ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων
Πληροφορικής

ΥΛΟΠΟΙΗΣΗ ΠΛΑΤΦΟΡΜΑΣ ENERGY DISAGGREGATION ΓΙΑ ΤΗΝ ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΤΗΣ ΕΝΕΡΓΕΙΑΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΣΤΑΣΙΟΣ ΚΑΤΣΟΥΛΑΣ

Επιβλέπων : Ευάγγελος Μαρινάκης
Επίκουρος Καθηγητής ΕΜΠ

Αθήνα, Οκτώβριος, 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων
Πληροφορικής

ΥΛΟΠΟΙΗΣΗ ΠΛΑΤΦΟΡΜΑΣ ENERGY DISAGGREGATION ΓΙΑ ΤΗΝ ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΤΗΣ ΕΝΕΡΓΕΙΑΚΗΣ ΚΑΤΑΝΑΛΩΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναστάσιος Κατσούλας

Επιβλέπων : Ευάγγελος Μαρινάκης
Επίκουρος Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 17 Οκτωβρίου 2025

.....	.....	.....
Ευάγγελος Μαρινάκης	Δημήτριος Ασκούνης	Ευάγγελος Σπηλιώτης,
Επίκουρος Καθηγητής, ΕΜΠ	Καθηγητής, ΕΜΠ	Επίκουρος Καθηγητής, ΕΜΠ

Αθήνα, Οκτώβριος, 2025

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

Περίληψη

Η ενεργειακή κατανάλωση και η αποδοτική διαχείρισή της αποτελούν κεντρικά ζητήματα για τις σύγχρονες κοινωνίες, καθώς η ανάγκη για ενεργειακή αποδοτικότητα και βιωσιμότητα γίνεται ολοένα και πιο επιτακτική. Στον πυρήνα αυτής της ανάγκης βρίσκεται η δυνατότητα παρακολούθησης και κατανόησης της κατανάλωσης ενέργειας σε ατομικό επίπεδο, δηλαδή σε επίπεδο συσκευών και νοικοκυριών. Η Ενεργειακή Κατανομή (**Energy Disaggregation**) αναδεικνύεται ως μια κρίσιμη τεχνολογία, η οποία επιτρέπει τον προσδιορισμό της κατανάλωσης ενέργειας που αποδίδεται σε κάθε συσκευή ξεχωριστά, χωρίς να απαιτούνται μετρητές ανά συσκευή. Με αυτόν τον τρόπο, οι καταναλωτές μπορούν να ενημερώνονται καλύτερα για τις ενεργειακές τους συνήθειες και να λαμβάνουν στοχευμένες αποφάσεις για τη μείωση της κατανάλωσης.

Παρά τα σημαντικά οφέλη που προσφέρει ο διαχωρισμός της κατανάλωσης ενέργειας, η υλοποίησή του σε πραγματικό χρόνο και σε μεγάλη κλίμακα αποτελεί τεράστια πρόκληση. Η διαδικασία απαιτεί τη συνεχή συλλογή και ανάλυση τεράστιων όγκων δεδομένων, που προέρχονται από ένα ευρύ πλήθος χρηστών και συσκευών, ενώ η ακρίβεια του αποτελέσματος επηρεάζεται από την ποικιλία των προφίλ κατανάλωσης και τα διαφορετικά μοτίβα χρήσης των ηλεκτρικών συσκευών. Η αντιμετώπιση αυτής της πρόκλησης απαιτεί την υιοθέτηση νέων τεχνολογιών και την ανάπτυξη ισχυρών υποδομών, οι οποίες μπορούν να υποστηρίξουν τη ροή, την αποθήκευση και την επεξεργασία αυτών των δεδομένων σε πραγματικό χρόνο.

Η παρούσα διπλωματική εργασία εστιάζει στη μελέτη και την ανάπτυξη τέτοιων προηγμένων υποδομών και τεχνολογικών λύσεων, με στόχο να υποστηρίξει αποτελεσματικά την Ενεργειακή Κατανομή σε ευρεία κλίμακα. Στο πλαίσιο αυτό, διερευνώνται οι βέλτιστες πρακτικές για τη συλλογή και αποθήκευση των δεδομένων ενώ παρουσιάζονται καινοτόμες μέθοδοι και εργαλεία επεξεργασίας ροών δεδομένων για την ανάλυσή τους σε πραγματικό χρόνο.

Με τον τρόπο αυτό, η εργασία στοχεύει να συμβάλει ουσιαστικά στη δημιουργία ενός βιώσιμου και αποδοτικού ενεργειακού οικοσυστήματος, προσφέροντας τόσο θεωρητική όσο και εφαρμοσμένη γνώση για την επόμενη γενιά έξυπνων ενεργειακών λύσεων.

Λέξεις-κλειδιά: Μηχανική Μάθηση, Ροές δεδομένων, IoT, Flink, Kafka, Kubernetes

Abstract

Energy consumption and its efficient management constitute central issues for modern societies, as the need for energy efficiency and sustainability is becoming increasingly urgent. At the core of this need lies the ability to monitor and understand energy consumption at an individual level namely, at the level of devices and households. Energy Disaggregation emerges as a critical technology, enabling the determination of energy consumption attributed to each individual device without requiring separate meters for each device. In this way, consumers can be better informed about their energy habits and make targeted decisions to reduce their consumption.

Despite the significant benefits offered by disaggregating energy consumption, implementing it in real time and on a large scale remains a formidable challenge. The process requires the continuous collection and analysis of vast amounts of data from a wide variety of users and devices, while the accuracy of the results is affected by the diversity of consumption profiles and different usage patterns of electrical appliances. Addressing this challenge necessitates the adoption of new technologies and the development of robust infrastructures capable of supporting the flow, storage, and real-time processing of these data.

This thesis focuses on the study and development of such advanced infrastructures and technological solutions, aimed at effectively supporting Energy Disaggregation on a large scale. In this context, best practices for data collection, streaming, and storage are investigated, and innovative methods and tools for real-time data analysis are presented.

By doing so, this work aspires to contribute substantially to the creation of a sustainable and efficient energy ecosystem, offering both theoretical and applied knowledge for the next generation of “smart” energy solutions.

Key-words: Machine learning, Streaming Data Pipeline, IoT, Flink, Kafka, Kubernetes

Ευχαριστίες

Θα ήθελα, αρχικά, να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Ευάγγελο Μαρινάκη, τόσο για την αρχική του ιδέα και την ευκαιρία που μου έδωσε να ασχοληθώ με το συγκεκριμένο θέμα, όσο και για την καθοδήγησή του και την εξαιρετική συνεργασία καθόλη τη διάρκεια εκπόνησης της διπλωματικής εργασίας. Επίσης, οφείλω να ευχαριστήσω τους υποψήφιους διδάκτορες Ιωάννη Παπία, Φίλιππα Σερέπα και Νίκο Δημητρόπουλο, για την αμέριστη βοήθειά τους, την έμπνευση που μου προσέφεραν και τη συνεργασία μας, που έκανε αυτή την πορεία πιο ευχάριστη και δημιουργική.

Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στην οικογένειά μου και σε όλους όσους με στήριξαν και ήταν δίπλα μου σε αυτή μου την πορεία.

Αναστάσιος Κατσούλας

Οκτώβριος, 2025

Πίνακας περιεχομένων

Κατάλογος Σχημάτων

Κατάλογος Ακρωνυμίων

IoT	Internet of Things
ML	Machine learning
DL	Deep learning
NILM	Non-Intrusive Load Monitoring
FFT	Fast Fourier Transform
HHM	Hidden Markov Model
DBSCAN	Density-Bases Spatial Clustering of Apllications with Noise
MLP	MultiLayer Perceptron
SVM	Support Vector Machines
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
HA	High Availability
K8s	Kubernetes
API	Application Programming Interface
REST	REpresentational State Transfer
DNS	Domain Name System
IP	Internel Protocol
PV	Persistent Volume
PVC	Persistent Volume Claim
SSL	Secure Sockets Layer
CRD	Custom Resource Definition
MQTT	Message Queing Telemetry Transport
MQTT-SN	Message Queuing Telemetry Transport for Sensor Networks
TLS	Transport Layer Security
WS	Web Socket
WSS	Web Socket Secure
CoAP	Constrained Application Protocol
LwM2M	Lightweight Machine to Machine
SQL	Structured Query Language
LDAP	Lightweight Directory Access Protocol
STOMP	Simple (or Streaming) Text Oriented Messaging Protocol
JWT	JSON Web Token
ACL	Access Control List
QoS	Quality of Service
AWS	Amazon Web Services
HPA	Horizontal Pod Autoscaler
RBAC	Role Bases Access Control
DAG	Directed Acyclic Graph
RTC	Real Time Clock
GUI	Graphical User Interface

Εισαγωγή

Κίνητρο & Στόχος

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η σχεδίαση και υλοποίηση μιας υποδομής που θα μπορεί να ανταπεξέλθει στις απαιτήσεις της κατανομής της κατανάλωσης ενέργειας σε πραγματικό χρόνο, διαχειριζόμενη τεράστιους όγκους δεδομένων που προέρχονται από ένα μεγάλο δίκτυο συσκευών και χρηστών. Η υποδομή αυτή θα πρέπει να είναι σε θέση να συλλέγει δεδομένα από συσκευές IoT, να τα διανέμει μέσω message Brokers και να τα αναλύει σε πραγματικό χρόνο χρησιμοποιώντας τεχνολογίες όπως Kafka και Flink.

Συγκεκριμένα, η εργασία θα εστιάσει στα εξής:

1. **Συλλογή Δεδομένων**: Η ανάπτυξη και διασύνδεση μέσω προσομοίωσης ενός δικτύου συσκευών IoT, που θα μπορούν να καταγράφουν την ενεργειακή κατανάλωση σε επίπεδο συσκευής και να μεταδίδουν αυτά τα δεδομένα σε κεντρικά συστήματα.
2. **Ανάλυση Δεδομένων σε Πραγματικό Χρόνο**: Η ανάπτυξη μιας πλατφόρμας ανάλυσης δεδομένων που θα χρησιμοποιεί τεχνολογίες ροών δεδομένων για το μετασχηματισμό ώστε να δημιουργηθούν ουσιώδη και πλήρη σύνολα δεδομένων για τα μοντέλα μηχανικής μάθησης σε πραγματικό χρόνο.
3. **Αξιολόγηση Απόδοσης & Ορθότητας**: Η αξιολόγηση της αποδοτικότητας, της κλιμακωσιμότητας καθώς και της ακρίβειας της προτεινόμενης υποδομής, μέσω δοκιμών με τεχνητά δεδομένα. Η εργασία θα εξετάσει κατά πόσο η προτεινόμενη υποδομή μπορεί να διαχειριστεί μεγάλους όγκους δεδομένων και να αποδώσει ακριβή σύνολα δεδομένων που θα χρησιμοποιηθούν στην κατανομή των φορτίων των επιμέρους συσκευών σε πραγματικό χρόνο.

Με αυτόν τον τρόπο, η διπλωματική εργασία θα προσφέρει μια ολοκληρωμένη λύση για την υλοποίηση του διαχωρισμού κατανάλωσης ενέργειας σε πραγματικό χρόνο, συμβάλλοντας σημαντικά στην επίτευξη των στόχων της ενεργειακής αποδοτικότητας και της βιωσιμότητας.

Μεθοδολογία

Στην εργασία αυτή ακολουθήθηκε μεθοδολογία design-science με έμφαση στη σχεδίαση, υλοποίηση και εμπειρική αξιολόγηση μιας end-to-end ροής ενεργειακής κατανομή. Η υποδομή αναπτύχθηκε σε Kubernetes (MicroK8s) με EMQX (MQTT) για λήψη μετρήσεων, Apache Kafka για αποθήκευση και Apache Flink για επεξεργασία ροών σε πραγματικό χρόνο. Μοντελοποιήσαμε δεδομένα ολικής ισχύος και καταστάσεων συσκευών και δημιουργήσαμε συνθετικά σύνολα για ελεγχόμενες δοκιμές.

Διάρθρωση της εργασίας

Στο Κεφάλαιο 2 παρουσιάζεται το γνωστικό πλαίσιο της ενεργειακής κατανομής, ορίζεται το πρόβλημα, αποσαφηνίζονται οι παρεμβατικές και μη παρεμβατικές προσεγγίσεις και συνοψίζονται οι βασικές τεχνικές μηχανικής μάθησης που χρησιμοποιούνται για την αποσύνθεση της συνολικής κατανάλωσης σε επιμέρους φορτία. Στη συνέχεια μεταφέρεται το επίκεντρο στις υποδομές μεγάλης κλίμακα, πώς τα cluster μηχανημάτων επιτρέπουν ανθεκτικότητα και κλιμακωσιμότητα με έμφαση στο Kubernetes. Αναλύουμε δομές και πρωτόκολλα για την αποδοτική συλλογή δεδομένων από IOT συσκευές, την ανθεκτική αποθήκευση και την δρομολόγηση τους για επεξεργασία σε πραγματικό χρόνο.

Στο κεφάλαιο 3 αναλύεται η αρχιτεκτονική της προτεινόμενης λύσης, τα επιμέρους δομικά στοιχεία, οι επιλογές παραμετροποίησης και η μοντελοποίηση των δεδομένων χρηστών και συσκευών. Τεκμηριώνεται ο μηχανισμός παραγωγής συνθετικών δεδομένων και οι τεχνικές που ακολουθήσαμε. Έπειτα αναλύεται η διαδρομή των δεδομένων από τη διασύνδεση των χρηστών και των συσκευών με το EMQX, τους κανόνες επεξεργασίας στον Broker και την αποθήκευση τους στο Kafka. Τέλος, αναλύουμε την επεξεργασία που πραγματοποιούμε στα δεδομένα μας μέσω του Flink και η δημιουργία ροών για πρόβλεψη κατανάλωσης (συνεχή εξαγωγή αποτελεσμάτων) και την εκπαίδευση μοντέλων (συγκέντρωση/εμπλουτισμός δεδομένων σε παρτίδες) μηχανικής μάθησης, με αναφορά στις αποφάσεις σχεδίασης που εξασφαλίζουν αξιοπιστία, επεκτασιμότητα και λειτουργικότητα της διαδικασίας από άκρο σε άκρο.

Στα δύο τελευταία κεφάλαια συνοψίζονται τα αποτελέσματα και τα πορίσματα από την ανάπτυξη και αξιολόγηση της υποδομής, η δυνατότητα συλλογής και επεξεργασίας ενεργειακών δεδομένων σε κλίμακα. Η ομαλή συνεργασία EMQX–Kafka–Flink πάνω σε Kubernetes και οι προϋποθέσεις για σταθερή λειτουργία σε πραγματικό χρόνο. Επισημαίνονται τα όρια και οι προκλήσεις και προτείνονται κατευθύνσεις για επέκταση και βελτίωση ώστε να χρησιμοποιηθεί σε περιβάλλοντα με πραγματικά δεδομένα.

.

1. Θεωρητικό Υπόβαθρο

Ενεργειακή Κατανομή (Energy Disaggregation)

Εισαγωγή

Η ενεργειακή κατανομή, ευρέως γνωστή και ως Non-Intrusive Load Monitoring (NILM), αποτελεί μια καινοτόμο τεχνική που στοχεύει στην αναλυτική καταγραφή και απεικόνιση της συνολικής κατανάλωσης ηλεκτρικής ενέργειας ενός νοικοκυριού ή μιας εγκατάστασης, διαχωρίζοντάς την σε μεμονωμένες συνιστώσες. Μέσω αυτής της μεθόδου, επιχειρείται η ακριβής αντιστοίχιση της κατανάλωσης σε συγκεκριμένες συσκευές ή φορτία, χωρίς να απαιτείται η εγκατάσταση ξεχωριστών μετρητών για καθεμία από αυτές. Η διαδικασία στηρίζεται στην ανάλυση του συνολικού ηλεκτρικού σήματος, ένα μοναδικό "αποτύπωμα", που αφήνει η λειτουργία κάθε συσκευής, επιτρέποντας έτσι την παρακολούθηση και κατανόηση της ενεργειακής χρήσης με έναν μη επεμβατικό, οικονομικό και ταυτόχρονα αξιόπιστο τρόπο.

