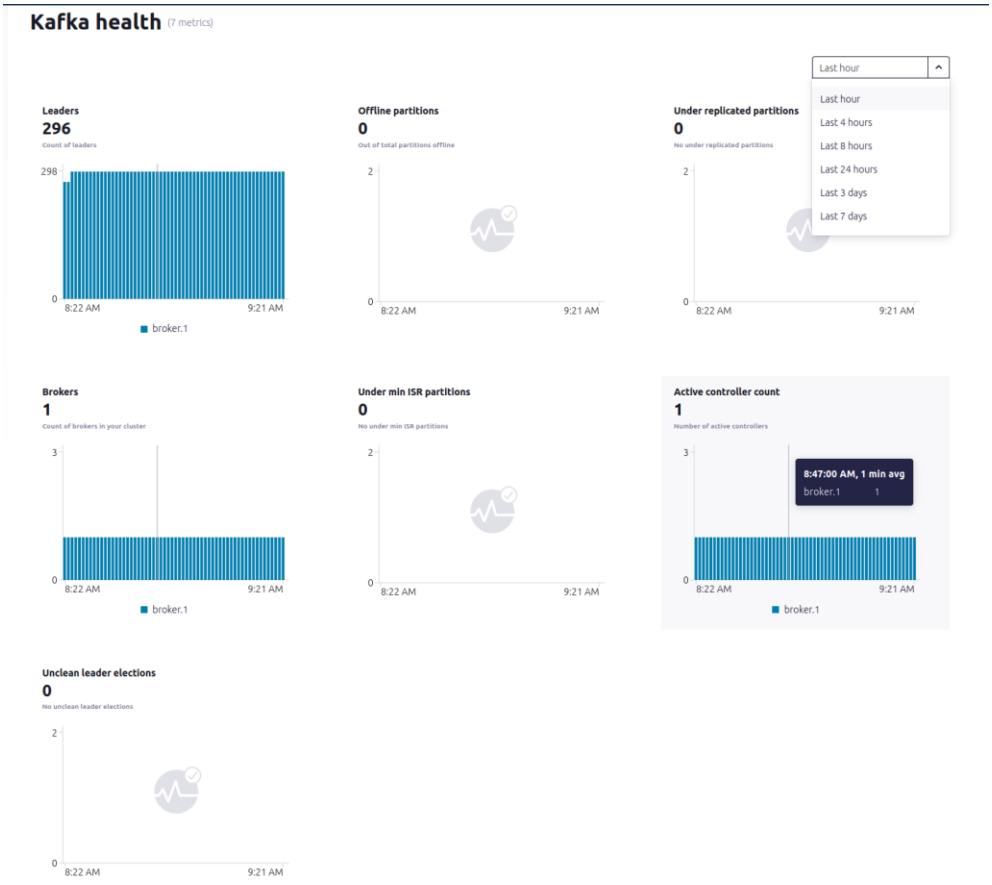
Εικόνα 7-8: Οι διαθέσιμοι Kafka clusters, όπως παρουσιάζονται στο Control Center



Εικόνα 7-9: Παρακολούθηση των καταναλισκόμενων μηνυμάτων

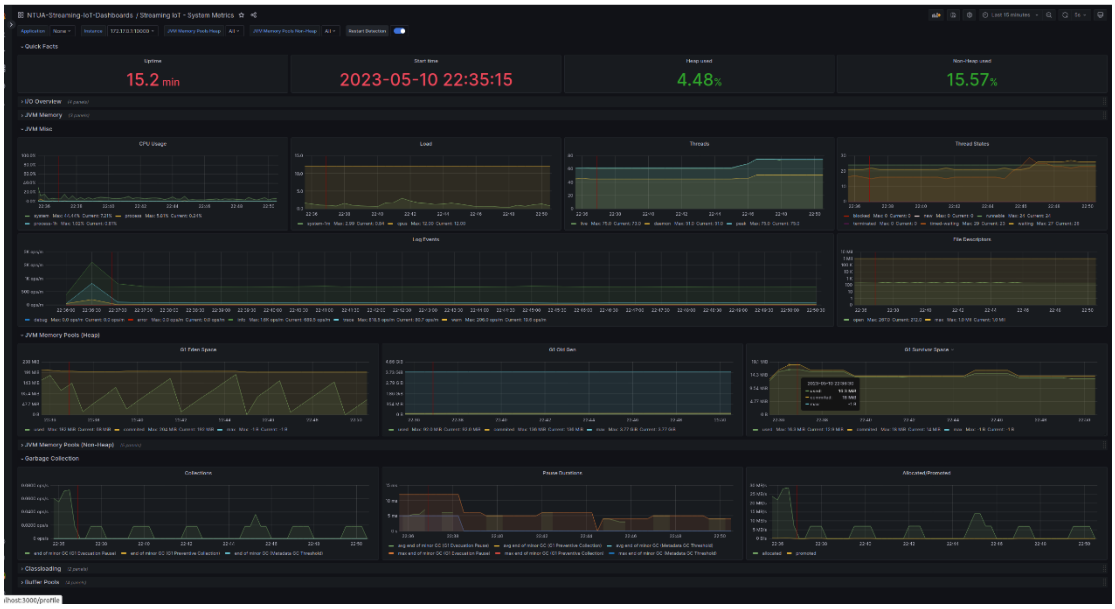


Εικόνα 7-10: Παρακολούθηση μετρικών για το Apache Kafka

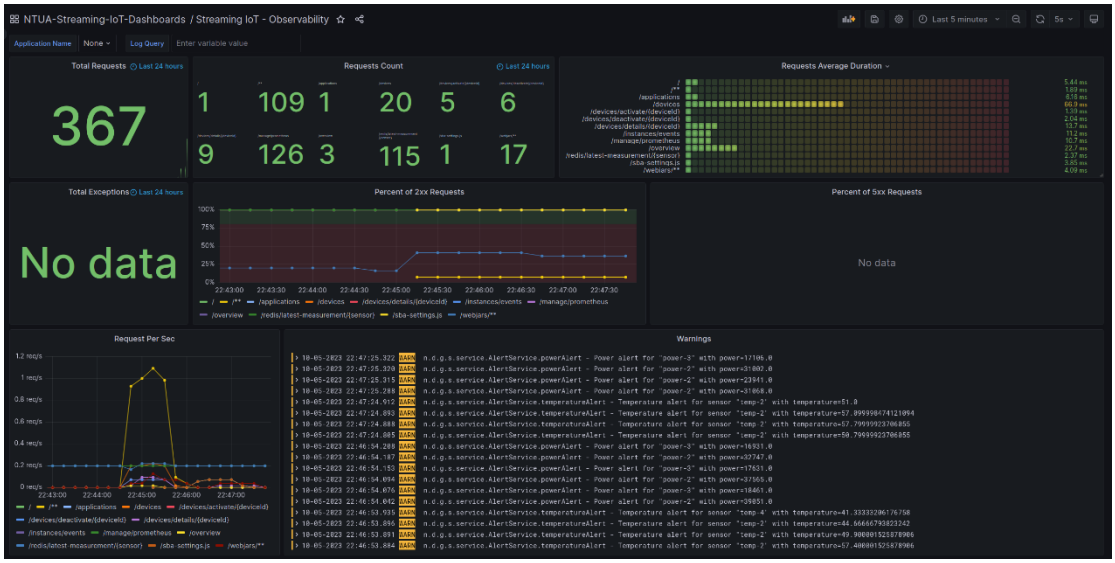


Εικόνα 7-11: Πληροφορίες για την κατάσταση του συστήματος Apache Kafka

Για την παρακολούθηση της Spring Boot εφαρμογής, κατασκευάστηκαν στο Grafana τα dashboards των εικόνων 7-12 και 7-13.



Εικόνα 7-12: Μετρικές συστήματος από την Spring Boot εφαρμογή. Πληροφορίες για: κίνηση δικτύου, κίνηση Εισόδου-Εξόδου, μνήμη JVM (στοίβα, σωρός), συλλογή σκουπιδιών, κλάσεις της εφαρμογής, cpu, νήματα.