Η ανάγκη για τέτοιου είδους λεπτομερή χαρτογράφηση της κατανάλωσης προέκυψε κυρίως λόγω της συνεχώς αυξανόμενης ζήτησης ενέργειας, σε συνδυασμό με την επιδίωξη μείωσης του περιβαλλοντικού αντίκτυπου και των λειτουργικών εξόδων. Μέσα από την ενεργειακή κατανομή, οι χρήστες, τόσο οικιακοί καταναλωτές όσο και μεγάλες βιομηχανικές εγκαταστάσεις μπορούν να εντοπίσουν συσκευές με υψηλή κατανάλωση, να βελτιστοποιήσουν τη χρήση τους, και τελικά να συμβάλουν πιο ενεργά στη βιώσιμη διαχείριση της ενέργειας. Επιπλέον, το NILM αποτελεί κρίσιμο εργαλείο για τον σχεδιασμό πιο έξυπνων ηλεκτρικών δικτύων (Smart Grids), τα οποία απαιτούν ακριβείς και πραγματικού χρόνου πληροφορίες για τον έλεγχο της ζήτησης και της προσφοράς ενέργειας.

Για να επιτευχθεί ο παραπάνω στόχος, έχουν προταθεί ποικίλες μέθοδοι και αλγόριθμοι, αξιοποιώντας εξελιγμένες τεχνικές επεξεργασίας σήματος, μηχανικής και βαθιάς μηχανικής μάθησης. Αυτές οι μέθοδοι περιλαμβάνουν την ανάλυση χαρακτηριστικών μοτίβων (patterns) του ηλεκτρικού σήματος, την αναγνώριση μεταβατικών φαινομένων κατά την ενεργοποίηση και απενεργοποίηση μιας συσκευής, καθώς και την εφαρμογή αλγορίθμων ταξινόμησης & συσταδοποίησης, με σκοπό τον διαχωρισμό των φορτίων. Η ερευνητική δραστηριότητα γύρω από την ενεργειακή κατανομή είναι έντονη και συνεχώς εξελίσσεται, ακολουθώντας τις απαιτήσεις της βιομηχανίας και των καταναλωτών για ακόμη μεγαλύτερη ακρίβεια, ταχύτητα και ευελιξία στη διαδικασία αναγνώρισης των φορτίων.[1]

Η ενεργειακή κατανομή, πέραν της θεωρητικής της σημασίας, βρίσκει ευρεία εφαρμογή σε διάφορους τομείς της καθημερινότητας και της βιομηχανίας. Από την οικιακή χρήση, όπου ο καταναλωτής έχει τη δυνατότητα να διαχειρίζεται καλύτερα τον λογαριασμό του και να βελτιώνει τις συνήθειες κατανάλωσής του, μέχρι τις εμπορικές και βιομηχανικές εγκαταστάσεις, όπου η ενεργειακή κατανομή συμβάλλει στην αποδοτικότερη λειτουργία των συστημάτων, στη διαχείριση της ζήτησης, καθώς και στη συντήρηση και στην πρόβλεψη βλαβών. Στα επόμενα υποκεφάλαια, θα αναλυθούν εκτενέστερα οι διάφορες εφαρμογές, τα οφέλη που απορρέουν από αυτές, καθώς και οι προκλήσεις που αντιμετωπίζει η επιστημονική κοινότητα στον χώρο του NILM.

Intrusive vs. Non-Intrusive Approaches

Ο επιμερισμός της κατανάλωσης σε συσκευές αποτελεί κομβικό σημείο για την αποτελεσματική διαχείριση της ηλεκτρικής ενέργειας, καθώς παρέχει αναλυτική πληροφόρηση για τις ενεργοβόρες συσκευές εντός ενός οικιακού ή βιομηχανικού περιβάλλοντος. Στην πράξη, έχουν διαμορφωθεί δύο κύριες προσεγγίσεις για την καταγραφή και τον εντοπισμό της κατανάλωσης κάθε συσκευής: η *intrusive (παρεμβατική)* και η *non-intrusive (μη παρεμβατική)*.

Intrusive Approaches

Οι παρεμβατικές μέθοδοι απαιτούν την τοποθέτηση επιπρόσθετων αισθητήρων σε κάθε συσκευή ή σε μικρές ομάδες συσκευών, προκειμένου να μετρηθεί με ακρίβεια η κατανάλωση ρεύματος. Αν και παρέχουν υψηλή ακρίβεια και λεπτομερείς μετρήσεις, η εγκατάστασή τους είναι συνήθως δαπανηρή και χρονοβόρα, ειδικά σε μεγάλα κτίρια ή βιομηχανικά συγκροτήματα. Επιπλέον, η ανάγκη για εκτεταμένη καλωδίωση ή ασύρματους αισθητήρες προκαλεί επιπλέον πολυπλοκότητα ως προς τη συντήρηση και την ασφάλεια των δεδομένων.

Non-Intrusive Load Monitoring (NILM)

Σε αντιδιαστολή, η μη παρεμβατική προσέγγιση βασίζεται στην ανάλυση της συνολικής κατανάλωσης ρεύματος ή και άλλων μετρήσεων, όπως αυτή καταγράφεται από έναν ή λίγους κεντρικούς μετρητές (π.χ. στον κεντρικό ηλεκτρικό πίνακα). Χάρη σε εξελιγμένους αλγορίθμους επεξεργασίας σήματος και τεχνικές μηχανικής μάθησης, η συνολική κατανάλωση μπορεί να «διαχωριστεί» (disaggregated) σε επιμέρους συνιστώσες, κατανέμοντας την ενέργεια ανά συσκευή ή ομάδα συσκευών. Έτσι, δεν απαιτείται ξεχωριστός αισθητήρας σε κάθε συσκευή, γεγονός που μειώνει σημαντικά το κόστος εγκατάστασης και συντήρησης.

Ωστόσο, η επιτυχία της προσέγγισης NILM εξαρτάται σε μεγάλο βαθμό από την ποιότητα των συλλεγόμενων δεδομένων, την ύπαρξη επαρκούς εκπαίδευσης σε παρόμοια μοτίβα ενεργειακής χρήσης και τη δυνατότητα των αλγορίθμων να διαχωρίζουν πολλαπλές συσκευές που ενδέχεται να λειτουργούν ταυτόχρονα. Επίσης, η πολυπλοκότητα των μοντέλων μηχανικής μάθησης αυξάνεται όταν επιχειρείται υψηλή ακρίβεια στις εκτιμήσεις, ειδικά σε περιπτώσεις όπου το ενεργειακό προφίλ διαφέρει σημαντικά μεταξύ συσκευών ή περιβάλλοντος.[2]

Συσχέτιση με την παρούσα εργασία

Η προσέγγιση NILM είναι στενά συνυφασμένη με την προσπάθεια που περιγράφει η συγκεκριμένη διπλωματική, καθώς επιδιώκεται η ανάλυση και πρόβλεψη ποια συσκευή είναι ενεργή, χωρίς να εγκαθίστανται μεμονωμένοι αισθητήρες σε κάθε σημείο. Με τη χρήση αλγορίθμων μηχανικής μάθησης σε κεντρικές μετρήσεις και την αξιοποίηση σύγχρονων τεχνολογιών συλλογής και επεξεργασίας δεδομένων (όπως EMQX, Kafka, Flink και Kubernetes), μπορούμε να πετύχουμε αξιόπιστο διαχωρισμό της ενεργειακής κατανάλωσης, συνδυάζοντας την οικονομική αποδοτικότητα της μη παρεμβατικής μεθόδου με την ακρίβεια που προκύπτει από κατάλληλα εκπαιδευμένα μοντέλα.

Υπάρχουσες Τεχνικές

Οι τεχνικές ενεργειακής κατανομής επιδιώκουν να αναλύσουν το συνολικό σήμα κατανάλωσης και να το αποδώσουν σε επιμέρους συσκευές ή ομάδες συσκευών. Είτε ακολουθείται μια παρεμβατική προσέγγιση, είτε κάποια μη παρεμβατική, είναι αναγκαία μια μεθοδολογική εργαλειοθήκη που θα επιτρέψει την εξαγωγή αξιόπιστων χαρακτηριστικών και την ακριβή μοντελοποίηση της κατανάλωσης. Στη συνέχεια, παρουσιάζονται ορισμένες χαρακτηριστικές τεχνικές και προσεγγίσεις που έχουν προταθεί στη βιβλιογραφία.

Μετασχηματισμός Σήματος

Στην κατηγορία αυτή, βασικό βήμα αποτελεί η επεξεργασία και ανάλυση του ηλεκτρικού σήματος μέσα από κατάλληλους μετασχηματισμούς, ώστε να αποκαλυφθούν πρότυπα που διαφορετικά θα ήταν δυσδιάκριτα:

Μετασχηματισμός Fourier (FFT):

Μέσω του μετασχηματισμού Fourier στο σήμα, έχουμε μια αναπαράσταση από το πεδίο του χρόνου στο πεδίο της συχνότητας. Κάθε συσκευή έχει το δικό της *«συχνοτικό αποτύπωμα»*, δηλαδή ένα ξεχωριστό μοτίβο συχνοτήτων που παράγεται κατά τη λειτουργία της. Για παράδειγμα, οι κινητήρες και οι συμπιεστές, όπως αυτοί που βρίσκονται σε ψυγεία και κλιματιστικά, εμφανίζουν χαρακτηριστικές χαμηλές συχνότητες και αρμονικές που σχετίζονται με τη βασική συχνότητα του ηλεκτρικού δικτύου (50Hz ή 60Hz). Οι λαμπτήρες LED και άλλες ηλεκτρονικές συσκευές παράγουν σήματα υψηλής συχνότητας λόγω των κυκλωμάτων τους, ενώ οι συσκευές θέρμανσης, όπως οι θερμοσίφωνες και οι ηλεκτρικές κουζίνες, παρουσιάζουν ένα πιο σταθερό μοτίβο κατανάλωσης με χαμηλές συχνοτικές συνιστώσες. Έτσι, μπορούμε να αναλύσουμε τις συχνότητες που υπάρχουν στο συνολικό σήμα κατανάλωσης ενός νοικοκυριού και να τις αντιστοιχίσουμε σε συγκεκριμένες συσκευές. Η διαδικασία περιλαμβάνει τη συλλογή δεδομένων κατανάλωσης, την εφαρμογή του Μ/Σ Fourier (FFT) για τη δημιουργία της αναπαράστασης στο πεδίο των συχνοτήτων και την ανάλυση των χαρακτηριστικών που εμφανίζονται. Ο FFT είναι ιδιαίτερα χρήσιμος για την αναγνώριση συσκευών που έχουν διακριτά συχνοτικά χαρακτηριστικά, όπως οι κινητήρες και τα ηλεκτρονικά κυκλώματα. Ωστόσο, δεν είναι πάντα αρκετός από μόνος του, καθώς ορισμένες συσκευές έχουν παρόμοια συχνοτικά χαρακτηριστικά, γεγονός που καθιστά δύσκολο τον ακριβή διαχωρισμό τους

Στη συνέχεια, αυτά τα χαρακτηριστικά μπορούν να χρησιμοποιηθούν για την αναγνώριση των συσκευών, είτε με τη χρήση κανόνων είτε σε συνδυασμό με αλγόριθμους μηχανικής μάθησης.[3]

Μετασχηματισμός Wavelet:

Σε αντίθεση με τον Μ/Σ Fourier, ο μετασχηματισμός wavelet (WT) επιτρέπει την ταυτόχρονη επισκόπηση χρόνου-συχνοτήτων, καθιστώντας ευκολότερη την ανίχνευση βραχυχρόνιων μεταβολών στο σήμα, όπως οι διακυμάνσεις κατά την εκκίνηση ή διακοπή μιας συσκευής.

Στατιστικές Μέθοδοι

Οι στατιστικές προσεγγίσεις επικεντρώνονται στη μοντελοποίηση της πιθανότητας εμφάνισης ή λειτουργίας συσκευών

Κρυφά Μαρκοβιανά Μοντέλα (HMMs):

Τα HMMs θεωρούν ότι η τρέχουσα κατάσταση λειτουργίας (on/off) ή τα στάδια λειτουργίας μιας συσκευής είναι *«κρυφά»,* ενώ οι μετρήσεις ισχύος αποτελούν τις παρατηρήσεις. Ένα HMM μπορεί να εκπαιδευτεί για κάθε τύπο συσκευής ή ομάδα συσκευών, επιτρέποντας την πρόβλεψη του πότε μια συσκευή βρίσκεται σε συγκεκριμένη κατάσταση.

Παραγοντικά Κρυφά Μαρκοβιανά Μοντέλα (Factorial HMMs):

Επεκτείνουν την απλή δομή των HMMs εισάγοντας πολλαπλές κρυφές διεργασίες ταυτόχρονα, αντανακλώντας ουσιαστικά τη συνύπαρξη διαφορετικών συσκευών. Κατ' αυτόν τον τρόπο, καταγράφεται καλύτερα η πολύπλοκη και αλληλεπιδραστική φύση του ολικού σήματος κατανάλωσης σε ένα νοικοκυριό ή βιομηχανικό χώρο.

Η στατιστική προσέγγιση είναι συχνά πιο εύκολη στην ερμηνεία (σε σχέση με τεχνολογίες βαθιάς μάθησης) και οι υπολογιστικές απαιτήσεις της είναι συνήθως μικρότερες· γεγονός που την καθιστά κατάλληλη για περιβάλλοντα με περιορισμένους πόρους ή για περιπτώσεις όπου ο χρόνος απόκρισης πρέπει να είναι σχετικά μικρός.

Μέθοδοι Βασισμένες σε Κανόνες

Οι τεχνικές αυτές αξιοποιούν ευρετικές τεχνικές και προκαθορισμένους κανόνες για την ταυτοποίηση συσκευών:

Καθορισμός κατωφλιών:

Αυτή η μέθοδος μολονότι είναι εξαιρετικά γρήγορη, είναι αρκετά περιορισμένη στο σύνολο των συσκευών που μπορεί να διακρίνει. Κανόνες ορίζουν κβαντισμένα πεδία όπου μεταφράζονται σε καταστάσεις λειτουργίας συσκευών. Παραδείγματος χάριν, μια ξαφνική αύξηση της κατανάλωσης κατά 100 W μπορεί να συσχετίζεται με ενεργοποίηση μιας συγκεκριμένης συσκευής.

Λογικές συνθήκες (if-then-else):

Χρησιμοποιούνται για να τακτοποιηθούν μοτίβα έναυσης και σβησίματος (on/off) βάσει συγκεκριμένων χαρακτηριστικών όπως ο χρόνος λειτουργίας και η αναμενόμενη διακύμανση στο σήμα.

Παρόλο που οι μέθοδοι αυτοί μπορούν να είναι πολύ γρήγοροι και ελαφρείς υπολογιστικά, τείνουν να αδυνατούν να παράσχουν ακριβή αποτελέσματα σε πιο περίπλοκα περιβάλλοντα όπου πολλές συσκευές λειτουργούν ταυτόχρονα με ποικίλα μοτίβα. Επιπλέον, η έντονη εξάρτηση σε χειροκίνητη ρύθμιση κανόνων μειώνει την επεκτασιμότητα αυτών των συστημάτων.

Μηχανική Μάθηση

Η μηχανική μάθηση (ML) έχει εξελιχθεί σε καίριας σημασίας τεχνολογία για την ενεργειακή κατανομή, ειδικά στη μη παρεμβατική μορφή (NILM). Τα σύγχρονα εργαλεία ML ενισχύουν σημαντικά την ακρίβεια αναγνώρισης συσκευών και τη δυνατότητα επεξεργασίας μεγάλων, ετερογενών ροών δεδομένων.

Επιβλεπόμενη Μάθηση (Supervised Learning)

Η επιβλεπόμενη μάθηση στη μη-επεμβατική ανάλυση φορτίων βασίζεται στη διαθεσιμότητα ετικετοποιημένων δεδομένων, όπου για κάθε καταγεγραμμένο δείγμα γνωρίζουμε τη συσκευή-στόχο ή την κατάστασή της. Η ουσία αυτής της προσέγγισης είναι ότι ο αλγόριθμος εκπαιδεύεται να αναγνωρίζει αντιστοιχίες μεταξύ των μετρήσεων και της συγκεκριμένης συσκευής, εκμεταλλευόμενος ενδείξεις όπως αιχμές στο σήμα, αρμονικά χαρακτηριστικά ή άλλες διακριτές μεταβολές ισχύος. Στο πλαίσιο αυτό, τα πολύ-επίπεδα νευρωνικά δίκτυα (MLPs) προσφέρουν τη δυνατότητα ανάλυσης και μοντελοποίηση μη γραμμικών σχέσεων ανάμεσα στο σήμα και τις ετικέτες, επιτρέποντας την αναγνώριση πολύπλοκων προτύπων κατανάλωσης που δεν θα ήταν ευδιάκριτα με απλούστερες μεθόδους

Μηχανές Υποστήριξης Διανυσμάτων (SVMs):

Οι μηχανές υποστήριξης διανυσμάτων (SVMs) έχουν αποδειχθεί ιδιαίτερα ισχυρές σε εφαρμογές ταξινόμησης, αξιοποιώντας κατάλληλες συναρτήσεις πυρήνα ώστε να εντοπίζουν όρια διαχωρισμού μεταξύ διαφορετικών κατηγοριών φορτίων. Το κύριο πλεονέκτημα της επιβλεπόμενης μάθησης έγκειται στη συχνά υψηλότερη ακρίβεια, ακριβώς επειδή οι ετικέτες επιτρέπουν στο μοντέλο να προσαρμόζεται πιο αποτελεσματικά στα ιδιαίτερα χαρακτηριστικά κάθε συσκευή [4]

Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

Σε περιπτώσεις όπου τα δεδομένα είναι μη επισημασμένα (δεν υπάρχουν προκαθορισμένες ετικέτες για κάθε συσκευή), οι αλγόριθμοι μη επιβλεπόμενης μάθησης προσπαθούν να βρουν εσωτερικές δομές στα δεδομένα, ομαδοποιώντας τιμές που μοιάζουν μεταξύ τους ή προσπαθούν να αναλύσουν το σήμα σαν χρονοσειρά.

Η μη επιβλεπόμενη μάθηση είναι ιδιαίτερα χρήσιμη σε αρχικά στάδια εγκατάστασης, όπου δεν είναι γνωστό εκ των προτέρων ποιες συσκευές ή κατηγορίες συσκευών υπάρχουν σε έναν χώρο.

Αλγόριθμοι συσταδοποίησης (Clustering algorithms)

k-means:

Το k-means clustering είναι ένας αλγόριθμος μη επιβλεπόμενης μάθησης που χρησιμοποιείται για την ομαδοποίηση δεδομένων σε συστάδες (clusters) με βάση την ομοιότητά τους. Η διαδικασία ξεκινά με την τυχαία τοποθέτηση K κεντροειδών (centroids), τα οποία αναπροσαρμόζονται επαναληπτικά έως ότου οι αποστάσεις μεταξύ των σημείων κάθε

συστάδας και του αντίστοιχου κεντροειδούς ελαχιστοποιηθούν. Για την μέτρηση των αποστάσεων χρησιμοποιείται συνήθως η Ευκλείδεια απόσταση.

Hierarchical Clustering:

Στο Hierarchical Agglomerative Clustering τα δεδομένα ομαδοποιούνται σε ιεραρχική δομή με βάση την ομοιότητά τους. Ξεκινά με κάθε σημείο δεδομένων ως ξεχωριστό cluster και συγχωνεύει επαναληπτικά τα πιο όμοια clusters έως ότου δημιουργηθεί ένα δενδροδιάγραμμα. Η απόσταση μεταξύ των clusters μπορεί να μετρηθεί με διάφορες μεθόδους, όπως τη μονή συνδεσιμότητα (single linkage) ή την πλήρη συνδεσιμότητα (complete linkage)

DBSCAN:

Το DBSCAN είναι ένας αλγόριθμος ομαδοποίησης που βασίζεται στην πυκνότητα των δεδομένων και μπορεί να αναγνωρίσει clusters οποιουδήποτε σχήματος, διαχωρίζοντας ταυτόχρονα τα θορυβώδη δεδομένα (outliers). Λειτουργεί ομαδοποιώντας σημεία δεδομένων που βρίσκονται εντός μιας καθορισμένης απόστασης εμβέλειας (ε - έψιλον), ενώ απομονώνει τα σημεία που δεν ανήκουν σε καμία πυκνή περιοχή.

Βαθιά Μάθηση (Deep Learning)

Συνελικτικά Νευρωνικά Δίκτυα (CNNs):

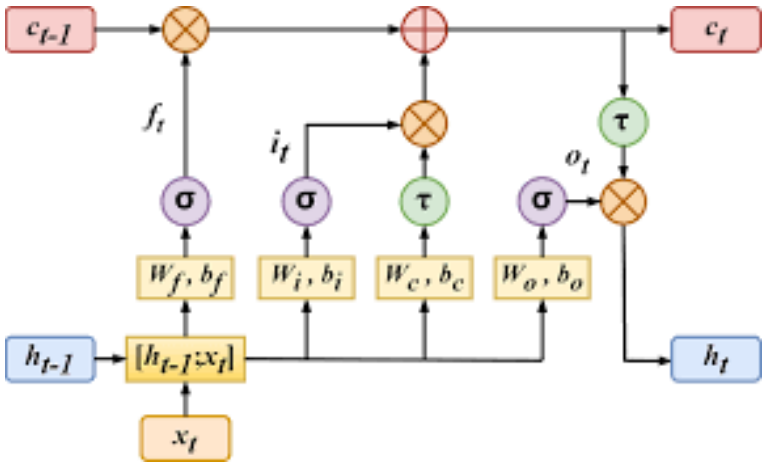
Τα συνελικτικά νευρωνικά δίκτυα (CNN) είναι μοντέλα βαθιάς μάθησης που έχουν σχεδιαστεί αρχικά για την επεξεργασία εικόνων, αλλά έχουν βρει εφαρμογές και σε άλλους τομείς, όπως η ανάλυση χρονικών σειρών. Βασίζεται στη χρήση συνελικτικών φίλτρων, τα οποία *«διασχίζουν»* το εισερχόμενο σήμα για να εξάγουν τοπικά μοτίβα και χαρακτηριστικά, όπως ακμές και γωνίες, μειώνοντας έτσι την πολυπλοκότητα των δεδομένων και επιτρέποντας στο μοντέλο να εστιάσει σε ουσιώδεις πληροφορίες. Αυτή η δομή τους τα καθιστά ιδιαίτερα αποτελεσματικά σε εργασίες ταξινόμησης και αναγνώρισης, όπου η χωρική οργάνωση των δεδομένων παίζει κρίσιμο ρόλο. [5]

Αναδρομικά Νευρωνικά Δίκτυα (RNNs, LSTMs):

Τα αναδρομικά νευρωνικά δίκτυα (RNN) έχουν σχεδιαστεί για την επεξεργασία σειριακών δεδομένων, όπου η χρονική διάσταση και η αλληλουχία των στοιχείων είναι καθοριστικής σημασίας. Μέσω της αναδρομικής σύνδεσης, κάθε χρονική στιγμή του δικτύου λαμβάνει υπόψη του όχι μόνο την τρέχουσα είσοδο, αλλά και την πληροφορία από προηγούμενα βήματα, δημιουργώντας έτσι μια μνήμη που βοηθά στην κατανόηση των μακροχρόνιων εξαρτήσεων. Αυτή η ιδιότητα καθιστά τα RNN ιδανικά για εφαρμογές όπως η επεξεργασία φυσικής γλώσσας, η ανάλυση σειρών χρόνου και άλλων διαδικασιών όπου η ακολουθία των δεδομένων είναι κρίσιμη.

Τα LSTM (Long Short-Term Memory) αποτελούν μια εξελιγμένη παραλλαγή των αναδρομικών νευρωνικών δικτύων, σχεδιασμένη για να ξεπεράσει τα προβλήματα που αντιμετωπίζουν τα παραδοσιακά RNN όσον αφορά τη μακροπρόθεσμη εξάρτηση. Μέσω της χρήσης εξειδικευμένων μηχανισμών, όπως οι πύλες εισόδου, λήθης και εξόδου, τα LSTM καταφέρνουν να ελέγχουν τη ροή των πληροφοριών, επιτρέποντας στο δίκτυο να διατηρεί σημαντικές πληροφορίες για μεγαλύτερα χρονικά διαστήματα και να απορρίπτει περιττές.

Αυτή η ικανότητα τα καθιστά εξαιρετικά αποτελεσματικά σε εφαρμογές που απαιτούν μακροπρόθεσμη μνήμη, όπως η μετάφραση κειμένου, η ανάλυση βίντεο και άλλες πολύπλοκες χρονικές ακολουθίες.[6]



Εικόνα 1: Κύτταρο LSTM

Συνεισφορά της Μηχανικής Μάθησης

Η ενσωμάτωση αλγορίθμων μηχανικής μάθησης στις τεχνικές ενεργειακής κατανομής επιφέρει σημαντικά πλεονεκτήματα όπως:

- **Αύξηση Ακρίβειας**:
  Τα μοντέλα ML, ιδίως τα βαθιά νευρωνικά δίκτυα, μπορούν να αναγνωρίσουν και να διαχωρίσουν ακόμα και συσκευές με πολύ παρόμοια μοτίβα κατανάλωσης.
- **Αυτοματοποίηση**:
  Η μείωση της ανάγκης ανθρώπινης επίβλεψης καθιστά τις λύσεις πιο εύχρηστες και αποδοτικές σε πραγματικά περιβάλλοντα.
- **Προσαρμοστικότητα**:
  Τα μοντέλα μπορούν να μαθαίνουν ή να ανανεώνονται με καινούρια δεδομένα, επιτρέποντας την παρακολούθηση συσκευών που εμφανίζονται με τον καιρό ή την αναπροσαρμογή σε αλλαγές συνθηκών (όπως η μείωση της τάσης δικτύου, φθορά συσκευής & μεταβολές στη σύνθεση του νοικοκυριού).

Χαρακτηριστικά Συνόλου Δεδομένων για την Εκπαίδευση Μοντέλων Ενεργειακής Αποσύνθεσης

Για την εκπαίδευση και αξιολόγηση μοντέλων μηχανικής μάθησης στο πλαίσιο της ενεργειακής κατανομής, απαιτείται η ύπαρξη συνόλων δεδομένων με σαφώς καθορισμένα και συνεπή χαρακτηριστικά. Το κάθε σύνολο δεδομένων (dataset) πρέπει να περιλαμβάνει τόσο αθροιστικές μετρήσεις (aggregate), δηλαδή τη συνολική κατανάλωση ενός χώρου, όσο και επιμερισμένες μετρήσεις ανά συσκευή, οι οποίες χρησιμοποιούνται ως δεδομένα αναφοράς (ground-truth). Ο συνδυασμός αυτός επιτρέπει την εποπτευόμενη εκπαίδευση μοντέλων, καθώς δίνει τη δυνατότητα να συσχετιστούν οι μεταβολές στο συνολικό σήμα με συγκεκριμένες συσκευές.

Η συχνότητα δειγματοληψίας αποτελεί κρίσιμο παράγοντα, καθώς επηρεάζει την ικανότητα του μοντέλου να ανιχνεύει μεταβατικές καταστάσεις και μικρές διακυμάνσεις ισχύος. Δεδομένα υψηλής χρονικής ανάλυσης αποτυπώνουν πληρέστερα τα χαρακτηριστικά λειτουργίας των συσκευών και διευκολύνουν τη διάκριση των ενεργειακών γεγονότων.

Παράλληλα, το σύνολο δεδομένων οφείλει να περιλαμβάνει ποικιλία συσκευών και καταστάσεων λειτουργίας, ώστε το μοντέλο να μπορεί να μάθει γενικεύσιμα μοτίβα κατανάλωσης. Αντίστοιχα, η πολυμορφία στο περιβάλλον συλλογής (διαφορετικά νοικοκυριά ή εγκαταστάσεις) ενισχύει τη δυνατότητα προσαρμογής του μοντέλου σε νέες συνθήκες. Επιπλέον, είναι απαραίτητη η ορθή ευθυγράμμιση χρονικών στιγμών και η διαχείριση ελλιπών ή θορυβωδών τιμών, ώστε τα δεδομένα να είναι καθαρά και συγκρίσιμα.[7]

Εν κατακλείδι, η ύπαρξη αναλυτικών μεταδεδομένων (τύπος συσκευής, χαρακτηριστικά μέτρησης, συνθήκες συλλογής) διευκολύνει την ερμηνεία και την επαναχρησιμοποίηση των δεδομένων, επιτρέποντας τη σύγκριση διαφορετικών πειραματικών προσεγγίσεων. Η ποιότητα και η πληρότητα του dataset αποτελούν θεμέλιο για την επιτυχή εκπαίδευση αλγορίθμων ενεργειακής αποσύνθεσης και τη γενίκευση των αποτελεσμάτων σε πραγματικά περιβάλλοντα.

Δομή Δεδομένων για Πρόβλεψη και Ανάλυση Χρονοσειρών

Η πρόβλεψη ενεργειακής κατανάλωσης και η κατανομή φορτίων αποτελούν κατεξοχήν προβλήματα ανάλυσης χρονοσειρών. Καθώς οι μετρήσεις ισχύος μεταβάλλονται και εξαρτώνται από προηγούμενες καταστάσεις του συστήματος, το dataset οφείλει να διατηρεί χρονική συνέχεια και ακριβή σειρά γεγονότων. Κάθε εγγραφή αντιπροσωπεύει μια χρονική στιγμή ή παράθυρο μετρήσεων (εύρος λίγων δευτερολέπτων), συνοδευόμενο από τις αντίστοιχες τιμές ισχύος, τάσης, ρεύματος ή άλλων συναφών μεγεθών.

Η δομή αυτή επιτρέπει στα μοντέλα πρόβλεψης να εντοπίζουν μοτίβα στο χρόνο, όπως επαναλαμβανόμενους κύκλους λειτουργίας ή σταδιακές αυξομειώσεις κατανάλωσης και να "μαθαίνουν" χρονικές εξαρτήσεις μεταξύ των δειγμάτων. Στην πράξη, η εκπαίδευση πραγματοποιείται σε παράθυρα ολίσθησης (sliding windows), όπου κάθε τμήμα δεδομένων χρησιμοποιείται για την πρόβλεψη της επόμενης χρονικής στιγμής ή της κατάστασης μιας συσκευής[8][9].

Επομένως, η σωστή αναπαράσταση του dataset ως χρονοσειρά, με πλήρη και συγχρονισμένα χρονικά σημεία, αποτελεί βασική προϋπόθεση για την επιτυχή εφαρμογή αλγορίθμων πρόβλεψης και αποσύνθεσης κατανάλωσης.