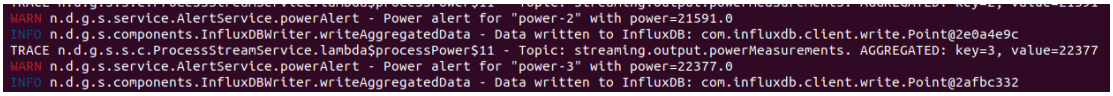


Εικόνα 7-13: Πληροφορίες για τα αιτήματα που δέχεται η εφαρμογή. Συνολικός αριθμός αιτημάτων, αιτήματα ανά endpoint, χρόνος απόκρισης ανά endpoint, γράφημα με αιτήματα ανά δευτερόλεπτο, πίνακας με τα logs της εφαρμογής που έχουν κατηγορία WARN και πάνω. Το κουτί όπου αναγράφεται «No data» αφορά τον αριθμό των εξαιρέσεων στο επιλεγμένο χρονικό διάστημα.

Ειδοποιήσεις (Alerting)

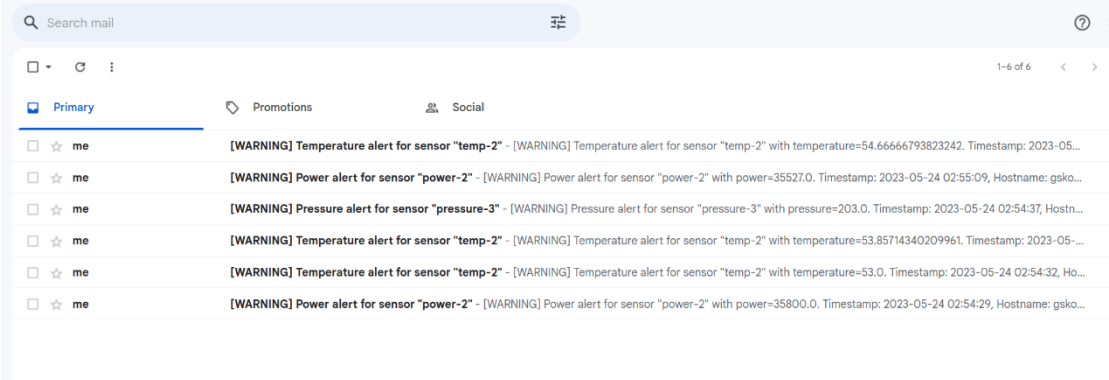
Alerts στο σύστημά μπορούν να προέλθουν από διαφορετικές πηγές, καθώς πολλά διαφορετικά συστήματα μπορεί να εμφανίσουν πρόβλημα ή ακόμα και να αποτύχουν.

Κατά τη διάρκεια της επεξεργασίας των μηνυμάτων από το Kafka Streams, εάν κάποια τιμή είναι υψηλή τότε εμφανίζεται στα logs ένα μήνυμα WARNING. Εάν κάποια τιμή είναι υπερβολικά υψηλή, τότε η κατάσταση θεωρείται κρίσιμη για το σύστημα των μετρήσεων. Εκτυπώνεται ένα μήνυμα επιπέδου «Error» και αποστέλλεται ένα σήμα για την απενεργοποίηση κάποιας συσκευής.



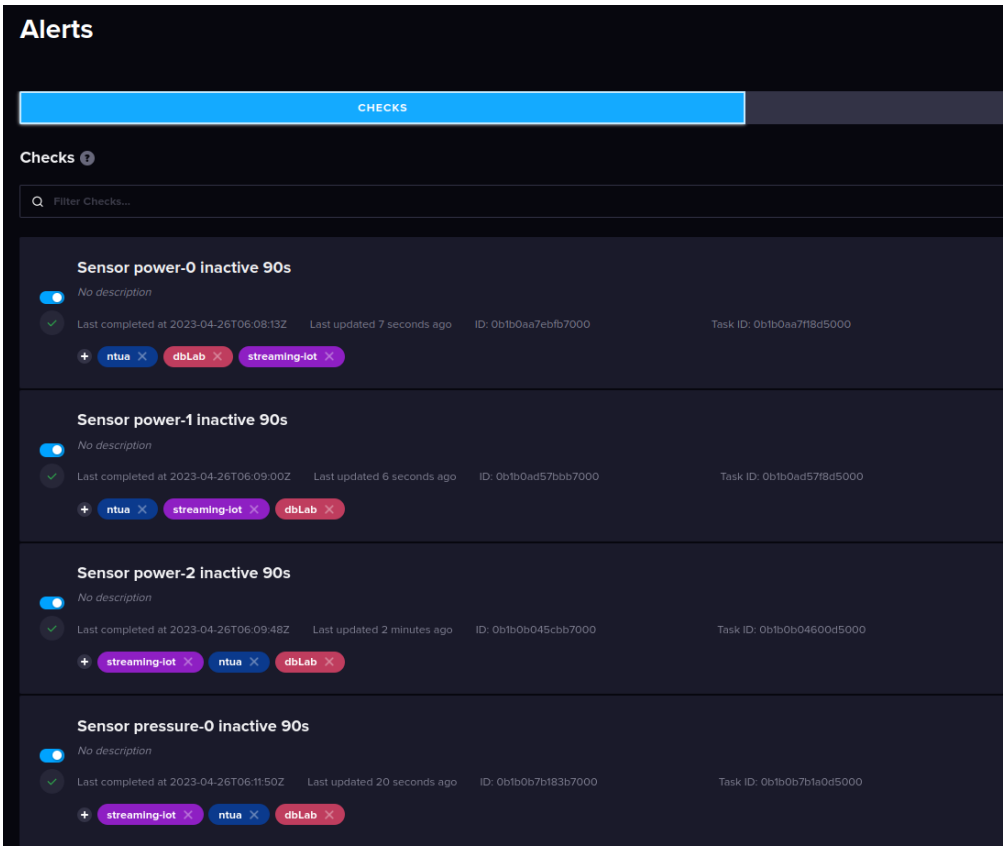
Εικόνα 7-14: Warnings στα logs

Επίσης, έχει υλοποιηθεί λειτουργικότητα για την αυτόματη αποστολή e-mails. Mails αποστέλλονται για κάθε warning και error που ειδοποίηση που εμφανίζεται στα logs, όπως περιγράφεται παραπάνω. Για την αποστολή e-mails δημιουργήθηκε ένα gmail (με όνομα streamingiot@gmail.com) ειδικά για τις ανάγκες της εφαρμογής, προκειμένου να χρησιμοποιηθεί ως smtp server. Το gmail επιλέχθηκε για την ανάπτυξη της εφαρμογής διότι παρέχει έως και 500 e-mails ημερησίως δωρεάν. Στο παραγωγικό περιβάλλον μπορεί να αντικατασταθεί από κανονικό smtp server ή άλλη υπηρεσία.