Εφαρμογές της Ενεργειακής κατανομής

Η ενεργειακή κατανομή έχει εξελιχθεί σε ένα σημαντικό εργαλείο στον τομέα της ενέργειας, παρέχοντας λεπτομερή πληροφόρηση για την κατανάλωση ηλεκτρικής ενέργειας σε επίπεδο συσκευής. Οι εφαρμογές της είναι πολυάριθμες και καλύπτουν ένα ευρύ φάσμα τομέων, από την οικιακή χρήση έως τη βιομηχανία και τη διαχείριση δικτύων. Στην ενότητα αυτή, θα αναλύσουμε τις κύριες εφαρμογές της ενεργειακής αποσύνθεσης, εξετάζοντας πώς συμβάλλει στη βελτίωση της ενεργειακής αποδοτικότητας, της διαχείρισης και της βιωσιμότητας.

Οικιακές Εφαρμογές

Προσωπική Ενημέρωση και Ευαισθητοποίηση Καταναλωτών:

Η ενεργειακή κατανομή επιτρέπει στους οικιακούς χρήστες να αποκτήσουν λεπτομερή εικόνα της κατανάλωσης ενέργειας των επιμέρους συσκευών τους. Μέσω εφαρμογών και έξυπνων μετρητών, οι καταναλωτές μπορούν να παρακολουθούν σε πραγματικό χρόνο πόση ενέργεια καταναλώνει κάθε συσκευή, ενθαρρύνοντας την ευαισθητοποίησή τους σχετικά με την ενεργειακή χρήση. Αυτό οδηγεί σε πιο συνειδητές επιλογές, όπως η αντικατάσταση ενεργοβόρων συσκευών ή η αλλαγή συνηθειών για τη μείωση της κατανάλωσης.[10]

Μείωση Κόστους Ενέργειας:

Με την κατανόηση των προτύπων κατανάλωσης, οι καταναλωτές μπορούν να λάβουν μέτρα για τη μείωση του ενεργειακού τους κόστους. Για παράδειγμα, μπορούν να προγραμματίσουν τη χρήση ορισμένων συσκευών σε ώρες εκτός αιχμής, όπου τα τιμολόγια ενέργειας είναι χαμηλότερα, ή να απενεργοποιήσουν συσκευές που καταναλώνουν ενέργεια σε κατάσταση αναμονής.

Βελτίωση Άνεσης και Ευκολίας:

Οι έξυπνες οικιακές συσκευές που ενσωματώνουν τεχνολογίες ενεργειακής κατανομής μπορούν να προσαρμόζουν τη λειτουργία τους ανάλογα με τις ανάγκες του χρήστη, βελτιώνοντας την άνεση και την ευκολία.

Βιομηχανικές και Εμπορικές Εφαρμογές

Βελτιστοποίηση Ενεργειακής Απόδοσης

Στους βιομηχανικούς και εμπορικούς τομείς, η ενεργειακή κατανομή χρησιμοποιείται για την παρακολούθηση της κατανάλωσης ενέργειας σε διαφορετικά τμήματα ή μηχανήματα. Αυτό επιτρέπει τον εντοπισμό περιοχών όπου η ενεργειακή απόδοση μπορεί να βελτιωθεί, οδηγώντας σε μείωση του λειτουργικού κόστους και του περιβαλλοντικού αποτυπώματος.

Προληπτική Συντήρηση

Με την ανάλυση των προτύπων κατανάλωσης, είναι δυνατόν να εντοπιστούν ανωμαλίες που υποδεικνύουν πιθανά προβλήματα ή φθορές σε εξοπλισμό. Αυτό επιτρέπει τη διενέργεια προληπτικής συντήρησης πριν από την εμφάνιση σοβαρών βλαβών, μειώνοντας το χρόνο διακοπής λειτουργίας και τα κόστη επισκευής.

Διαχείριση Φορτίου και Απόκριση Ζήτησης

Οι επιχειρήσεις μπορούν να χρησιμοποιήσουν την ενεργειακή κατανομή για να διαχειριστούν το φορτίο τους πιο αποτελεσματικά. Μέσω προγραμμάτων απόκρισης ζήτησης, μπορούν να προσαρμόζουν την κατανάλωσή τους σε περιόδους υψηλής ζήτησης στο δίκτυο, συμβάλλοντας στη σταθερότητα του συστήματος και ενδεχομένως επωφελούμενες από οικονομικά κίνητρα.

Διαχείριση Δικτύων και Έξυπνα Δίκτυα

Βελτίωση Σταθερότητας και Αξιοπιστίας Δικτύου

Οι πάροχοι ενέργειας μπορούν να χρησιμοποιήσουν δεδομένα ενεργειακής κατανομής για να αποκτήσουν καλύτερη κατανόηση της ζήτησης σε διαφορετικές περιοχές και χρονικές στιγμές. Αυτό επιτρέπει την καλύτερη πρόβλεψη της ζήτησης και τη βελτιστοποίηση της παραγωγής και της διανομής ενέργειας, βελτιώνοντας τη σταθερότητα και την αξιοπιστία του δικτύου.

Ενσωμάτωση Ανανεώσιμων Πηγών Ενέργειας

Η ενεργειακή κατανομή διευκολύνει την ενσωμάτωση ανανεώσιμων πηγών ενέργειας στο δίκτυο, παρέχοντας λεπτομερή πληροφόρηση που απαιτείται για την εξισορρόπηση της διαλείπουσας φύσης των ανανεώσιμων πηγών, όπως η ηλιακή και η αιολική ενέργεια.[11]

Ανίχνευση Απωλειών και Απάτης

Μέσω της λεπτομερούς ανάλυσης της κατανάλωσης, οι πάροχοι μπορούν να εντοπίσουν ασυνήθιστα πρότυπα που υποδεικνύουν απώλειες ενέργειας ή περιπτώσεις κλοπής ηλεκτρικής ενέργειας, επιτρέποντας τη λήψη μέτρων για την αντιμετώπισή τους.

Περιβαλλοντικές και Κοινωνικές Επιπτώσεις

Μείωση Εκπομπών Διοξειδίου του Άνθρακα

Η βελτίωση της ενεργειακής αποδοτικότητας μέσω της ενεργειακής αποσύνθεσης συμβάλλει στη μείωση των εκπομπών CO_2, υποστηρίζοντας τους στόχους για την αντιμετώπιση της κλιματικής αλλαγής.

Προώθηση Βιώσιμης Κατανάλωσης

Ενθαρρύνοντας τους καταναλωτές να υιοθετήσουν πιο βιώσιμες πρακτικές, η ενεργειακή κατανομή συμβάλλει στην προώθηση μιας κουλτούρας βιώσιμης κατανάλωσης και περιβαλλοντικής συνείδησης.

Προσωποποιημένες Υπηρεσίες και Εξυπηρέτηση Πελατών

Εξατομικευμένες Προτάσεις Εξοικονόμησης Ενέργειας

Με βάση τα δεδομένα κατανάλωσης, οι πάροχοι ενέργειας και οι εταιρείες ενεργειακών υπηρεσιών μπορούν να προσφέρουν στους πελάτες εξατομικευμένες συμβουλές και προτάσεις για τη μείωση της κατανάλωσης και του κόστους.

Βελτίωση Εξυπηρέτησης Πελατών

Η λεπτομερής πληροφόρηση επιτρέπει την καλύτερη κατανόηση των αναγκών των πελατών, οδηγώντας σε βελτίωση των υπηρεσιών και της ικανοποίησης των πελατών.

.

Κεντρικά Συστήματα Εφαρμογών μεγάλης κλίμακας

Εισαγωγή

Στη σύγχρονη ψηφιακή εποχή, η τεχνολογική εξέλιξη έχει δημιουργήσει ένα περιβάλλον στο οποίο οι επιχειρήσεις και οι οργανισμοί εξαρτώνται ολοένα και περισσότερο από τις πληροφοριακές τεχνολογίες για την αποτελεσματική λειτουργία και ανταπόκρισή τους στις απαιτήσεις της αγοράς. Τα κεντρικά συστήματα εφαρμογών μεγάλης κλίμακας διαδραματίζουν κρίσιμο ρόλο στην υποστήριξη κάθε κρίσιμης επιχειρησιακής λειτουργίας, επιτρέποντας τη διαχείριση τεράστιου όγκου δεδομένων και την παράλληλη λειτουργία πολλαπλών εφαρμογών, οι οποίες συχνά διαφέρουν ως προς την τεχνολογία, την αρχιτεκτονική και τις επιχειρησιακές απαιτήσεις τους.

Ιστορική Εξέλιξη και Θεμελιώδεις Αρχές

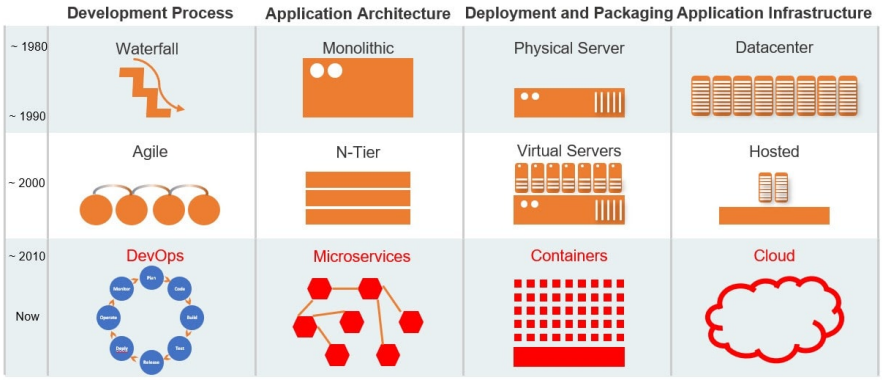
Παραδοσιακά, τα κεντρικά συστήματα είχαν σχεδιαστεί για να υποστηρίζουν σταθερές, προβλέψιμες ανάγκες με έμφαση στην αξιοπιστία και την ασφάλεια. Ωστόσο, η ραγδαία αύξηση του όγκου των δεδομένων και η πολυπλοκότητα των επιχειρησιακών λειτουργιών έχουν αναδείξει περιορισμούς στις παραδοσιακές προσεγγίσεις, όπως η έλλειψη επεκτασιμότητας και η δυσκολία στην ταχεία υλοποίηση νέων λειτουργιών. Σήμερα, οι απαιτήσεις για υψηλή διαθεσιμότητα, ταχύτητα επεξεργασίας και ευελιξία στην ανάπτυξη αναγκάζουν τους οργανισμούς να επανεξετάσουν τις υφιστάμενες τεχνολογικές λύσεις και να υιοθετήσουν νέες αρχιτεκτονικές που μπορούν να ανταπεξέλθουν στις προκλήσεις ενός συνεχώς μεταβαλλόμενου περιβάλλοντος.

Κύριες Προκλήσεις των Κεντρικών Συστημάτων Εφαρμογών

Η διαχείριση και λειτουργία των κεντρικών συστημάτων εφαρμογών μεγάλης κλίμακας ενέχει πληθώρα προκλήσεων, μεταξύ των οποίων:

1. Κλιμακωσιμότητα και ευελιξία:
   Η παραδοσιακή κεντρική επεξεργασία δυσκολεύεται να ανταποκριθεί στις μεταβαλλόμενες ανάγκες των επιχειρήσεων, όπου οι αιχμές στη ζήτηση και η ανάγκη για γρήγορη ανάπτυξη νέων λειτουργιών απαιτούν δυναμικές λύσεις.
2. Αυτοματοποίηση και διαχείριση:
   Η ενσωμάτωση αυτοματοποιημένων διαδικασιών στην παραδοσιακή διαχείριση των συστημάτων παραμένει συχνά ανεπαρκής, οδηγώντας σε αυξημένη πιθανότητα σφαλμάτων και δυσκολίες στη συντήρηση και αναβάθμιση των υποδομών.
3. Ασφάλεια και αξιοπιστία:
   Η διαρκής απειλή από κυβερνοεπιθέσεις και άλλες μορφές ηλεκτρονικών κινδύνων καθιστούν την ενσωμάτωση προηγμένων μηχανισμών ασφάλειας αναγκαία, προκειμένου να διασφαλιστεί η ακεραιότητα και η διαθεσιμότητα των κρίσιμων υπηρεσιών.[12]

Η αντιμετώπιση των παραπάνω προκλήσεων έχει οδηγήσει στην ανάπτυξη νέων τεχνολογικών προσεγγίσεων και μεθοδολογιών.



Εικόνα 2: Εξέλιξη μεθόδων και αρχιτεκτονικών εφαρμογών

Η χρήση μικροϋπηρεσιών επιτρέπει τη διαχωρισμένη ανάπτυξη και κλιμάκωση των εφαρμογών, διευκολύνοντας την ταχύτερη ενσωμάτωση νέων λειτουργιών και την προσαρμογή στις μεταβαλλόμενες ανάγκες των επιχειρήσεων.

Η υιοθέτηση υπηρεσιών Cloud computing, επιτρέπει την οριζόντια και κάθετη κλιμάκωση των υποδομών, καθώς και την ευέλικτη κατανομή των πόρων, μειώνοντας ταυτόχρονα το κόστος και την πολυπλοκότητα στη διαχείριση.

Τέλος, οι πρακτικές διαδικασιών συνεχούς ενσωμάτωσης και ανάπτυξης, συμβάλλουν στη μείωση του χρόνου ανάπτυξης και στην αύξηση της αξιοπιστίας των συστημάτων μέσω αυτοματοποιημένων διαδικασιών δοκιμών και παρακολούθησης.

Η ενότητα που ακολουθεί θα εστιάσει στις προκλήσεις που αντιμετωπίζουν οι σύγχρονες επιχειρήσεις εξαιτίας των περιορισμών των παραδοσιακών κεντρικών συστημάτων επεξεργασίας και θα προετοιμάσει το έδαφος για την παρουσίαση μεθόδων και εργαλείων που έχουν αναπτυχθεί με στόχο να αντιμετωπίσουν τις ανάγκες αυτής της νέας εποχής.[13]

Συστοιχίες Μηχανημάτων (Clustering)

Το clustering αναφέρεται στη σύνδεση πολλαπλών διακομιστών ή κόμβων για να λειτουργήσουν ως ένα ενιαίο σύστημα. Αυτή η αρχιτεκτονική ενισχύει την απόδοση, τη διαθεσιμότητα και την επεκτασιμότητα, κατανέμοντας τα φορτία εργασίας σε πολλαπλές μηχανές. Τα clusters είναι καίρια για τη διαχείριση μεγάλων εφαρμογών, διασφαλίζοντας ελάχιστο χρόνο διακοπής και παρέχοντας ανοχή σε σφάλματα.

Σε ένα περιβάλλον clustering, εάν ένας κόμβος αποτύχει, οι υπόλοιποι μπορούν να αναλάβουν το φορτίο εργασίας, ελαχιστοποιώντας τις διακοπές στην υπηρεσία - μια έννοια γνωστή ως υψηλή διαθεσιμότητα. Το load balancing είναι επίσης μια κομβική λειτουργία σε cluster υποδομές, καθώς το σύστημα κατανέμει την εισερχόμενη δικτυακή κίνηση σε πολλαπλούς

διακομιστές για να αποτρέψει την υπερφόρτωση ενός μεμονωμένου κόμβου. Η επεκτασιμότητα επιτυγχάνεται με την προσθήκη περισσότερων κόμβων στο cluster, επιτρέποντας στο σύστημα να ανταποκρίνεται εύκολα σε αυξημένες απαιτήσεις. Η ανοχή σε σφάλματα διασφαλίζει ότι το σύστημα συνεχίζει να λειτουργεί κανονικά, ακόμα κι αν κάποια από τα στοιχεία του αποτύχουν.

Συστήματα Ενορχήστρωσης (Orchestrators)

Η ταχεία εξέλιξη του cloud computing, σε συνδυασμό με την καθιέρωση της τεχνολογίας των container ως βασικός τρόπος ανάπτυξης εφαρμογών, οδήγησε στην ανάγκη δημιουργίας ευέλικτων και αυτοματοποιημένων συστημάτων διαχείρισης υποδομής.

Τα συστήματα ενορχήστρωσης (orchestrators), τα οποία αναλαμβάνουν να διαχειρίζονται συστοιχίες υπολογιστικών κόμβων και εκατοντάδων ή χιλιάδων container, προσφέρουν ολοκληρωμένες λύσεις για την ανάπτυξη, την παρακολούθηση, την κλιμάκωση, αλλά και την ανάκαμψη των εφαρμογών σε περίπτωση σφαλμάτων ή αστοχιών.

Μεταξύ των πλέον διαδεδομένων συστημάτων ενορχήστρωσης συγκαταλέγονται το Docker Swarm, το Apache Mesos και, κυρίως, το Kubernetes. Όλα τα παραπάνω διαθέτουν δικά τους χαρακτηριστικά που τους προσδίδουν ξεχωριστές ιδιαιτερότητες, σκοπός όλων είναι η απλοποίηση και αυτοματοποίηση της λειτουργίας πολύπλοκων περιβαλλόντων οπού αναπτύσσουμε εφαρμογές ενσωματωμένες σε containers. Ακολουθεί μια ανάλυση των βασικών στόχων που επιτυγχάνουν τα συστήματα ενορχήστρωσης, ενδεικτικά:

Απλούστευση ανάπτυξης και συντήρησης:

- Τα συστήματα ενορχήστρωσης παρακολουθούν διαρκώς τους κόμβους του cluster και τις εφαρμογές που εκτελούνται πάνω σε αυτούς.
- Σε περίπτωση που κάτι πάει στραβά (π.χ. ένα container σταματήσει να είναι λειτουργικό ή ένας κόμβος βγει εκτός λειτουργίας), αυτοματοποιούν δράσεις όπως η επανεκκίνηση ή η μεταφορά τους σε άλλον διαθέσιμο κόμβο.
- Η διαδικασία αυτή μειώνει την ανθρώπινη παρέμβαση και ταυτόχρονα επιταχύνει την αντιμετώπιση σφαλμάτων, βελτιώνοντας τη σταθερότητα τη υποδομής.

Διασφάλιση υψηλής διαθεσιμότητας (HA):

- Η συνεχής λειτουργία των υπηρεσιών είναι κρίσιμη για τις περισσότερες επιχειρήσεις.
- Εάν κάποιο σημείο του συστήματος παρουσιάσει βλάβη, ο ρόλος του orchestrator είναι να “διατηρήσει τη ροή” επαναφέροντας αυτόματα τις υπηρεσίες. Με αυτόν τον τρόπο, ελαχιστοποιείται ο χρόνος αδράνειας μια υπηρεσίας και προστατεύεται η εμπειρία των χρηστών.

Δυναμική κλιμάκωση:

- Ένα από τα κύρια πλεονεκτήματα των microservices είναι η ευκολία με την οποία μπορούμε να αναπτύξουμε πολλαπλά instances της ίδιας υπηρεσίας (scale out).

- Τα συστήματα ενορχήστρωσης επιτρέπουν την αυτόματη κλιμάκωση της υποδομής βάσει πραγματικού φορτίου ή συγκεκριμένων μετρικών.
- Επίσης, όταν η ζήτηση μειωθεί, τα επιπλέον instances τερματίζονται (scale in), απελευθερώνοντας πόρους ώστε να επιτύχουμε μείωση του κόστους.

Κεντρικός τροπος παραμετροποίησης (Centralized Configuration Management):

- Οι orchestrators παραμετροποιούνται μέσω δηλωτικών αρχείων (manifests), συνήθως σε μορφή YAML ή JSON, για να καθορίσουν την επιθυμητή κατάσταση (desired state) των εφαρμογών.
- Στα αρχεία αυτά περιγράφονται σημαντικές λεπτομέρειες, όπως ο αριθμός των instance της εφαρμογής που πρέπει να τρέχουν, οι κανόνες δικτύωσης, οι απαιτούμενοι πόροι και οι τρόποι ενημέρωσης.
- Αυτή η πρακτική βοηθά στην ευκολότερη αναπαραγωγή περιβαλλόντων σε διαφορετικά στάδια (development, staging, production) και διευκολύνει την αντιμετώπιση προβλημάτων με την καταγραφή ακριβώς του τι τρέχει πού.

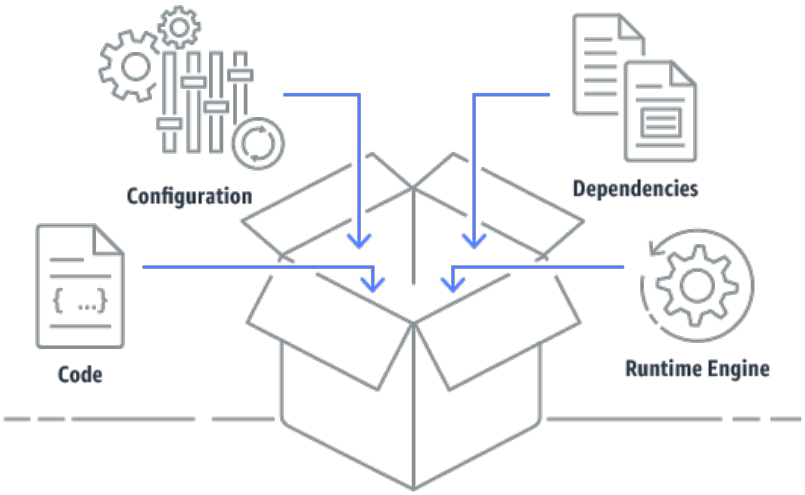
Τέλος, αξίζει να σημειωθεί ότι κάθε σύστημα ενορχήστρωσης έχει τα δικά του πλεονεκτήματα και αδυναμίες, όπως διαφορετικά μοντέλα ασφάλειας, τρόπους οργάνωσης υπηρεσιών και εργαλεία διαχείρισης. Ωστόσο, σε όλες τις περιπτώσεις, ο κεντρικός στόχος παραμένει κοινός: η αποτελεσματική και αυτοματοποιημένη διαχείριση των container και των υποδομών, ώστε οι ομάδες ανάπτυξης λογισμικού και οι διαχειριστές συστημάτων να μπορούν να ανταποκρίνονται σε σύνθετες απαιτήσεις χωρίς να αυξάνεται υπερβολικά η επιχειρησιακή πολυπλοκότητα.

Το σύστημα Kubernetes

Το Kubernetes, συχνά αναφερόμενο ως K8s (Κυβερνήτης), είναι μια πλατφόρμα ανοικτού κώδικα σχεδιασμένη να αυτοματοποιεί την ανάπτυξη, την επεκτασιμότητα και τη λειτουργία εφαρμογών. Αρχικά αναπτύχθηκε από την Google και τώρα συντηρείται από το Cloud Native Computing Foundation (CNCF), ενώ έχει εδραιωθεί ως η πιο σύνηθες πρακτική για κλιμακούμενα παραγωγικά συστήματα.

Βασικές Έννοιες

Στην καρδιά του Kubernetes βρίσκεται η διαχείριση των **containers**, τα οποία είναι ελαφριές, φορητές εκτελέσιμες εικόνες που περιλαμβάνουν το λογισμικό και όλες τις εξαρτήσεις του. Τα container επιτρέπουν στις εφαρμογές να εκτελούνται με συνέπεια σε διαφορετικά περιβάλλοντα, από την ανάπτυξη έως την παραγωγή



.

Εικόνα 3: Απλοϊκή αναπαράσταση της τεχνολογίας των container

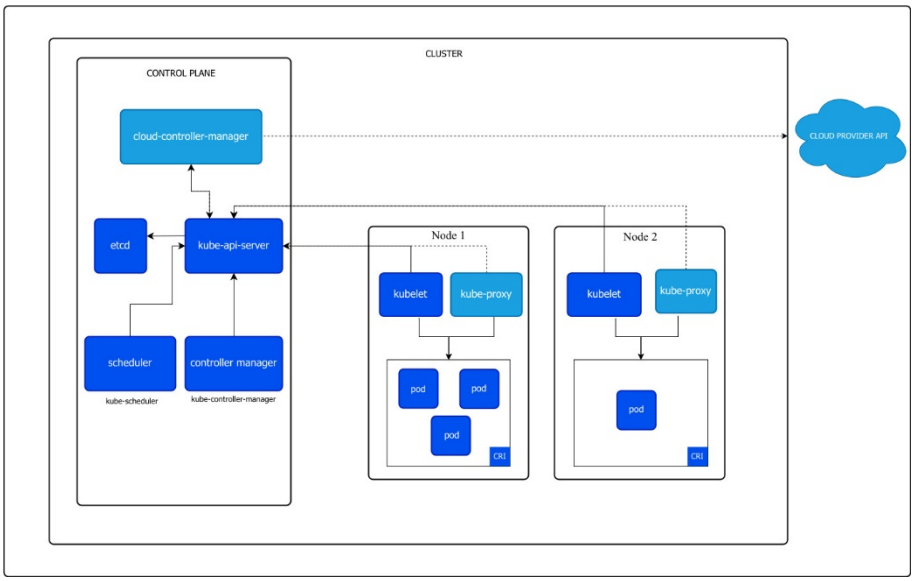
Η μικρότερη μονάδα ανάπτυξης στο Kubernetes είναι το **pod**. Ένα pod αντιπροσωπεύει ένα μεμονωμένο στιγμιότυπο μιας διαδικασίας και μπορεί να περιλαμβάνει ένα ή περισσότερα container που μοιράζονται αποθηκευτικό χώρο, δίκτυο και μια προδιαγραφή για το πώς να εκτελούνται τα container. Τα pods είναι εφήμερα από τη φύση τους: αν ένα pod αποτύχει, το Kubernetes θα δημιουργήσει ένα νέο βάσει της καθορισμένης επιθυμητής κατάστασης.

Nodes

Τα Nodes είναι τα φυσικά ή εικονικά μηχανήματα που φιλοξενούν τα Pods. Διακρίνονται σε δύο κατηγορίες: τα Master Nodes και τα Worker Nodes. Τα Master Nodes εκτελούν διεργασίες που αφορούν τη διαχείριση του cluster, όπως ο προγραμματισμός των Pods και η διατήρηση της κατάστασης του συστήματος. Τα Worker Nodes φιλοξενούν τα Pods που εκτελούν τις εφαρμογές. Στην περίπτωση του Apache Kafka, τα Worker Nodes θα φιλοξενούσαν τα Pods που τρέχουν τους Kafka Brokers, επιτρέποντας την επεξεργασία των ροών δεδομένων.

Αρχιτεκτονική του Kubernetes

Το Kubernetes ακολουθεί μια αρχιτεκτονική master-worker, που αποτελείται από το **Control Plane** (κύριος κόμβος) και τους **κόμβους εργασίας** (Worker Nodes).



Εικόνα 4:Αρχιτεκτονική Kubernetes Cluster

Στοιχεία του Control Plane:

- **API Server (kube-apiserver)**: Λειτουργεί ως το κεντρικό διαχειριστικό στοιχείο του Kubernetes, εκθέτοντας το Kubernetes API. Είναι η κύρια οντότητα διαχείρισης που επεξεργάζεται τις λειτουργίες REST, τις επικυρώνει και ενημερώνει την κατάσταση του cluster ανάλογα.
- **etcd**: Ένα συνεπές και ιδιαίτερα διαθέσιμο αποθηκευτικό σύστημα τύπου κλειδιού-τιμής που χρησιμοποιείται ως το αποθετήριο δεδομένων του Kubernetes για όλα τα δεδομένα του cluster. Αποθηκεύει τα δεδομένα διαμόρφωσης, τα οποία αντιπροσωπεύουν την επιθυμητή κατάσταση του cluster ανά πάσα στιγμή.
- **Scheduler (kube-scheduler)**: Αναθέτει pods σε κόμβους βάσει των απαιτήσεων πόρων και της διαθεσιμότητας, καθώς και άλλων περιορισμών και πολιτικών. Διασφαλίζει ότι τα φορτία εργασίας κατανέμονται αποδοτικά στο cluster.
- **Controller Manager (kube-controller-manager)**: Εκτελεί τις διεργασίες των controllers που ρυθμίζουν την κατάσταση του cluster, όπως οι controllers κόμβων, οι controllers αντιγραφής και οι controllers τερματικών σημείων (endpoints). Οι controllers είναι υπεύθυνοι για την παρακολούθηση των αλλαγών στο cluster και τη διασφάλιση ότι η τρέχουσα κατάσταση αντιστοιχεί στην επιθυμητή.

Στοιχεία ενός Worker Node:

- **Kubelet**: Ένας agent που τρέχει σε κάθε κόμβο εργασίας, διασφαλίζοντας ότι τα container εκτελούνται σε pods όπως αναμένεται. Επικοινωνεί με τον API server και διαχειρίζεται τα container μέσω του container runtime.
- **Container Runtime**: Το λογισμικό που είναι υπεύθυνο για την εκτέλεση των container. Τα πιο διαδεδομένα ειναι το Docker, το containerd και το CRI-O.

- **Kube-proxy**: Διατηρεί τους κανόνες δικτύου στους κόμβους και επιτρέπει την επικοινωνία του δικτύου με τα pods από εσωτερικές ή εξωτερικές συνδέσεις του δικτύου. Διαχειρίζεται την εκχώρηση διευθύνσεων IP.[14]

Χαρακτηριστικά και πλεονεκτήματα του Kubernetes

Το **Kubernetes** προσφέρει μια σειρά ισχυρών χαρακτηριστικών που το καθιστούν την ιδανική λύση για την ενορχήστρωση container:

Αυτοματοποιημένες Αναβαθμίσεις και Ανάκληση Εκδόσεων: Το Kubernetes μπορεί να διαχειριστεί την ανάπτυξη νέων εκδόσεων εφαρμογών και να επιστρέψει σε προηγούμενες εκδόσεις εάν εντοπιστούν προβλήματα. Αυτή η αυτοματοποίηση μειώνει το downtime και μετριάζει τον κίνδυνο που σχετίζεται με τις ενημερώσεις.

Δημιουργία υπηρεσιών εξυπηρέτησης και Load Balancing: Το Kubernetes μπορεί να εκθέτει container χρησιμοποιώντας ονόματα DNS ή διευθύνσεις IP και να ισορροπεί την κυκλοφορία του δικτύου για να διασφαλίσει τη σταθερότητα και την αποδοτική χρήση των πόρων.

Self-healing: Το σύστημα επανεκκινεί αυτόματα τα αποτυχημένα container, αντικαθιστά container σε περίπτωση αποτυχίας ενός κόμβου και διακόπτει container που δεν ανταποκρίνονται στους ελέγχους υγείας που έχει ορίσει ο χρήστης.

Οριζόντια Κλιμάκωση: Οι εφαρμογές μπορούν να κλιμακωθούν αυτόματα με βάση τους πόρους που χρησιμοποιούν, όπως η χρήση CPU, ή χειροκίνητα, ανάλογα με τις ανάγκες.

Βασικές οντότητες του συστήματος Kubernetes

Εισαγωγή

Η ανάπτυξη εφαρμογών στο Kubernetes περιλαμβάνει συνήθως τον ορισμό της επιθυμητής κατάστασης της εφαρμογής και των στοιχείων της χρησιμοποιώντας αρχεία **YAML** ή **JSON**. Αυτά τα αρχεία διαμόρφωσης, γνωστά ως manifest, περιγράφουν τους πόρους που χρειάζονται, όπως αναπτύξεις, υπηρεσίες και τόμους.