Εικόνα 7-15: Ειδοποιήσεις μέσω e-mail

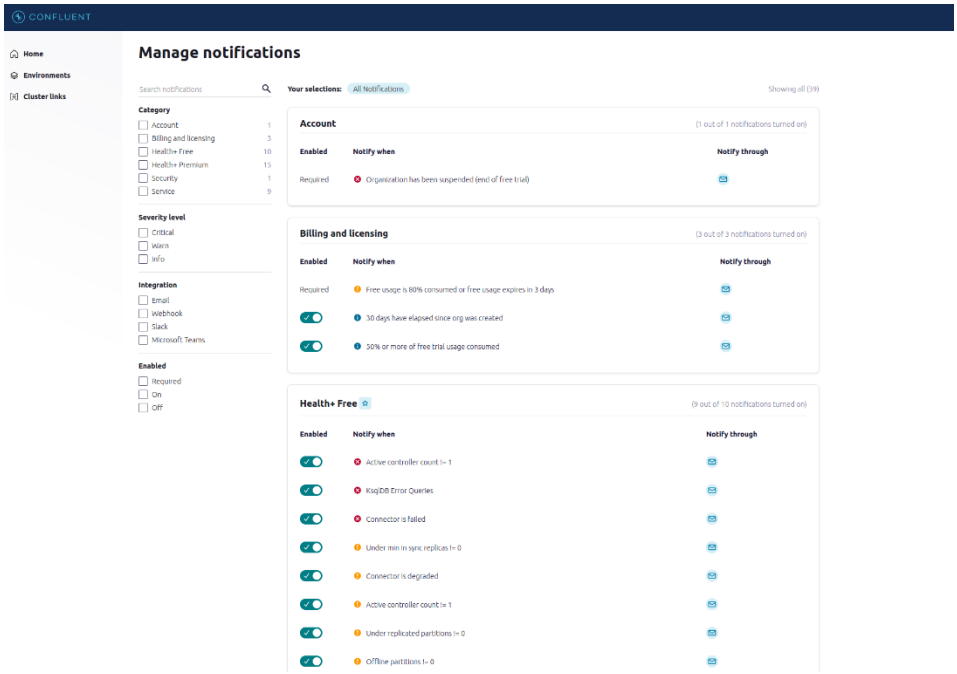
Επίσης, alerts μπορεί να προκύψουν και από την InfluxDB. Αν για αρκετή ώρα δε λαμβάνονται μετρήσεις από κάποιο αισθητήρα, τότε έχει ρυθμιστεί η InfluxDB ώστε να στέλνει μια ειδοποίηση στη Spring Boot εφαρμογή. Η ειδοποίηση αποστέλλεται μέσω HTTP αιτήματος. Έχει υλοποιηθεί στη Spring Boot εφαρμογή ένα endpoint προκειμένου να είναι δυνατή η αποστολή alerts από εξωτερικά συστήματα. Το Spring Boot λαμβάνει την ειδοποίηση και είναι ευθύνη του να τη χειριστεί, ανάλογα με τις ανάγκες της κάθε περίπτωσης.



Εικόνα 7-16: Ειδοποιήσεις που έχουν ρυθμιστεί στην InfluxDB. Εάν για 90 δευτερόλεπτα δεν έχει ληφθεί μέτρηση από έναν αισθητήρα, τότε μια ειδοποίηση στέλνεται στη Spring Boot εφαρμογή

Alerts παράγονται επίσης από το Control Panel (εικόνα 7-16). Το Control Panel παρακολουθώντας το Apache Kafka προσφέρει εργαλεία για την αποστολή ειδοποιήσεων σε μια ευρεία γκάμα από επιλογές, όπως e-mail, PagerDuty, Slack channel, MS Teams channel κ.α.

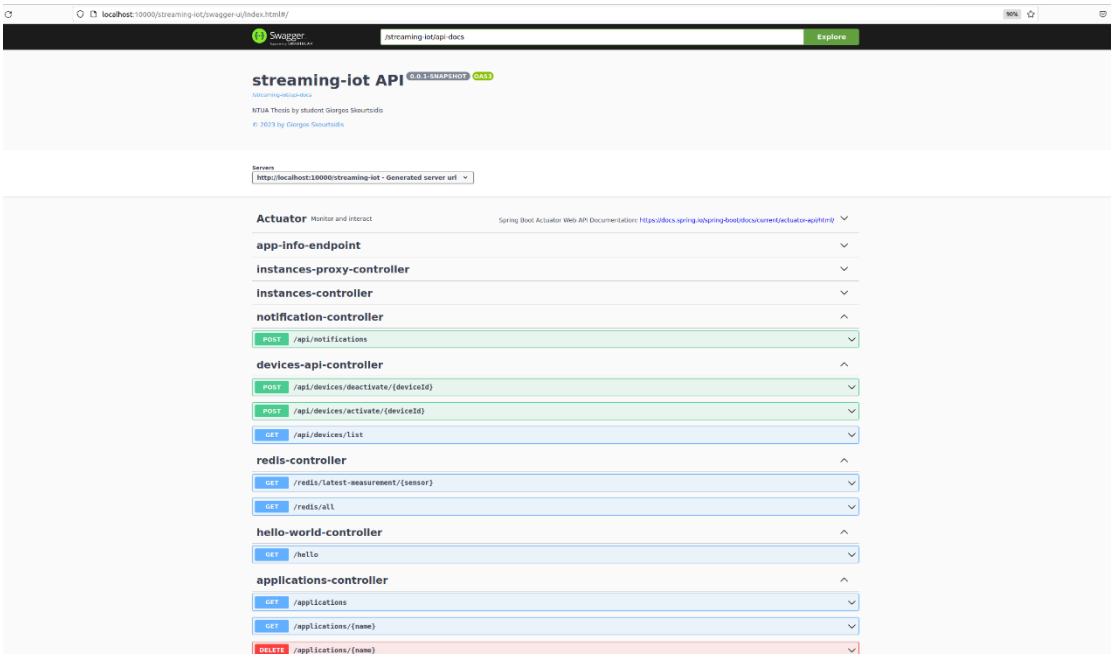
Στην τωρινή εφαρμογή επιλέχθηκε το e-mail.



Εικόνα 7-17: Ειδοποιήσεις μέσω του Control Center

Διαχείριση του συστήματος

Για τις ανάγκες τις διαχείρισης του συστήματος, χρησιμοποιείται το Swagger UI. Πρόκειται για ένα εργαλείο που συγκεντρώνει όλα τα endpoints της Spring Boot εφαρμογής. Ο χρήστης μπορεί να επιβλέψει και να χρησιμοποιήσει τα endpoints της εφαρμογής, λαμβάνοντας χρήσιμες πληροφορίες ή ακόμα και ενέργειες. Επιπλέον, παρέχονται endpoints με πληροφορίες για την Spring Boot εφαρμογή. Μπορούμε με χρήση http request να λάβουμε την έκδοση (version) της εφαρμογής, πληροφορίες για το git repository (commit id, όνομα χρήστη,date κ.α.) από το οποίο έγινε build το jar που τρέχει το spring boot. Επίσης μπορούμε να λάβουμε πληροφορίες για την υγεία της εφαρμογής, το χρόνο όπου αυτή εκκινήθηκε, τα ονόματα των ενεργών κλάσεων της εφαργμογής, μετρικές, λίστα με τις συσκευές καθώς και αν αυτές είναι ενεργές και άλλα. Μπορεί να γίνει έλεγχος και διαχείριση της εφαρμογής χρησιμοποιώντας μόνο αιτήματα http. Στην εικόνα 7-17 παρουσιάζεται το Swagger UI της εφαρμογής.

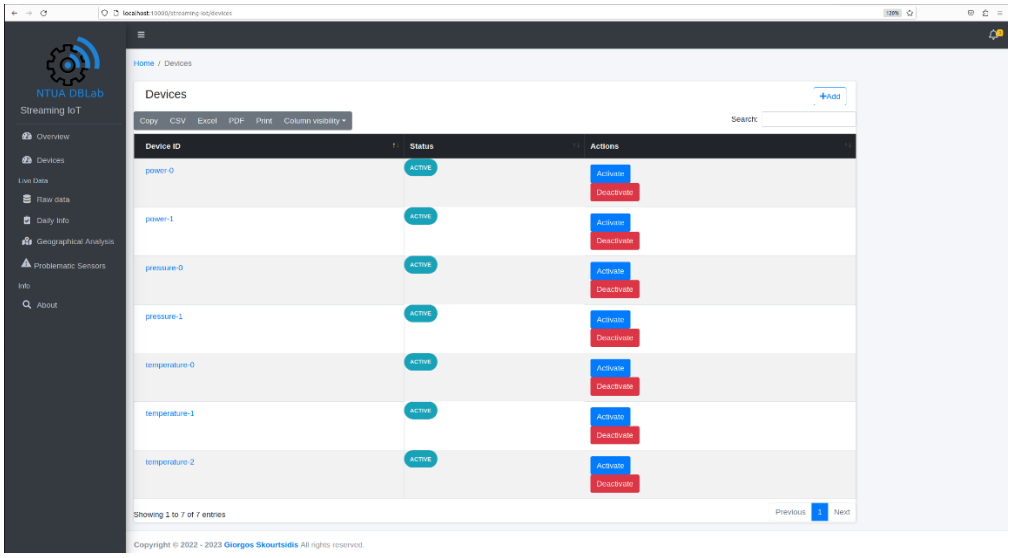


Εικόνα 7-18: Η διεπαφή του Swagger UI

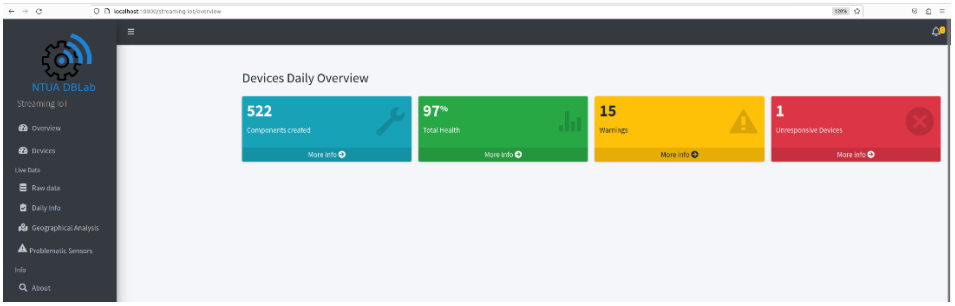
Για να είναι φιλικό προς το χρήστη το τωρινό σύστημα, κρίνεται χρήσιμη η δημιουργία γραφικού περιβάλλοντος. Μέσω αυτής της πλατφόρμας είναι δυνατή μια γρήγορη επίβλεψη του συστήματος. Επιπλέον, είναι δυνατή η υποβολή αιτήματος απενεργοποίησης κάποιας συσκευής, με το πάτημα ενός κουμπιού (εικόνα 7-18).

Το γραφικό περιβάλλον που έχει υλοποιηθεί αποτελεί πρόταση και σίγουρα όχι τελική λύση. Περισσότερα αναλύονται στο κεφάλαιο 8. Αποτελείται από μια γενική σελίδα επίβλεψης (εικόνα 7-19), μια σελίδα όπου παρουσιάζονται όλες οι διαθέσιμες συσκευές σε ένα πίνακα με πληροφορίες (εικόνα 7-18), μία σελίδα όπου λαμβάνονται περισσότερες πληροφορίες για μία συσκευή (εικόνα 7-20) και από μια σελίδα όπου παρουσιάζονται τα πρωτογενή δεδομένα σε γράφημα (εικόνα 7-21).

Το γραφικό περιβάλλον κατασκευάστηκε με χρήση Thymeleaf, JavaScript (jQuery), HTML και CSS. Οι πληροφορίες σχετικά με τις τοποθεσίες των συσκευών και τα ονόματά τους αποθηκεύονται σε μία βάση δεδομένων Postgres. Το γράφημα με τα πρωτογενή δεδομένα που παρουσιάζεται στην εικόνα 7-21 ανανεώνεται σε ζωντανό χρόνο, καθώς γίνεται χρήση της τεχνολογίας WebSockets.

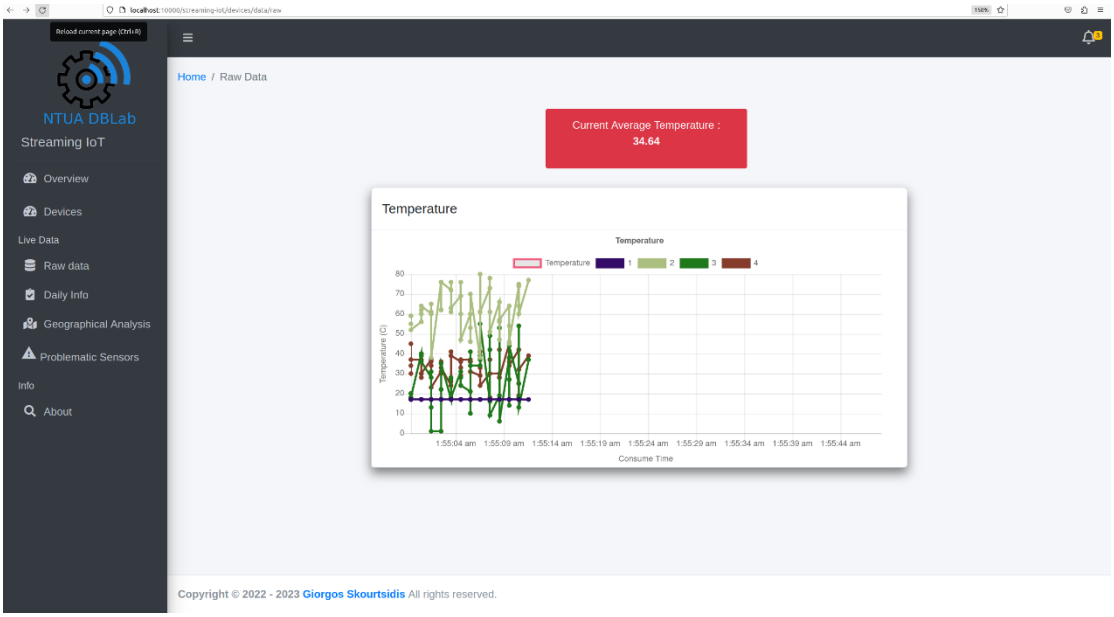


Εικόνα 7-19: Στη σελίδα παρουσίαζονται σε μορφή πίνακα όλες οι διαθέσιμες συσκευές. Η στήλη «Status» δηλώνει εάν αυτή η συσκευή είναι ενεργή και παράγει μετρήσεις τη συγκεκριμένη χρονική στιγμή. Με χρήση των κουμπιών «activate/deactivate» μπορεί μια συσκευή να ενεργοποιηθεί ή να απενεργοποιηθεί αντίστοιχα.Επίσης υπάρχει η δυνατότητα για ordering στις συσκευές με βάση το όνομα, καθώς και δυνατότητα free text search σε όλα τα πεδία (κουτάκι πάνω δεξιά)



Εικόνα 7-20: Μια γενική σελίδα όπου παρουσιάζονται συγκεντρωτικές πληροφορίες για το σύστημα

Εικόνα 7-21: Σε αυτή τη σελίδα παρουσιάζονται περισσότερες πληροφορίες για τον εκάστοτε μετρητή. Η τιμή «Latest measurement» αφορά την τελευταία τιμή που παράχθηκε για έναν μετρητή από το Kafka Streams. Κάθε 5 δευτερόλεπτα γίνεται έλεγχος για το αν παράχθηκε νέα τιμή. Η ανανέωση γίνεται αυτόματα. Η τιμή λαμβάνεται από τη Redis.



Εικόνα 7-22: Γράφημα όπου παρουσιάζονται τα πρωτογενή δεδομένα σε ζωντανό χρόνο με χρήση web sockets. Τα δεδομένα αντλούνται από το Kafka. Το κόκκινο κουτί παρουσιάζει το μέσο όρο θερμοκρασιών από την πιο πρόσφατη τιμή του κάθε αισθητήρα και ανανεώνεται και αυτό σε ζωντανό χρόνο. Ο αισθητήρας θερμοκρασίας 1 έχει ρυθμιστεί ώστε να προσομοιώνει μια συσκευή όπου η θερμοκρασία παραμένει σταθερή και μεγάλη μεταβολή συνιστά πρόβλημα.