Χρησιμοποιώντας το εργαλείο γραμμής εντολών **kubectl**, εφαρμόζουμε αυτές τις διαμορφώσεις στο cluster. Το control plane του Kubernetes φροντίζει να διασφαλίσει ότι η πραγματική κατάσταση του cluster αντιστοιχεί στην επιθυμητή κατάσταση που περιγράφεται στα αρχεία διαμόρφωσης. Αυτή η διαδικασία περιλαμβάνει τη δημιουργία pods, τη διατήρηση της υγείας τους και την κλιμάκωσή τους ανάλογα με τις ανάγκες

Pod: Η βασική μονάδα ανάπτυξης στο Kubernetes. Ένα Pod αντιπροσωπεύει ένα ή περισσότερα container που είναι στενά συνδεδεμένα και μοιράζονται το ίδιο network namespace και αποθηκευτικούς τόμους. Τα Pods είναι οι μικρότερες δομικές μονάδες.

Service: Ένα αντικείμενο που ορίζει ένα λογικό σύνολο Pods και μια πολιτική πρόσβασης σε αυτά. Τα Services παρέχουν μια σταθερή διεύθυνση IP και όνομα DNS για ένα σύνολο Pods, επιτρέποντας σε άλλα μέρη της εφαρμογής σας ή σε εξωτερικούς πελάτες να επικοινωνούν με αυτά με αξιοπιστία, ακόμα και αν τα Pods δημιουργούνται και καταστρέφονται.

Deployment: Διαχειρίζεται ένα σύνολο πανομοιότυπων Pods και διασφαλίζει ότι εκτελούνται ο επιθυμητός αριθμός αντιγράφων (replicas). Τα Deployments επιτρέπουν την ομαλή ενημέρωση των εφαρμογών, εφαρμόζοντας αλλαγές σταδιακά και επιστρέφοντας στην προηγούμενη κατάσταση αν χρειαστεί.

ReplicaSet: Διασφαλίζει ότι εκτελούνται ένας συγκεκριμένος αριθμός αντιγράφων Pods σε κάθε δεδομένη στιγμή. Τα ReplicaSets χρησιμοποιούνται από τα Deployments για να διατηρούν τον επιθυμητό αριθμό Pods.

StatefulSet: Διαχειρίζεται την ανάπτυξη και την κλιμάκωση των Pods που απαιτούν επίμονο (persistent) state ή σταθερές δικτυακές ταυτότητες. Τα StatefulSets χρησιμοποιούνται για εφαρμογές που απαιτούν state, όπως οι βάσεις δεδομένων, όπου κάθε Pod χρειάζεται μια μοναδική ταυτότητα και σταθερό αποθηκευτικό χώρο.

DaemonSet: Διασφαλίζει ότι ένα συγκεκριμένο Pod εκτελείται σε όλους ή σε μερικούς από τους κόμβους στο cluster. Τα DaemonSets είναι χρήσιμα για την εκτέλεση υπηρεσιών σε επίπεδο συστήματος, όπως συλλέκτες log ή monitoring agents σε κάθε κόμβο.

Job: Δημιουργεί ένα ή περισσότερα Pods για να εκτελέσουν μια εργασία και διασφαλίζει ότι ένας συγκεκριμένος αριθμός αυτών ολοκληρώνεται με επιτυχία. Τα Jobs χρησιμοποιούνται για εργασίες μικρής διάρκειας, όπως επεξεργασία δεδομένων ή batch jobs.

CronJob: Προγραμματίζει Jobs να εκτελούνται σε συγκεκριμένες ώρες ή χρονικά διαστήματα, παρόμοια με τα cron jobs σε συστήματα Unix. Τα CronJobs είναι χρήσιμα για περιοδικές εργασίες όπως backups ή δημιουργία αναφορών.

ConfigMap: Αποθηκεύει δεδομένα διαμόρφωσης σε ζεύγη κλειδιού-τιμής. Τα ConfigMaps καθιστούν δυνατό τον διαχωρισμό των ρυθμίσεων μιας εφαρμογής από τον υπό εκτέλεση κώδικα, κάνοντας τις εφαρμογές φορητές και εύκολες στη διαχείριση.

Secret: Παρόμοια με τα ConfigMaps, αλλά σχεδιασμένα για την αποθήκευση ευαίσθητων πληροφοριών όπως κωδικοί πρόσβασης, tokens ή κλειδιά. Τα Secrets παρέχουν έναν ασφαλή τρόπο αποθήκευσης και διαχείρισης εμπιστευτικών δεδομένων που χρειάζονται οι εφαρμογές.

PersistentVolume (PV): Αντιπροσωπεύει ένα κομμάτι αποθήκευσης στο cluster που έχει παρασχεθεί από έναν διαχειριστή ή δυναμικά μέσω ενός StorageClass. Τα PVs είναι πόροι στο cluster που παρέχουν ανθεκτική αποθήκευση ανεξάρτητη από τον κύκλο ζωής των Pods.

PersistentVolumeClaim (PVC): Μια αίτηση αποθήκευσης από έναν χρήστη ή μια εφαρμογή. Τα PVCs ζητούν πόρους αποθήκευσης από τα PVs βάσει μεγέθους και τρόπων πρόσβασης. Τα Pods χρησιμοποιούν τα PVCs για να προσαρτήσουν αποθηκευτικούς τόμους και να αποκτήσουν πρόσβαση σε δεδομένα.

Ingress: Διαχειρίζεται την εξωτερική πρόσβαση σε υπηρεσίες εντός του cluster, συνήθως μέσω HTTP ή HTTPS. Το Ingress ορίζει κανόνες για τη δρομολόγηση εξωτερικής κίνησης στις κατάλληλες υπηρεσίες βάσει ονομάτων host ή μονοπατιών. Μπορεί επίσης να χειριστεί load balancing, SSL termination και εικονική φιλοξενία (virtual hosting).

Network Policy: Ορίζει κανόνες για το πώς επιτρέπεται στα Pods να επικοινωνούν μεταξύ τους και με εξωτερικά δίκτυα. Τα NetworkPolicies παρέχουν έναν τρόπο ελέγχου της ροής της κυκλοφορίας στο επίπεδο διεύθυνσης IP ή πόρτας, ενισχύοντας την ασφάλεια στο cluster.

Namespace: Παρέχει έναν τρόπο διαίρεσης των πόρων του cluster μεταξύ πολλών χρηστών ή ομάδων. Τα Namespaces δημιουργούν εικονικά clusters μέσα σε ένα φυσικό cluster, επιτρέποντας την απομόνωση και την οργάνωση των πόρων.

Endpoint: Κρατά πληροφορίες σχετικά με τις διευθύνσεις δικτύου των Pods που αντιστοιχίζονται από έναν Service selector. Τα Endpoints χρησιμοποιούνται εσωτερικά από το Kubernetes για να παρακολουθεί ποια Pods παρέχουν μια συγκεκριμένη υπηρεσία.

HorizontalPodAutoscaler: Προσαρμόζει αυτόματα τον αριθμό των Pods σε ένα Deployment, ReplicaSet ή StatefulSet βάσει των παρατηρούμενων μετρικών όπως η χρήση CPU. Αυτό βοηθά τις εφαρμογές να κλιμακώνονται ανάλογα με τη ζήτηση.

VerticalPodAutoscaler: Προσαρμόζει αυτόματα τις αιτήσεις και τα όρια για CPU και μνήμη για τα container μέσα σε ένα Pod. Βοηθά στη βελτιστοποίηση της χρήσης πόρων, προτείνοντας ή επιβάλλοντας τις κατάλληλες ρυθμίσεις πόρων.

StorageClass: Περιγράφει διαφορετικούς τύπους αποθήκευσης που είναι διαθέσιμοι στο cluster, συνήθως αντιστοιχώντας σε επίπεδα ποιότητας υπηρεσίας ή πολιτικές δημιουργίας αντιγράφων ασφαλείας. Τα StorageClasses επιτρέπουν τη δυναμική παροχή PersistentVolumes.

LimitRange: Καθορίζει ελάχιστα και μέγιστα όρια χρήσης πόρων για Pods ή Containers μέσα σε ένα Namespace. Τα LimitRanges βοηθούν στην πρόληψη της υπερβολικής κατανάλωσης πόρων από μεμονωμένες εφαρμογές.

ResourceQuota: Ορίζει περιορισμούς στο συνολικό ποσό των πόρων που μπορούν να καταναλωθούν σε ένα Namespace. Τα ResourceQuotas διασφαλίζουν τη δίκαιη κατανομή των πόρων του cluster μεταξύ διαφορετικών έργων ή ομάδων.

ServiceAccount: Παρέχει μια ταυτότητα για διεργασίες που εκτελούνται σε ένα Pod για να αλληλεπιδράσουν με το Kubernetes API. Τα ServiceAccounts χρησιμοποιούνται για έλεγχο ταυτότητας και έλεγχο πρόσβασης μέσα στο cluster.

Role and ClusterRole: Ορίζουν δικαιώματα για να εκτελούν ενέργειες σε πόρους του Kubernetes. Τα Roles είναι περιορισμένα σε ένα Namespace, ενώ τα ClusterRoles έχουν δικαιώματα σε ολόκληρο το cluster. Καθορίζουν ποιες ενέργειες επιτρέπονται σε ποιους πόρους.

RoleBinding and ClusterRoleBinding: Συνδέουν ένα Role ή ClusterRole με έναν χρήστη, ομάδα ή ServiceAccount. Τα RoleBindings χρησιμοποιούνται μέσα σε ένα Namespace, ενώ τα ClusterRoleBindings χρησιμοποιούνται σε επίπεδο cluster.

CustomResourceDefinition (CRD): Σας επιτρέπει να δημιουργήσετε τους δικούς σας προσαρμοσμένους πόρους και να επεκτείνετε το Kubernetes API. Τα CRDs σας επιτρέπουν να ορίζετε και να διαχειρίζεστε νέους τύπους αντικειμένων προσαρμοσμένων στις ανάγκες της εφαρμογής σας.

PriorityClass: Αναθέτει μια προτεραιότητα στα Pods, επηρεάζοντας τη σειρά με την οποία προγραμματίζονται ή απομακρύνονται. Τα Pods με υψηλότερη προτεραιότητα προγραμματίζονται πριν από εκείνα με χαμηλότερη προτεραιότητα και είναι λιγότερο πιθανό να απομακρυνθούν όταν οι πόροι είναι περιορισμένοι.

IngressClass: Ορίζει τον τύπο του Ingress controller που θα πρέπει να εφαρμόσει τους κανόνες που ορίζονται από τα Ingress resources.

Lease: Χρησιμοποιείται εσωτερικά για την εκλογή ηγέτη μεταξύ των στοιχείων στο Kubernetes. Τα Leases βοηθούν στον συντονισμό των δραστηριοτήτων σε κατανεμημένα συστήματα.[15]

Εργαλεία χειρισμού και επέκτασης του συστήματος Kubernetes

Helm Chart

Σε ένα σύγχρονο περιβάλλον ανάπτυξης και διαχείρισης εφαρμογών με χρήση Kubernetes, η πολυπλοκότητα του συστήματος αυξάνεται σημαντικά καθώς οι εφαρμογές και οι υπηρεσίες γίνονται ολοένα και πιο πολλές. Σε αυτό το πλαίσιο, το Helm αναδεικνύεται ως ένα κρίσιμο εργαλείο χειρισμού και επέκτασης, επιτρέποντας στους διαχειριστές να διαχειρίζονται αποτελεσματικά το κύκλο ζωής των εφαρμογών.

Βασικές Αρχές και Δομή του Helm Chart:

Το **Helm** λειτουργεί ως ο διαχειριστής πακέτων για το Kubernetes, παρέχοντας ένα σύστημα ομαδοποίησης των πόρων μέσω των *Charts*. Ένα Helm Chart είναι ουσιαστικά μια συλλογή αρχείων που περιέχουν όλα τα απαραίτητα metadata και προδιαγραφές για τη δημιουργία,

διαχείριση και ενημέρωση ενός συνόλου σχετικών πόρων Kubernetes. Η βασική δομή ενός Helm Chart περιλαμβάνει:

- **Chart.yaml:** Το αρχείο αυτό περιέχει βασικές πληροφορίες για το chart, όπως το όνομα, την έκδοση, και μια σύντομη περιγραφή της εφαρμογής ή της υπηρεσίας που περιγράφεται.
- **values.yaml:** Σε αυτό το αρχείο ορίζονται οι παραμετροποιήσιμες τιμές που χρησιμοποιούνται στα πρότυπα (templates). Η δυνατότητα αυτή επιτρέπει την εύκολη προσαρμογή των παραμέτρων της εφαρμογής χωρίς την ανάγκη τροποποίησης των πηγαίων αρχείων.
- **Templates:** Ο κατάλογος αυτός περιέχει τα πρότυπα που, με τη βοήθεια του εργαλείου templating, μετατρέπονται σε τελικά αρχεία ρυθμίσεων (manifests) για το Kubernetes. Μέσω των templates, μπορούν να δημιουργηθούν δυναμικά οι πόροι που απαιτούνται για την εφαρμογή.
- **Dependencies:** Σε ορισμένα charts, υπάρχει η δυνατότητα αναφοράς σε εξωτερικά charts, διευκολύνοντας έτσι τη διαχείριση πολύπλοκων εφαρμογών που αποτελείται από πολλαπλές διασυνδεδεμένες υπηρεσίες.[16]

Πλεονεκτήματα και δυνατότητες του Helm:

Η υιοθέτηση του Helm ως εργαλείο διαχείρισης του Kubernetes προσφέρει πληθώρα πλεονεκτημάτων, τα οποία συμβάλλουν στην αποτελεσματική ανάπτυξη και συντήρηση εφαρμογών:

- **Απλοποίηση της Διαχείρισης Εφαρμογών:** Μέσω του Helm, η εγκατάσταση, αναβάθμιση και διαχείριση των εφαρμογών γίνεται με μία απλή εντολή. Αυτό μειώνει τις πιθανότητες ανθρώπινου λάθους και επιταχύνει τις διαδικασίες ανάπτυξης.
- **Έλεγχος Εκδόσεων και Rollback:** Το Helm επιτρέπει την παρακολούθηση και διαχείριση των εκδόσεων των εφαρμογών. Σε περίπτωση που μια αναβάθμιση παρουσιάσει προβλήματα, είναι εύκολη η επιστροφή σε μια προηγούμενη, σταθερή έκδοση.
- **Διαχείριση Εξαρτήσεων:** Μέσω της δυνατότητας καθορισμού εξαρτήσεων μεταξύ charts, μπορεί να διαχειριστεί κανείς πολύπλοκα οικοσυστήματα εφαρμογών, όπου κάθε υπηρεσία μπορεί να συνδέεται με άλλες.
- **Αυτοματισμός και Ενοποιημένες Διαδικασίες:** Η ενσωμάτωση του Helm στα CI/CD pipelines επιτρέπει την αυτόματη διαχείριση αλλαγών, αναβαθμίσεων και αναπροσαρμογών, συμβάλλοντας στη σταθερότητα και την αξιοπιστία του συστήματος.
- **Ευελιξία και Επεκτασιμότητα:** Η δυνατότητα δημιουργίας προσαρμοσμένων charts επιτρέπει την κάλυψη εξειδικευμένων αναγκών του περιβάλλοντος ανάπτυξης, καθιστώντας το Helm ένα εξαιρετικά ευέλικτο εργαλείο.

Εφαρμογές σε Περιβάλλοντα Παραγωγής:

Σε περιβάλλοντα όπου η αξιοπιστία και η ταχύτητα ανάπτυξης είναι κρίσιμες, όπως στα cloud-native συστήματα, το Helm αποτελεί βασικό εργαλείο. Οι διαχειριστές συστημάτων μπορούν να επωφεληθούν από:

- **Ελαχιστοποίηση του Κόστους και του Χρόνου Ανάπτυξης:** Με την επαναχρησιμοποίηση έτοιμων charts που προσφέρονται από την ενεργή κοινότητα, μειώνεται σημαντικά ο χρόνος που απαιτείται για την ανάπτυξη νέων εφαρμογών.
- **Ιστορικό εκδόσεων:** Η δυνατότητα rollback και η διαχείριση εκδόσεων επιτρέπουν την άμεση αντιμετώπιση προβλημάτων που μπορεί να προκύψουν κατά τη διάρκεια λειτουργίας σε παραγωγικό περιβάλλον.

Kubernetes Operators

Οι **Operators** είναι μια μέθοδος επέκτασης των δυνατοτήτων του Kubernetes για τη διαχείριση σύνθετων εφαρμογών. Ενσωματώνουν εξειδικευμένη γνώση τομέα στο API του Kubernetes, αυτοματοποιώντας όχι μόνο την ανάπτυξη, αλλά και τη συνολική διαχείριση του κύκλου ζωής των εφαρμογών.

Το Πρότυπο των Operators:

Το πρότυπο των operators αξιοποιεί την επεκτασιμότητα του Kubernetes μέσω των **Custom Resource Definitions (CRDs)** και των προσαρμοσμένων controllers. Τα CRDs σας επιτρέπουν να ορίσετε νέους τύπους πόρων που επεκτείνουν το API του Kubernetes, αντιπροσωπεύοντας την επιθυμητή κατάσταση ενός προσαρμοσμένου πόρου.

Ένας **Controller** παρακολουθεί αυτούς τους προσαρμοσμένους πόρους και λαμβάνει δράσεις για να συμβιβάσει την τρέχουσα κατάσταση με την επιθυμητή κατάσταση. Αυτή η διαδικασία είναι γνωστή ως βρόχος συμφιλίωσης (reconciliation loop), μια βασική έννοια στο control plane του Kubernetes.

Πώς Λειτουργούν οι Operators

1. Οι Operators ξεκινούν με τον ορισμό των CRDs που αντιπροσωπεύουν την επιθυμητή κατάσταση της εφαρμογής. Για παράδειγμα, ένα CRD μπορεί να ορίζει έναν πόρο MySQLCluster με προδιαγραφές για τον αριθμό των αντιγράφων, τις ρυθμίσεις αποθήκευσης και τις πολιτικές δημιουργίας αντιγράφων ασφαλείας.
2. Ο controller του operator παρακολουθεί συνεχώς τους προσαρμοσμένους πόρους για οποιεσδήποτε αλλαγές ή ενημερώσεις.
3. *Consolidation step*, όταν εντοπίζεται μια αλλαγή, ο controller εκτελεί τις απαραίτητες ενέργειες για να εναρμονίσει την πραγματική κατάσταση του cluster με την επιθυμητή κατάσταση που ορίζεται στο CRD. Αυτό μπορεί να περιλαμβάνει τη δημιουργία ή τη διαγραφή pods, την ενημέρωση ρυθμίσεων ή την έναρξη αντιγράφων ασφαλείας.

Η έκδοση MicroK8s

Το MicroK8s είναι μία ελαφριά, ολοκληρωμένη και πιστοποιημένη από το CNCF (Cloud Native Computing Foundation) διανομή του Kubernetes, που αναπτύσσεται και συντηρείται από την Canonical (την εταιρεία πίσω από το Ubuntu). Αποτελεί μία λύση που επιτρέπει την ταχεία ανάπτυξη και εκτέλεση ενός cluster Kubernetes σε οποιαδήποτε περιβάλλοντα, από τοπικές μηχανές ανάπτυξης μέχρι servers ή cloud instances, χωρίς πολύπλοκες διαδικασίες εγκατάστασης και ρυθμίσεων. Αποτελεί μια ευέλικτη λύση, ή όπως λέγεται μια batteries included διανομή Kubernetes που απευθύνεται τόσο σε προγραμματιστές όσο και σε διαχειριστές συστημάτων. Ταυτόχρονα, όμως, δεν αποκλείεται η αξιοποίησή του σε παραγωγικές εγκαταστάσεις, ιδίως όπου απαιτείται γρήγορη ανάπτυξη μικρότερων clusters λόγω των δυνατοτήτων όπως η υποστήριξη πολλαπλών κόμβων και το ευρύ φάσμα επεκτάσιμων λειτουργιών του (addons)

Συλλογή Δεδομένων από απομακρυσμένες συσκευές

Εισαγωγή

Οι απομακρυσμένες συσκευές, γνωστές και ως συσκευές "edge", αποτελούν τα σημεία επαφής μεταξύ του φυσικού κόσμου και των ψηφιακών συστημάτων επεξεργασίας δεδομένων. Βρίσκονται στην περιφέρεια του δικτύου και διαδραματίζουν κρίσιμο ρόλο στη συλλογή, την αρχική επεξεργασία και τη μετάδοση δεδομένων προς τα κεντρικά συστήματα. Τα χαρακτηριστικά τους ποικίλλουν, ωστόσο παρουσιάζουν κοινά στοιχεία που τις καθιστούν ιδιαίτερα σημαντικές στο πλαίσιο των δικτύων IoT (Internet of Things).

Ένα βασικό χαρακτηριστικό των απομακρυσμένων συσκευών είναι το ενεργειακό τους προφίλ. Πολλές από αυτές λειτουργούν με χαμηλή κατανάλωση ενέργειας, γεγονός που επιτρέπει την παρατεταμένη λειτουργία τους χωρίς συχνή φόρτιση ή αντικατάσταση μπαταριών. Αυτό είναι ιδιαίτερα σημαντικό για συσκευές που βρίσκονται σε απομακρυσμένες ή δυσπρόσιτες περιοχές. Άλλες συσκευές μπορεί να παρουσιάζουν κανονική κατανάλωση ενέργειας, ειδικά όταν απαιτείται υψηλότερη επεξεργαστική ισχύς ή συνεχής συνδεσιμότητα με το δίκτυο.

Επιπλέον, αυτές οι συσκευές συχνά χαρακτηρίζονται από τη χρήση απλών και ελαφριών πρωτοκόλλων επικοινωνίας λόγω των περιορισμένων υπολογιστικών και ενεργειακών πόρων τους. Η χρήση σύνθετων πρωτοκόλλων θα ήταν αντιπαραγωγική και θα μπορούσε να επηρεάσει αρνητικά την απόδοσή τους. Συνεπώς, προτιμώνται πρωτόκολλα που απαιτούν ελάχιστο εύρος ζώνης και χαμηλή κατανάλωση ενέργειας.

Ένα ακόμη χαρακτηριστικό είναι η λειτουργία τους σε περιβάλλοντα με περιορισμένη ή ασταθή συνδεσιμότητα. Οι απομακρυσμένες συσκευές πρέπει να είναι ανθεκτικές σε διακοπές δικτύου και να μπορούν να αποθηκεύουν ή να επεξεργάζονται δεδομένα τοπικά μέχρι την αποκατάσταση της σύνδεσης. Αυτό απαιτεί σχεδιασμό που λαμβάνει υπόψη τις ιδιαιτερότητες των δικτύων στα οποία λειτουργούν.

Η ικανότητά τους να συλλέγουν και να μεταδίδουν δεδομένα σε πραγματικό χρόνο τις καθιστά απαραίτητες για μια πληθώρα εφαρμογών, συμπεριλαμβανομένης της ενεργειακής διαχείρισης.

Εφαρμογές στην Ενεργειακή Κατανομή Φορτίων

Στον τομέα της ενεργειακής κατανομής φορτίων, η συλλογή ακριβών δεδομένων κατανάλωσης είναι ζωτικής σημασίας για την ανάλυση και την βελτιστοποίηση της χρήσης ενέργειας. Οι απομακρυσμένες συσκευές που χρησιμοποιούνται για τη μέτρηση της κατανάλωσης ενέργειας πρέπει να σχεδιάζονται με γνώμονα τα προαναφερθέντα χαρακτηριστικά, ώστε να εξασφαλίζεται η αποδοτική και αξιόπιστη λειτουργία τους.

Ένα κρίσιμο ζήτημα που ανακύπτει είναι ο λόγος για τον οποίο δεν εγκαθίστανται αισθητήρες μέτρησης σε κάθε μεμονωμένη συσκευή. Η απόφαση αυτή βασίζεται σε πολλούς παράγοντες:

Οικονομικό Κόστος: Η εγκατάσταση αισθητήρων σε κάθε συσκευή είναι οικονομικά ασύμφορη. Το κόστος αγοράς, εγκατάστασης και συντήρησης πολλαπλών αισθητήρων μπορεί

να είναι ιδιαίτερα υψηλό, ειδικά σε περιβάλλοντα με μεγάλο αριθμό συσκευών, όπως βιομηχανικές εγκαταστάσεις ή μεγάλα κτίρια γραφείων.

Πολυπλοκότητα Συστήματος: Η διαχείριση ενός δικτύου με πλήθος αισθητήρων αυξάνει σημαντικά την πολυπλοκότητα του συστήματος. Απαιτούνται προηγμένα πρωτόκολλα για τον συγχρονισμό και την επικοινωνία των αισθητήρων, καθώς και ισχυρά συστήματα επεξεργασίας για την ανάλυση του μεγάλου όγκου δεδομένων που παράγονται.

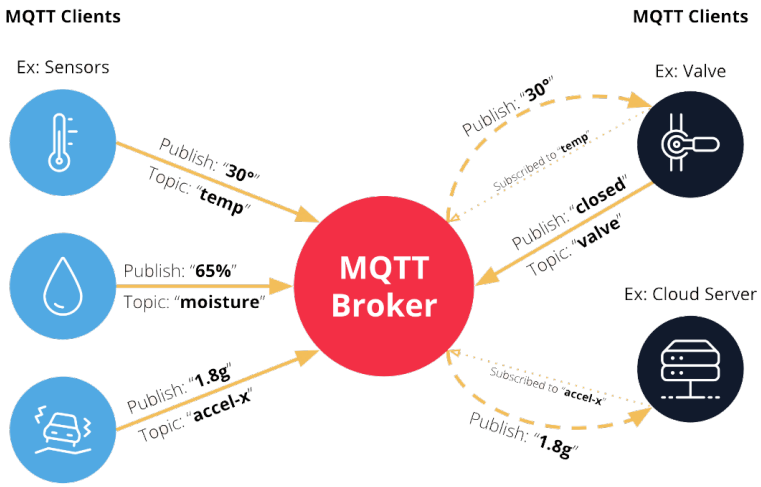
Περιορισμοί Συνδεσιμότητας: Πολλές συσκευές μπορεί να βρίσκονται σε περιοχές με περιορισμένη συνδεσιμότητα ή σε περιβάλλοντα που δεν υποστηρίζουν αξιόπιστη ασύρματη επικοινωνία. Αυτό καθιστά δύσκολη τη συνεχή και αξιόπιστη μετάδοση δεδομένων από κάθε μεμονωμένη συσκευή προς το κεντρικό σύστημα.

Ενεργειακή Απόδοση: Η προσθήκη αισθητήρων σε κάθε συσκευή μπορεί να αυξήσει την κατανάλωση ενέργειας, κάτι που είναι ανεπιθύμητο, ειδικά για συσκευές που έχουν σχεδιαστεί για να είναι ενεργειακά αποδοτικές. Επιπλέον, οι αισθητήρες απαιτούν συντήρηση και πιθανή αντικατάσταση μπαταριών, προσθέτοντας περαιτέρω κόστη και επιβαρύνσεις.

Λαμβάνοντας υπόψη τους παραπάνω παράγοντες, προκρίνεται η χρήση κεντρικών μετρητών για τη συλλογή δεδομένων κατανάλωσης. Οι κεντρικοί μετρητές εγκαθίστανται σε στρατηγικά σημεία του δικτύου ηλεκτρικής ενέργειας και μετρούν τη συνολική κατανάλωση μιας ομάδας συσκευών ή ενός ολόκληρου κτιρίου. Αυτή η προσέγγιση μειώνει το κόστος εγκατάστασης και συντήρησης, ενώ παράλληλα απλοποιεί την αρχιτεκτονική του συστήματος.[17]

Το MQTT ως επικρατέστερο πρωτόκολλο επικοινωνίας

Για την αποτελεσματική μετάδοση των δεδομένων που συλλέγονται από τους κεντρικούς μετρητές προς τα συστήματα επεξεργασίας, απαιτείται ένα πρωτόκολλο επικοινωνίας που να είναι αποδοτικό, αξιόπιστο και συμβατό με τους περιορισμούς των απομακρυσμένων συσκευών. Το MQTT (Message Queuing Telemetry Transport) έχει αναδειχθεί ως η πλέον κατάλληλη επιλογή για αυτόν τον σκοπό.



Εικόνα 5: Αναπαράσταση επικοινωνίας Broker-IoT συσκευών

Το MQTT είναι ένα πρωτόκολλο που έχει σχεδιαστεί ειδικά για εφαρμογές όπου η δικτυακή συνδεσιμότητα είναι περιορισμένη ή ασταθής. Η βασική δομή του πρωτοκόλλου περιλαμβάνει τα εξής κύρια στοιχεία:

- **Broker**: Αναλαμβάνει τη διαχείριση των μηνυμάτων και την προώθησή τους από τους εκδότες (publishers) προς τους συνδρομητές (subscribers).
- **Client (Publisher/Subscriber)**: Κάθε πελάτης λειτουργεί ως κόμβος που μπορεί να εκδίδει (publish) και/ή να εγγράφεται (subscribe) σε συγκεκριμένα "topics".

Πλεονεκτήματα του MQTT

Ελαφρύ και Αποδοτικό:

Το MQTT απαιτεί ελάχιστο εύρος ζώνης και χαμηλή υπολογιστική ισχύ, καθιστώντας το ιδανικό για συσκευές με περιορισμένους πόρους. Η απλότητα του πρωτοκόλλου συμβάλλει στη μείωση της κατανάλωσης ενέργειας, παρατείνοντας τη διάρκεια ζωής των συσκευών και μειώνοντας το συνολικό κόστος λειτουργίας.

Ανθεκτικότητα σε Ασταθείς Συνδέσεις:

Το πρωτόκολλο έχει σχεδιαστεί για να λειτουργεί αξιόπιστα σε περιβάλλοντα με ασταθή ή περιορισμένη συνδεσιμότητα. Υποστηρίζει μηχανισμούς επανασύνδεσης και διασφαλίζει ότι τα μηνύματα θα παραδοθούν όταν η σύνδεση αποκατασταθεί.

Μοντέλο Εκδότη/Συνδρομητή:

Το μοντέλο αυτό επιτρέπει την αποσύνδεση των αποστολέων και των παραληπτών των μηνυμάτων, παρέχοντας ευελιξία και απλοποιώντας την αρχιτεκτονική του συστήματος. Οι συσκευές μπορούν να δημοσιεύουν δεδομένα σε συγκεκριμένα θέματα (topics) και οι ενδιαφερόμενοι μπορούν να εγγράφονται σε αυτά τα θέματα για να λαμβάνουν τα σχετικά μηνύματα.

Υποστήριξη Ασφάλειας:

Το MQTT παρέχει δυνατότητες για την ενσωμάτωση πρωτοκόλλων ασφαλείας, όπως το SSL/TLS, διασφαλίζοντας την ασφαλή μετάδοση των δεδομένων. Αυτό είναι ιδιαίτερα σημαντικό σε εφαρμογές όπου η προστασία των δεδομένων είναι κρίσιμη.