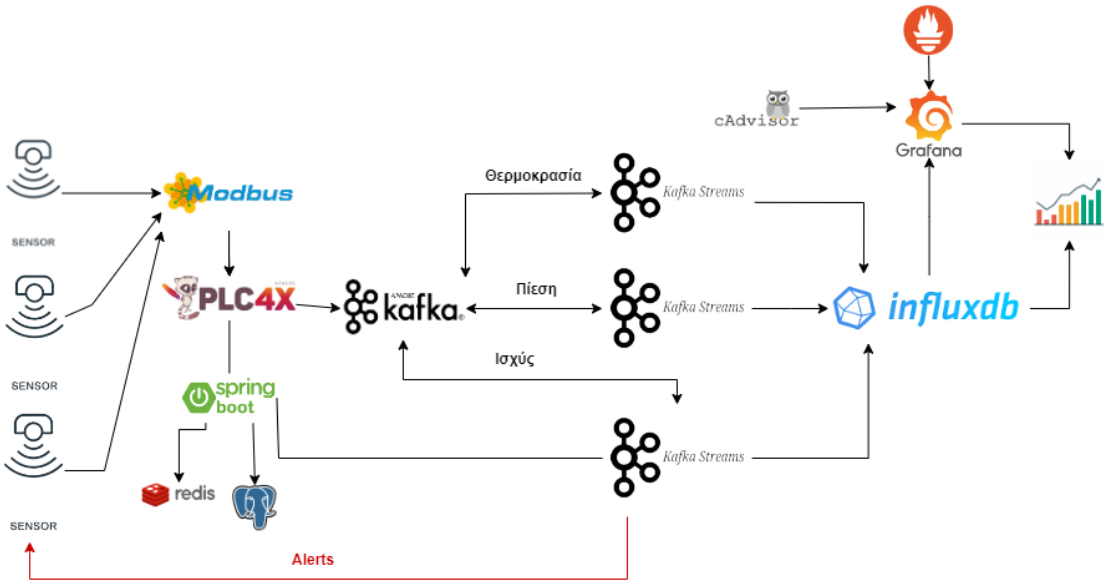
Ο στόχος που θέλει να πετύχει το γραφικό περιβάλλον είναι η ευκολία στο χειρισμό του συστήματος επεξεργασίας, καθώς και η εξαγωγή γνώσης χρήσιμης για την εταιρεία/οργανισμό. Για το λόγο αυτό προτείνονται σελίδες με διαφορετική στόχευση. Αρχικά ο χρήστης συνδέεται στη σελίδα «overview». Στόχος είναι να μπορεί να δει με μια ματιά εάν υπάρχει κάποιο πρόβλημα στο σύστημα παραγωγής των μετρήσεων. Σε εκείνη τη σελίδα βλέπει τα πρόσφατα warnings, καθώς και errors.

Εάν δεν υπάρχουν, τότε όλα είναι καλά και μπορεί να προχωρήσει σε σελίδες που του παρέχουν περισσότερη πληροφορία, όπως η λίστα με τις συσκευές και οι πληροφορίες ανά συσκευή. Για τις άλλες λειτουργικότητες που φαίνονται στο γραφικό περιβάλλον καθώς και για περισσότερη ανάλυση, ο αναγνώστης παραπέμπεται στο κεφάλαιο 8.

Συμπεράσματα

Σε αυτήν την εργασία παρουσιάστηκε ένα ολοκληρωμένο σύστημα παραγωγής, επεξεργασίας, αποθήκευσης και ανάλυσης δεδομένων σε ζωντανό χρόνο. Όλοι οι αρχικοί στόχοι επιτεύχθηκαν.

Το σύστημα είναι επεκτάσιμο, μπορεί να επεξεργαστεί δεδομένα από πολλές και διαφορετικές μεταξύ τους πηγές, είναι ανθεκτικό σε σφάλματα, παρέχει εργαλεία οπτικοποίησης των δεδομένων και εξαγωγής πληροφορίας από αυτά, έχει ενσωματωμένη παρακολούθηση της απόδοσης του και μπορεί να αποστέλλει e-mails εάν κάτι πάει λάθος. Σε περιπτώσεις κινδύνου, λαμβάνει ενέργειες για την αποτροπή ατυχήματος, στέλνοντας μήνυμα στο αρχικό σύστημα προκειμένου να απενεργοποιηθεί αυτόματα κάποια συσκευή. Χρησιμοποιεί αποδοτικές πηγές αποθήκευσης για τα δεδομένα χρονοσειρών (InfluxDB), ενώ τοποθετεί πρόσφατες τιμές που μπορεί να χρησιμοποιηθούν άμεσα σε μια προσωρινή μνήμη (Redis). Οι τεχνολογίες που χρησιμοποιήθηκαν υποστηρίζουν κατανεμημένη επεξεργασία, πράγμα που δίνει τη δυνατότητα κλιμάκωσης σε περίπτωση αύξησης των δεδομένων ή των αισθητήρων. Στο επόμενα κεφάλαιο 8 περιγράφονται επεκτάσεις που μπορούν να ενσωματωθούν στο υπάρχων σύστημα. Στην εικόνα 7-21 απεικονίζονται η συνολική αρχιτεκτονική του συστήματος.



Εικόνα 7-23: Συγκεντρωτικό διάγραμμα από όλα τα μέρη που αποτελούν το σύστημα

8 Μελλοντικές επεκτάσεις

Βελτιώσεις στην εφαρμογή

Τεχνικές βελτιώσεις

Το γραφικό περιβάλλον μπορεί να βελτιωθεί προκειμένου να είναι πιο εύχρηστο, αλλά και να παρέχει στη χρήση ολοκληρωμένη πληροφορία. Θα μπορούσαν οι ειδοποιήσεις να αποθηκεύονται στη βάση Postgres και στη συνέχεια να παρουσιάζονται στο γραφικό περιβάλλον. Με αυτόν τον τρόπο θα μπορεί ο χρήστης σε μία σελίδα να διακρίνει όλα τα προβλήματα. Σε αυτή τη σελίδα χρειάζονται επίσης διάφορα φίλτρα, όπως χρονικό διάστημα, βαρύτητα της ειδοποίησης, αναζήτηση ελεύθερου κειμένου.

Επιπλέον, στο γραφικό περιβάλλον είναι χρήσιμο να μπουν σελίδες με γραφήματα από τους μετρητές. Μπορούν να παρέχονται γραφήματα συγκεντρωτικά, ανά αισθητήρα, ανά κατηγορία μέτρησης και ανά γεωγραφική περιοχή.

Μια άλλη σημαντική λειτουργικότητα είναι η προσθήκη νέων συσκευών στο σύστημα. Αυτό θα μπορούσε να συμβεί, φτιάχνοντας μια σελίδα με μια φόρμα με πληροφορίες (όνομα, περιοχή, τύπος μέτρησης, και οποιαδήποτε άλλη χρήσιμη πληροφορία) οι οποίες θα αποθηκεύονται στην Postgres. Ο κώδικας θα πρέπει να φτιαχτεί με τέτοιο τρόπο ώστε να μπορεί το Kafka Streams να επεξεργάζεται αυτόματα τις τιμές της νέας συσκευής. Ειδικότερα, εάν ο αισθητήρας ανήκει σε νέα κατηγορία μέτρησης, τότε θα πρέπει να δημιουργούνται νέα θέματα Kafka και να εκκινείται νέο στιγμιότυπο του Kafka Streams προκειμένου να εξυπηρετεί αυτούς τους μετρητές.

Βελτιώσεις στην παραγόμενη πληροφορία

Μια σημαντική πληροφορία που θέλει να γνωρίζει κάποιος που διαχειρίζεται αυτήν την πλατφόρμα, είναι η γνώση το που βρίσκεται ένα πρόβλημα. Εάν υπάρχουν πολλαπλές περιοχές με πολλαπλούς αισθητήρες και ανομοιογενή δεδομένα, πολλές φορές η πληροφορία αυτή δεν είναι εύκολα να διασαφηνιστεί. Για το λόγο αυτό στο πλαϊνό μενού προστέθηκε μια καρτέλα «Problematic sensors». Σε αυτή τη σελίδα θα παρουσιάζονται συσκευές που παράγουν τις περισσότερες ειδοποιήσεις, ανά ημέρα. Θα παρουσιάζονται λοιπόν οι «χειρότερες» συσκευές. Είναι σημαντικό να υπάρχουν χρονικά φίλτρα, καθώς και δυνατότητα σύγκρισης ημερών/χρονικής περιόδου, καθώς μια συσκευή μπορεί να παράγει πολλές ειδοποιήσεις για μια χρονική περίοδο, αλλά με την πάροδο των ημερών η κατάσταση να βελτιώνεται και τελικά να είναι σε καλή κατάσταση σήμερα. Χρειάζεται δηλαδή επιτήρηση σε βάθος χρονικών διαστημάτων και όχι απλά παρουσίαση των τελευταίων τιμών.

Αυτή η πληροφορία είναι χρήσιμη για την ταυτοποίηση συσκευών που δημιουργούν πρόβλημα.

Επιπλέον, θα μπορούσαν να υπολογίζονται επιπλέον πληροφορίες μέσα από αυτό το σύστημα. Με την υπόθεση πως βρισκόμαστε σε μια μονάδα παραγωγής, το μπλε κουτί στην εικόνα 7-19 αφορά το σύνολο των παραγόμενων αγαθών που έχουν παραχθεί σήμερα.

Δείκτης Υγείας

Προχωρώντας ένα βήμα παραπέρα, θα μπορούσε θα καθοριστούν δείκτες υγείας για τους μετρητές κάθε κατηγορίας (Key Perfomance Indicator – KPI). Μέσω αυτού του δείκτη θα μπορούσαμε να υπολογίζουμε την υγεία/κατάστασή του για εύρος χρονικών τιμών.

Για κάθε τύπο αισθητήρα, πρέπει να καθοριστεί τι συνιστά "καλή" και "κακή" υγεία. Για έναν αισθητήρα θερμοκρασίας, αυτό μπορεί να είναι η ακρίβεια, η ανταπόκριση και η συνέπεια με την πάροδο του χρόνου. Για έναν αισθητήρα πίεσης, μπορεί να είναι η ευαισθησία, η ακρίβεια και η σταθερότητα. Για έναν αισθητήρα ισχύος, μπορεί να είναι η ακρίβεια και ο χρόνος λειτουργίας.

Προτεινόμενοι δείκτες που θα μπορούσαν να συμμετέχουν στην παραγωγή του συνολικού δείκτη απόδοσης είναι οι εξής:

Ρυθμός μεταβολής: Μια σημαντική αλλαγή στον ρυθμό των μετρήσεων θα μπορούσε να υποδηλώνει πρόβλημα με τον αισθητήρα. Παράδειγμα, οι ενδείξεις του αισθητήρα θερμοκρασίας αρχίζουν ξαφνικά να αυξάνονται ή να μειώνονται πιο γρήγορα από το συνηθισμένο.

Τυπική απόκλιση: Η τυπική απόκλιση μετρά το μέγεθος της διακύμανσης σε ένα σύνολο τιμών. Εάν οι ενδείξεις ενός αισθητήρα αρχίσουν να εμφανίζουν μεγαλύτερη τυπική απόκλιση από ό,τι συνήθως, αυτό θα μπορούσε να υποδεικνύει κάποιο πρόβλημα.

Ανωμαλίες: Εντοπίζει και επισημαίνει τυχόν ανωμαλίες στις ενδείξεις των αισθητήρων. Μια ανωμαλία μπορεί να είναι μια μεμονωμένη ένδειξη που διαφέρει σημαντικά από τις άλλες ή μπορεί να είναι ένα μοτίβο ενδείξεων που διαφέρει από τα συνήθη μοτίβα. Υπάρχουν απλοί και σύνθετοι τρόποι για την ανίχνευση των ανωμαλιών, όμως η ανάλυση τους ξεφεύγει από το τωρινό στόχο.

Ολίσθηση αισθητήρα: Αυτή είναι μια αλλαγή στις ενδείξεις του αισθητήρα με την πάροδο του χρόνου που δεν οφείλεται σε αλλαγές στη μετρούμενη ιδιότητα. Πρόκειται για κοινό πρόβλημα σε πολλούς τύπους αισθητήρων. Η μέτρηση της ολίσθησης μπορεί να είναι περίπλοκη, καθώς απαιτεί την παρατήρηση του αισθητήρα για μεγάλο χρονικό διάστημα και τη διάκριση της ολίσθησης από το πραγματικό σήμα. Μια συνήθης μέθοδος είναι η χρήση ενός μοντέλου γραμμικής παλινδρόμησης για την προσαρμογή μιας γραμμής στις ενδείξεις του αισθητήρα με την πάροδο του χρόνου. Η κλίση αυτής της γραμμής μπορεί να χρησιμοποιηθεί ως μέτρο της ολίσθησης. Πιο εξελιγμένες μέθοδοι περιλαμβάνουν τη χρήση ενός κινητού μέσου όρου για την εξομάλυνση των βραχυπρόθεσμων διακυμάνσεων και για να γίνουν πιο εμφανείς οι μακροπρόθεσμες τάσεις ή τη χρήση αλγορίθμων μηχανικής μάθησης για τη μοντελοποίηση και την πρόβλεψη της συμπεριφοράς του αισθητήρα.

Λόγος θορύβου προς σήμα (SNR): Πρόκειται για ένα μέτρο της ποσότητας θορύβου (τυχαία διακύμανση) στις ενδείξεις του αισθητήρα σε σύγκριση με το πραγματικό σήμα. Ένα υψηλότερο NSR μπορεί να υποδεικνύει πρόβλημα με τον αισθητήρα. Το SNR υπολογίζεται μετρώντας την ισχύ του σήματος (S) και την ισχύ του θορύβου (N). Η ισχύς του σήματος είναι συνήθως το τετράγωνο του μέσου όρου των ενδείξεων του αισθητήρα και η ισχύς του θορύβου είναι η διακύμανση των ενδείξεων του αισθητήρα. Το SNR εκφράζεται σε ντεσιμπέλ (dB): $SNR = 10 * log10(S/N)$.

Απώλεια δεδομένων: Εάν ο αισθητήρας χάνει σημεία δεδομένων στη ροή του (είτε δεν στέλνει δεδομένα στα αναμενόμενα διαστήματα είτε τα δεδομένα χάνονται κατά τη μεταφορά, είτε καταφθάσουν αλλιωμένα), αυτό μπορεί να είναι ένας σημαντικός δείκτης υγείας.

Καθυστέρηση (latency) αισθητήρα: Εάν ο χρόνος που χρειάζεται ο αισθητήρας για να ανταποκριθεί σε αλλαγές και να παρέχει μετρήσεις αυξάνεται, μπορεί να αποτελεί ένδειξη προβλημάτων.

Αριθμός αποτυχιών/σφαλμάτων: Εάν ένας αισθητήρας αποτυγχάνει συχνά ή αναφέρει σφάλματα, αυτό αποτελεί ένδειξη κακής υγείας.

Για κάθε έναν από τους παρακάτω δείκτες, είναι απαραίτητο να καθοριστεί μια βάση για το τιμές λαμβάνει συνήθως η κάθε μετρική. Θα το ονομάσουμε «Βασική» τιμή. Για να καθοριστεί το baseline του κάθε δείκτη, είναι απαραίτητη η παρατήρηση των μετρήσεων του αισθητήρα κάτω από κανονικές συνθήκες λειτουργίας.

Επίσης, είναι απαραίτητη η κανονικοποίηση των τιμών, ώστε το εύρος τιμών να βρίσκεται ανάμεσα από 0 και 1.

Παράδειγμα κανονικοποίησης:

$$\text{Κανονικοποιημένος ρυθμός μεταβολής} = \frac{\text{ρυθμός μεταβολής} \pm \textit{ελάχιστη τιμή ρυθμού μεταβολής}}{\text{μέγιστη} - \text{ελάχιστη τιμή ρυθμού μεταβολής}}$$

Για μετρικές όπως ο Αριθμός αποτυχιών ή η Απώλεια_Δεδομένων, όπου ακόμη και ένα μόνο περιστατικό μπορεί να επηρεάσει σημαντικά την υγεία του αισθητήρα,

δύναται να χρησιμοποιηθεί μια πιο ευαίσθητη προσέγγιση από την κανονικοποίηση. Αυτά τα συμβάντα συχνά υποδεικνύουν ένα κρίσιμο ζήτημα που χρήζει άμεσης προσοχής.

Μια συνήθης προσέγγιση είναι ο καθορισμός ενός κατωφλίου και η εφαρμογή μιας βαριάς ποινής σε περίπτωση υπέρβασης αυτού του κατωφλίου. Για παράδειγμα, θα μπορούσε οποιοδήποτε συμβάν απώλειας δεδομένων ή οποιοδήποτε σφάλμα να μειώνει σημαντικά τη βαθμολογία υγείας. Αυτή θα ήταν μια δυαδική προσέγγιση, όπου η παρουσία ενός μόνο συμβάντος ενεργοποιεί την ποινή.

Μια άλλη προσέγγιση είναι η χρήση μιας μη γραμμικής συνάρτησης για την κλιμάκωση αυτών των μετρήσεων. Για παράδειγμα, μπορεί να χρησιμοποιηθεί μια εκθετική συνάρτηση, όπου ο αντίκτυπος στη βαθμολογία υγείας αυξάνεται ραγδαία καθώς αυξάνεται ο αριθμός των σφαλμάτων. Αυτό θα αντανακλούσε το γεγονός ότι, ενώ ένα μεμονωμένο σφάλμα μπορεί να είναι διαχειρίσιμο, πολλαπλά σφάλματα υποδηλώνουν σοβαρό πρόβλημα.

Με βάση τα παραπάνω, ένας δείκτης για την υγεία των αισθητήρων θα μπορούσε να είναι ο εξής:

$$
\begin{aligned}
Δείκτης\ υγείας = 100 - [&w1 * (Ρυθμός\_Μεταβολής - Βασικός\_Ρυθμός\_Μεταβολής) \\
&+ w2 * (Τυπική\_Απόκλιση - Βασική\_Τυπική\_Απόκλιση) \\
&+ w3 * Ρυθμός\_Ανωμαλιών \\
&+ w4 * (Sensor\_Drift - Βασικό\_Sensor\_Drift) \\
&+ w5 * (NSR - Βασικό\_NSR) \\
&+ w6 * \min(Ρυθμός\_Απώλειας\_Δεδομένων, 1) \\
&\quad * Ποινή\_Απώλειας\_Δεδομένων \\
&+ w7 * (Καθυστέρηση - Βασική\_Καθυστέρηση) \\
&+ w8 * \exp(Αριθμός\_Σφαλμάτων) * Ποινή\_Σφαλμάτων
\end{aligned}
$$

Σε αυτή τη φόρμουλα, τα $w_{1\text{-}8}$ είναι τα βάρη για κάθε δείκτη. Το άθροισμά τους είναι ίσο με 1.

Το πράσινο κουτί στην εικόνα 7-19 (αρχική σελίδα του γραφικού περιβάλλοντος) αναφέρεται στον συνολικό δείκτη υγείας για όλους τους αισθητήρες του συστήματος για την σημερινή ημέρα. Είναι μια γενική εικόνα για το αν το σύστημα αντιμετωπίζει προβλήματα. Επιπλέον, αυτός ο δείκτης θα μπορούσε να χρησιμοποιηθεί για την ανίχνευση των «χειρότερων συσκευών» που περιγράφεται στο αμέσως προηγούμενο κεφάλαιο.

Η παραπάνω φόρμουλα αποτελεί πρόταση. Για τον καθορισμό ενός δείκτη υγείας που είναι ακριβής και αναποκρίνεται στην πραγματικότητα είναι απαραίτητη περισσότερη έρευνα. Για το λόγο αυτό, αφήνεται ως μελλοντική επέκταση.

Πρόβλεψη χρονοσειρών

Η πρόβλεψη δεδομένων αισθητήρων σε πραγματικό χρόνο περιλαμβάνει την πρόβλεψη μελλοντικών ενδείξεων αισθητήρων με βάση τα τρέχοντα και τα προηγούμενα δεδομένα.

Για να εκπαιδευτεί ένα μοντέλο που θα κάνει πρόβλεψη, είναι απαραίτητο να υπάρχει μεγάλος όγκος από παλαιότερα δεδομένα, προκειμένου να μάθει το μοντέλο διάφορα μοτίβα που πιθανόν να υπάρχουν.

Πριν την εκπαίδευση είναι απαραίτητη η προεπεξεργασία των δεδομένων. Αυτό μπορεί να περιλαμβάνει την αντιμετώπιση ελλιπών ή λανθασμένων τιμών, την εξομάλυνση ή το φιλτράρισμα για την απομάκρυνση του θορύβου και ενδεχομένως τον μετασχηματισμό των δεδομένων (π.χ. κλιμάκωση, κανονικοποίηση κ.λπ.).

Στη συνέχεια, πιθανώς θα χρειαστεί εξαγωγή χαρακτηριστικών, ανάλογα με τη μέθοδο πρόβλεψης που χρησιμοποιείται. Αυτά μπορεί να είναι απλά στατιστικά χαρακτηριστικά όπως ο μέσος όρος, η διάμεσος, η τυπική απόκλιση κ.λπ. ή πιο σύνθετα χαρακτηριστικά όπως τάσεις, εποχικότητα ή συνιστώσες συχνότητας.

Ως μοντέλο θα μπορούσε να επιλεχθεί ένα παραδοσιακό στατιστικό μοντέλο όπως το ARIMA (AutoRegressive Integrated Moving Average), μοντέλα χώρου καταστάσεων

(state space models) ή μέθοδοι εκθετικής εξομάλυνσης exponential smoothing). Εναλλακτικά, θα μπορούσε να είναι ένα μοντέλο μηχανικής μάθησης, όπως ένα αναδρομικό νευρωνικό δίκτυο (RNN) ή ένα νευρωνικό δίκτυο LSTM, τα οποία είναι κατάλληλα για πρόβλεψη χρονοσειρών. Στην επιλογή μοντέλου είναι επίσης σημαντικό να ληφθεί υπόψην το υπολογιστικό κόστος του συστήματος πρόβλεψής, καθώς θα πρέπει να εκτελείται σε πραγματικό χρόνο.

Αφού εκπαιδευτεί το μοντέλο, αποθηκεύεται (συνήθως σε μορφή pickle). Για την πρόβλεψη των δεδομένων, το μοντέλο φορτώνεται από ένα αρχείο και στη συνέχεια παράγει τις προβλέψεις, αφού φορτωθεί με τα δεδομένα. Οι προβλέψεις θα αποθηκεύονται στην InfluxDB και θα παρουσιάζονται στο γραφικό περιβάλλον.

Είναι απαραίτητο να οριστεί κάθε πότε θα τρέχει η πρόβλεψη. Θα μπορούσε να τρέχει:

1. Σε σταθερό διάστημα Πχ κάθε λεπτό, κάθε ώρα ή κάθε μέρα.
2. Κάθε φορά που ολοκληρώνεται ένα χρονικό παράθυρο από το Kafka Streams

Οι παραπάνω 2 επιλογές αποτελούν ένα απλό παράδειγμα. Ο χρόνος που θα παράγονται οι προβλέψεις απαιτεί περισσότερη έρευνα και πειραματισμό και πρέπει να γίνει σε συνδυασμό με την συνολικότερη αρχιτεκτονική του συστήματος πρόβλεψης.

Καθώς έρχονται νέα δεδομένα, το μοντέλο πρέπει να ενημερώνεται. Αυτό μπορεί να περιλαμβάνει περιοδική επανεκπαίδευση του μοντέλου ή τη χρήση μιας μεθόδου online μάθησης.

Τέλος, είναι απαραίτητη η αξιολόγηση της απόδοση του μοντέλου συγκρίνοντας τις προβλέψεις του με τις πραγματικές ενδείξεις των αισθητήρων.

Επιπρόσθετα στα παραπάνω, θα μπορούσε να γίνεται πρόβλεψη την υγεία των αισθητήρων (περιγράφεται στο αμέσως προηγούμενο κεφάλαιο), με την προϋπόθεση πως αυτός ο δείκτης έχει υλοποιηθεί.

9 Βιβλιογραφία

[1] M. A. A. da Cruz, J. J. P. C. Rodrigues, J. Al-Muhtadi, V. v. Korotaev, and V. H. C. de Albuquerque, “A Reference Model for Internet of Things Middleware,” IEEE Internet Things J, vol. 5, no. 2, pp. 871–883, Apr. 2018, doi: 10.1109/JIOT.2018.2796561.

[2] A. Karmakar, N. Dey, T. Baral, M. Chowdhury, and M. Rehan, “Industrial internet of things: A review,” 2019 International Conference on Opto-Electronics and Applied Optics, Optronix 2019, Mar. 2019, doi: 10.1109/OPTRONIX.2019.8862436.

[3] Y. Zhang, J. Ren, J. Liu, C. Xu, H. Guo, and Y. Liu, “A survey on emerging computing paradigms for big data,” Chinese Journal of Electronics, vol. 26, no. 1, pp. 1–12, Jan. 2017, doi: 10.1049/CJE.2016.11.016.

[4] L. Farhan, R. Kharel, O. Kaiwartya, M. Quiroz-Castellanos, A. Alissa, and M. Abdulsalam, “A Concise Review on Internet of Things (IoT)-Problems, Challenges and Opportunities,” 2018 11th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2018, Sep. 2018, doi: 10.1109/CSNDSP.2018.8471762.

[5] J. Kreps, I [love] logs : [event data, stream processing, and data integration]. 2014. Accessed: Mar. 19, 2023. [Online]. Available: https://www.oreilly.com/library/view/i-heart-logs/9781491909379/

[6] Hugh Jack, “Automating Manufacturing Systems with PLCs,” http://rdjoudjou.freehostia.com/livre/plcbook5_1.pdf, 2008. https://archive.org/details/ost-engineering-plcbook5_1/page/n15/mode/2up (accessed Feb. 15, 2023).

[7] “Modbus Protocol Overview.” https://www.fernhillsoftware.com/help/drivers/modbus/modbus-protocol.html (accessed Mar. 19, 2023).

[8] “What is Modbus? | Types of Modbus | RealPars.” https://realpars.com/modbus/ (accessed Mar. 25, 2023).

[9] “Using MODBUS for Process Control and Automation.” https://www.miinet.com/images/pdf/articles/Using_MODBUS_for_Process_Control_and_Automation.pdf (accessed Mar. 20, 2023).

[10] “MODBUS Interface Module User’s Manual-QJ71MB91-GX Configurator-MB (SW1D5C-QMBU-E).” https://dl.mitsubishielectric.com/dl/fa/document/manual/plc/sh080578eng/sh080578engj.pdf (accessed Mar. 20, 2023).

[11] “What is Modbus and How does it work? | Schneider Electric USA.” https://www.se.com/us/en/faqs/FA168406/ (accessed Mar. 19, 2023).

[12] Modbusorg, “MODBUS Application Protocol 1 1 b,” 2006. https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf (accessed Mar. 19, 2023).

[13] “Modbus Table Format and Data Types - PowerTag Link User Guide.” https://www.productinfo.schneider-electric.com/powertaglinkuserguide/powertag-link-user-guide/English/BM_PowerTag%20Link%20D%20User%20Manual_4af62430_T000501355.xml/$/TPC_ModbusTableFormatandDataTypes_4af62430_T000501590 (accessed Mar. 19, 2023).

[14] “Modbus manual TD80.” https://camatsystem.com/wp-content/uploads/2015/12/Modbus-manual-TD80.pdf (accessed Mar. 20, 2023).

[15] “Modbus Exception Codes.” https://product-help.schneider-electric.com/ED/ES_Power/NT-NW_Modbus_IEC_Guide/EDMS/DOCA0054EN/DOCA0054xx/Master_NS_Modbus_Protocol/Master_NS_Modbus_Protocol-5.htm (accessed Mar. 19, 2023).

[16] “Different Types of Modbus - Chipkin Automation Systems.” https://store.chipkin.com/articles/different-types-of-modbus-such-as-rtu-tcp-etc (accessed Mar. 19, 2023).

[17] “PROFINET Explained – PI North America.” https://us.profinet.com/profinet-explained/ (accessed Mar. 19, 2023).

[18] “OPC Unified Architecture - Wikipedia.” https://en.wikipedia.org/wiki/OPC_Unified_Architecture (accessed Apr. 24, 2023).

[19] Inray Industriesoftware, “A practical introduction to OPC UA.” https://www.opc-router.com/what-is-opc-ua/ (accessed Apr. 24, 2023).

[20] “INDUSTRIE 4.0 Smart Manufacturing for the Future”. https://www.pac.gr/bcm/uploads/industrie4-0-smart-manufacturing-for-the-future-en.pdf (accessed Apr. 24, 2023).

[21] “PLC4X –.” https://plc4x.apache.org/ (accessed Mar. 26, 2023).

[22] “Apache Kafka.” https://kafka.apache.org/documentation/ (accessed Mar. 19, 2023).

[23] “Apache Kafka: Use Cases.” https://kafka.apache.org/uses (accessed Mar. 27, 2023).

[24] M. Kleppmann, Making Sense of Stream Processing: the philosophy behind Apache Kafka and Scalable Stream Data Platforms. 2016. Accessed: Mar. 19, 2023. [Online]. Available: https://www.oreilly.com/library/view/making-sense-of/9781492042563/

[25] William P. Bejeck Jr., Kafka Streams in Action. 2018. Accessed: Mar. 19, 2023. [Online]. Available: https://www.manning.com/books/kafka-streams-in-action

[26] N. Narkhede, G. Shapira, and T. Palino, “Kafka: The Definitive Guide.”

[27] M. Seymour, Mastering Kafka Streams and ksqlDB : building real-time data systems by example. 2021.

[28] G. Wang et al., “Consistency and Completeness: Rethinking Distributed Stream Processing in Apache Kafka,” Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 2602–2613, 2021, doi: 10.1145/3448016.3457556.

[29] “Timely Stream Processing | Apache Flink.” https://nightlies.apache.org/flink/flink-docs-release-1.16/docs/concepts/time/ (accessed Mar. 20, 2023).

[30] T. Akidau et al., “Watermarks in Stream Processing Systems: Semantics and Comparative Analysis of Apache Flink and Google Cloud Dataflow,” vol. 14, no. 12, pp. 2150–8097, 2021, doi: 10.14778/3476311.3476389.

[31] T. Akidau et al., “The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing,” 2150.

[32] “Using watermark in Flink | CDP Private Cloud.” https://docs.cloudera.com/csa/1.2.0/flink-overview/topics/csa-watermark.html (accessed Mar. 25, 2023).

[33] “Apache Kafka Streams: Architecture.” https://kafka.apache.org/34/documentation/streams/architecture (accessed Mar. 20, 2023).

[34] M. Roberts, Designing even-driven systems : concepts and patterns for streaming services with Apache Kafka. 2018. Accessed: Mar. 19, 2023. [Online]. Available: https://www.oreilly.com/library/view/designing-event-driven-systems/9781492038252/

[35] “Industrial IoT Analytics | InfluxData.” https://www.influxdata.com/solutions/industrial-iot/ (accessed Mar. 26, 2023).

[36] E. Robinson, “Index and Time-Structured Merge Tree (TSM) Overview in InfluxDB 1.5,” 2018.

[37] A. Georgiou, “Intro to InfluxDB IOx Storage Engine,” 2023.

[38] “Edge Computing and Data Replication with InfluxDB,” 2022. https://get.influxdata.com/rs/972-GDU-533/images/Edge-Computing-and-Data-Replication-with-InfluxDB.pdf (accessed Mar. 26, 2023).

[39] “InfluxDB design principles | InfluxDB OSS 2.6 Documentation.” https://docs.influxdata.com/influxdb/v2.6/reference/key-concepts/design-principles/#schemaless-design (accessed Mar. 26, 2023).

[40] “Tools and Integrations with InfluxDB,” 2022. https://get.influxdata.com/rs/972-GDU-533/images/05-17-2021-Integations-Paper.pdf (accessed Mar. 26, 2023).

[41] “Industrial IoT with InfluxDB,” 2021. https://get.influxdata.com/rs/972-GDU-533/images/TechPaper_IIoT_Solution.pdf (accessed Mar. 26, 2023).

[42] “Query data stored in InfluxDB | InfluxDB OSS 2.6 Documentation.” https://docs.influxdata.com/influxdb/v2.6/query-data/ (accessed Mar. 26, 2023).

[43] “InfluxQL Continuous Queries | InfluxDB OSS 1.8 Documentation.” https://docs.influxdata.com/influxdb/v1.8/query_language/continuous_queries/#schedule-and-coverage (accessed Mar. 26, 2023).

[44] “Monitor data and send alerts | InfluxDB OSS 2.6 Documentation.” https://docs.influxdata.com/influxdb/v2.6/monitor-alert/ (accessed Mar. 26, 2023).

[45] P. Dix, “Time Series for Metrics, Real-Time Analytics and Sensor Data,” 2021.

[46] “Network Monitoring with InfluxData,” 2021. https://get.influxdata.com/rs/972-GDU-533/images/TechPaper_NW_Monitoring_Solution_02192019.pdf (accessed Mar. 26, 2023).

[47] “PLC4X –.” https://plc4x.apache.org/plc4j/ (accessed May 3, 2023).

[48] B. B. Rad, H. J. Bhatti, and M. Ahmadi, “An Introduction to Docker and Analysis of its Performance,” IJCSNS International Journal of Computer Science and Network Security, vol. 17, no. 3, 2017.