Ευρεία αποδοχή:

Χιλιάδες οργανισμοί και εταιρείες τον χρησιμοποιούν για ένα ευρύ φάσμα εφαρμογών. Στο πλαίσιο της ενεργειακής κατανομής φορτίων, το MQTT διευκολύνει την αποτελεσματική και αξιόπιστη μετάδοση των δεδομένων κατανάλωσης από τους κεντρικούς μετρητές προς τα κεντρικά συστήματα ανάλυσης. Η δυνατότητα μετάδοσης σε πραγματικό χρόνο επιτρέπει την άμεση επεξεργασία των δεδομένων και την εφαρμογή αλγορίθμων μηχανικής μάθησης για την ανάλυση και την πρόβλεψη της κατανάλωσης. Αυτό συμβάλλει στη βελτίωση της ενεργειακής αποδοτικότητας και στην αποτελεσματικότερη διαχείριση των πόρων.[18]

Η τεχνολογία EMQX

Το EMQX είναι ένας ανοιχτού κώδικα MQTT Broker υψηλής απόδοσης, γραμμένος κυρίως σε Erlang/OTP. Η χρήση της γλώσσας Erlang, η οποία έχει σχεδιαστεί για την ανάπτυξη κατανεμημένων και παράλληλων συστημάτων, προσδίδει στο EMQX ιδιότητες όπως:

- Σταθερότητα σε περιβάλλοντα υψηλού φόρτου.
- Δυνατότητα οριζόντιας κλιμάκωσης (horizontal scaling).
- Υψηλή ανοχή σε σφάλματα (fault tolerance).

Από την πλευρά της αγοράς, το EMQX υποστηρίζει τα πρωτόκολλα MQTT, MQTT-SN, CoAP, LwM2M, καθώς και WebSocket για διασύνδεση με εφαρμογές web. Επιπλέον, προσφέρει προηγμένους μηχανισμούς ασφαλείας, όπως έλεγχο ταυτότητας (authentication) και εξουσιοδότηση (authorization), χρησιμοποιώντας πολλαπλούς τύπους αποθετηρίων (backends), όπως βάσεις δεδομένων SQL/NoSQL, LDAP κ.ά.

Αρχιτεκτονική και Χαρακτηριστικά

Η αρχιτεκτονική του EMQX ακολουθεί την κατανεμημένη φιλοσοφία της Erlang/OTP, επιτρέποντάς του να δημιουργεί συστοιχίες (clusters) που μπορούν να επεξεργάζονται εκατομμύρια μηνύματα ανά δευτερόλεπτο. Μερικά βασικά χαρακτηριστικά είναι:

1. **Υποστήριξη Πολλαπλών Πρωτοκόλλων**: Το EMQX μπορεί να λειτουργήσει ως κεντρικός κόμβος (hub) για διαφορετικά πρωτόκολλα ανταλλαγής μηνυμάτων. Εκτός από το MQTT, υποστηρίζει HTTP, WebSockets, STOMP και άλλα, προσφέροντας ενιαία πλατφόρμα για εταιρικές και βιομηχανικές εφαρμογές.
2. **Υψηλή Επεκτασιμότητα (Scalability)**: Η δυνατότητα clustering επιτρέπει τη διανομή φόρτου σε πολλούς κόμβους. Έτσι, όταν αυξάνονται οι απαιτήσεις σε ρυθμό μετάδοσης δεδομένων (throughput) ή σε αριθμό συνδέσεων πελατών, προστίθενται επιπλέον κόμβοι για να διαμοιράσουν τον φόρτο.
3. **Ανοχή σε Σφάλματα (Fault Tolerance)**: Η Erlang/OTP διαθέτει μηχανισμούς επιτήρησης (supervision trees) και απομόνωσης διεργασιών (isolated processes), ώστε ένα σφάλμα σε μία διεργασία να μην επηρεάσει το υπόλοιπο σύστημα. Το EMQX εκμεταλλεύεται στο έπακρο αυτά τα χαρακτηριστικά, επιτυγχάνοντας αξιόπιστη λειτουργία 24/7.
4. **Συστήματα Authentication & Authorization**: Προσφέρει ενσωματωμένες επιλογές αυθεντικοποίησης (JWT, LDAP κ.λπ.) και εξουσιοδότησης (ACLs, βάση ρόλων), διευκολύνοντας την ασφαλή ενσωμάτωση με εταιρικά περιβάλλοντα.
5. **Πολιτικές Διαχείρισης Μηνυμάτων (Message Retention & QoS)**: Παρέχει λεπτομερή ρύθμιση για τη διατήρηση μηνυμάτων (retained messages) και υποστηρίζει τα τρία επίπεδα QoS (Quality of Service) του πρωτοκόλλου MQTT (QoS 0, QoS 1, QoS 2), καλύπτοντας κάθε ανάγκη αξιοπιστίας στην παράδοση μηνυμάτων.

6. **Plugins και API**: Η αρχιτεκτονική του EMQX είναι επεκτάσιμη μέσω plugins, επιτρέποντας τη δημιουργία ή την ενσωμάτωση προσαρμοσμένων λειτουργιών. Επιπλέον, τα REST APIs του Broker διευκολύνουν την παρακολούθηση και τον έλεγχο σε πραγματικό χρόνο.

Σύστημα Κανόνων (Rules Engine) και Ενσωμάτωση με Εξωτερικές Υπηρεσίες

Ένα από τα πιο καινοτόμα χαρακτηριστικά του EMQX είναι το σύστημα κανόνων (Rules Engine), το οποίο:

- Διευκολύνει τον ορισμό λογικών κανόνων σε πραγματικό χρόνο, ώστε να γίνεται φιλτράρισμα, επεξεργασία και προώθηση των μηνυμάτων.
- Επιτρέπει τη δυναμική μετατροπή και εμπλουτισμό των δεδομένων (data enrichment) μέσω SQL-like συντακτικού.
- Παρέχει συνδέσεις (bridges ή connectors) προς εξωτερικές υπηρεσίες cloud (AWS, Azure, Google Cloud), βάσεις δεδομένων (MySQL, PostgreSQL, MongoDB) και message queues (RabbitMQ, Kafka). Έτσι, ο Broker μπορεί να διαδραματίσει κεντρικό ρόλο σε ετερογενή περιβάλλοντα IoT ή βιομηχανικού αυτοματισμού, ως κόμβος ενοποίησης πολλαπλών δικτύων και υποσυστημάτων.

Κλιμάκωση και Επιπλέον Τεχνικές Ενίσχυσης Απόδοσης

Η ικανότητα κλιμάκωσης (scalability) του EMQX αποτελεί αποφασιστικό πλεονέκτημα για μεγάλα έργα IoT. Συγκεκριμένα:

- **Οριζόντια Κλιμάκωση (Horizontal Scaling)**: Με την προσθήκη επιπλέον κόμβων σε ένα cluster, αυξάνεται τόσο η χωρητικότητα σε ταυτόχρονες συνδέσεις, όσο και η δυνατότητα επεξεργασίας όγκου μηνυμάτων.
- **Κάθετη Κλιμάκωση (Vertical Scaling)**: Η αξιοποίηση περισσότερων πόρων σε επίπεδο hardware (επεξεργαστές, μνήμη) μπορεί να ενισχύσει την απόδοση ενός μεμονωμένου κόμβου, ειδικά όταν αυτός βρίσκεται σε περιβάλλον cloud.
- **Caching**: Το EMQX ενσωματώνει μηχανισμούς caching για τις πληροφορίες πιστοποίησης και εξουσιοδότησης, μειώνοντας την επίδραση των καθυστερήσεων που σχετίζονται με εξωτερικές βάσεις δεδομένων.
- **Αποσύνδεση Λειτουργιών (Offloading)**: Μέσω plugins ή εξωτερικών υπηρεσιών, κάποιες λειτουργίες επεξεργασίας δεδομένων μπορούν να "μεταφέρονται" σε άλλους κόμβους ή σε streaming πλατφόρμες (όπως το Apache Kafka), διατηρώντας τον Broker ελαφρύτερο για την καθαυτή μετάδοση μηνυμάτων.

Σύγκριση EMQX με άλλους MQTT Brokers

Υπάρχουν πολλοί MQTT Brokers στην αγορά. Μερικοί δημοφιλείς είναι οι Mosquitto (Eclipse Foundation), HiveMQ και VerneMQ. Μια συγκριτική θεώρηση με επίκεντρο το EMQX μπορεί να περιλαμβάνει τα εξής σημεία:

- **Απόδοση**: Το EMQX, χάρη στην Erlang/OTP, παρουσιάζει υψηλούς ρυθμούς επεξεργασίας μηνυμάτων και εντυπωσιακή σταθερότητα υπό φόρτο.
- **Επεκτασιμότητα & Clustering**: Σε αντίθεση με το Mosquitto, που δεν διαθέτει εγγενή αρχιτεκτονική clustering, το EMQX υποστηρίζει αξιόπιστη κλιμάκωση σε επίπεδο πολλαπλών κόμβων. Το VerneMQ επίσης βασίζεται σε Erlang και διαθέτει clustering, ωστόσο το EMQX τείνει να προσφέρει περισσότερες επιλογές σύνδεσης με τρίτα συστήματα (through connectors & plugins).
- **Διαχειριστικές Δυνατότητες**: Μέσω του ολοκληρωμένου πίνακα ελέγχου (dashboard), το EMQX προσφέρει ευκολία παρακολούθησης (monitoring) και ελέγχου. Ο HiveMQ, παρότι διαθέτει επίσης εξελιγμένο dashboard, απαιτεί συχνά επαγγελματική έκδοση για προηγμένες λειτουργίες.
- **Σύστημα Κανόνων**: Το ενσωματωμένο Rules Engine του EMQX αποτελεί ένα ιδιαίτερο πλεονέκτημα που το διαφοροποιεί. Δίνει τη δυνατότητα για "πλούσια" λογική επεξεργασία των δεδομένων χωρίς ανάγκη για πρόσθετα εργαλεία.

Συνολικά, το EMQX θεωρείται μία από τις κορυφαίες επιλογές MQTT Broker, ιδίως για περιβάλλοντα με απαιτήσεις υψηλής απόδοσης και πολυπλοκότητας διασύνδεσης.

Ασφάλεια και Πιστοποίηση

Στις σύγχρονες IoT εφαρμογές, η ασφάλεια αποκτά κεντρικό ρόλο. Το EMQX προσφέρει πολλαπλά επίπεδα ασφαλείας:

- **TLS/SSL**: Για κρυπτογραφημένη επικοινωνία με τους πελάτες.
- **Authentication**: Ορίζει πολλούς μηχανισμούς για την πιστοποίηση των συσκευών/χρηστών (basic, JWT, LDAP, OAuth κ.ά.).
- **Authorization & ACLs**: Επιτρέπει την προσαρμοσμένη διαχείριση κανόνων πρόσβασης σε "topics" ανάλογα με ομάδες, ρόλους ή μεμονωμένους χρήστες.
- **Audit Logs**: Καταγραφή γεγονότων και αλληλεπιδράσεων, που βοηθά στην ανάλυση και στην τήρηση κανονισμών (compliance).

EMQX & K8s

Η ανάγκη για κλιμακούμενες, ανθεκτικές και δυναμικά διαχειρίσιμες υποδομές έχει καθιερώσει το Kubernetes ως την επικρατέστερη επιλογή ενορχήστρωσης container σε

σύγχρονα περιβάλλοντα cloud. Παράλληλα, η εξάπλωση των εφαρμογών IoT και η αυξανόμενη απαίτηση για υποστήριξη τεράστιων όγκων μηνυμάτων οδήγησαν στην ανάπτυξη υψηλών επιδόσεων MQTT Brokers, όπως το EMQX. Ο συνδυασμός EMQX και Kubernetes είναι ιδανικός για εφαρμογές που απαιτούν:

1. **Αυτοματοποιημένη κλιμάκωση (auto-scaling)** όταν αυξάνεται ο φόρτος ανταλλαγής δεδομένων.
2. **Υψηλή διαθεσιμότητα** και δυνατότητα ανάκαμψης σε περίπτωση σφαλμάτων (fault tolerance).
3. **Εύκολη διανομή φορτίου** (load balancing) σε πολλούς κόμβους EMQX, με ταυτόχρονη διαχείριση του clustering.

EMQX Operator

Το EMQX Kubernetes Operator είναι ένα εργαλείο που αναπτύχθηκε με σκοπό να απλοποιήσει και να αυτοματοποιήσει τη διαχείριση του MQTT Broker EMQX σε περιβάλλοντα Kubernetes. Αξιοποιεί την αρχιτεκτονική και τις πρακτικές των Operators, δηλαδή εξειδικευμένων ελεγκτών που μπορούν να εκφράζουν την "επιχειρησιακή γνώση" (operational knowledge) μιας εφαρμογής, ώστε να εξασφαλίζεται κλιμακούμενη, σταθερή και εύκολα διαχειρίσιμη εγκατάσταση του EMQX.

Ο **EMQX Kubernetes Operator** έρχεται να καλύψει την ανάγκη για αυτοματοποιημένη και αξιόπιστη διαχείριση ενός EMQX cluster σε Kubernetes. Συγκεκριμένα:

Δημιουργία & Παραμετροποίηση

Ο Operator εισάγει ένα ή περισσότερα CRDs (Custom Resource Definitions) , μέσω των οποίων ο χρήστης μπορεί να παραμετροποιήσει και να ορίσει:

- Το μέγεθος του cluster (πόσοι κόμβοι θα τρέχουν).
- Την επιθυμητή έκδοση του EMQX ή τις ρυθμίσεις ενημέρωσης.
- Παραμέτρους αυτόματης κλιμάκωσης (auto-scaling).
- Ρυθμίσεις ασφάλειας (πιστοποιητικά TLS, κανόνες ACL).

Εκτέλεση Λειτουργιών σε Cluster

- Μόλις ο Operator διαπιστώσει ότι δημιουργήθηκε ή άλλαξε ένας πόρος EMQX, ξεκινά μια διαδικασία ελέγχου (reconciliation) για να φέρει τη "πραγματική κατάσταση" (actual state) του EMQX στα pods, σε συμφωνία με την "επιθυμητή κατάσταση" (desired state) που καθορίζεται στο YAML.
- Υποστηρίζει **rolling updates**, επιτρέποντας σταδιακή αναβάθμιση των κόμβων χωρίς διακοπή της υπηρεσίας.

Παρακολούθηση & Επικύρωση (Monitoring & Validation)

- Ο Operator επικυρώνει ότι η εγκατάσταση του EMQX είναι συνεπής με τις προδιαγραφές (το επιθυμητό QoS επίπεδο, τα plugins που πρέπει να είναι ενεργοποιημένα).
- Παρέχει μηχανισμούς για παρατηρησιμότητας (observability), με ενδείξεις για την κατάσταση των pods, την κίνηση των μηνυμάτων, πιθανές αστοχίες κτλ.

Κύρια Οφέλη της Χρήσης του Operator

1. **Αυτοματοποιημένη Κλιμάκωση (Auto-Scaling)**

   Μέσω της συνεργασίας με τον **Horizontal Pod Autoscaler (HPA)** του Kubernetes, ο EMQX Operator μπορεί να αυξομειώνει αυτόματα τον αριθμό των EMQX pods, ανάλογα με το φορτίο σε πραγματικό χρόνο (αν ο αριθμός των συνδέσεων MQTT ή η χρήση πόρων αυξηθεί).

2. **Rolling Updates Χωρίς Διακοπή**

   Η διαδικασία ενημέρωσης (upgrade) του EMQX γίνεται σταδιακά: ο Operator "κατεβάζει" έναν κόμβο για αναβάθμιση, ελέγχει τη σταθερότητα του cluster, και συνεχίζει με τον επόμενο, διατηρώντας την υπηρεσία διαθέσιμη.

3. **Εύκολη Ανάπτυξη & Συντήρηση**

   Οι σύνθετες ρυθμίσεις (π.χ. διασυνδέσεις με βάσεις δεδομένων, πιστοποίηση χρηστών, ενεργοποίηση plugins) μπορούν να οριστούν μία φορά σε αρχεία YAML. Ο Operator εγγυάται τη συνεπή εφαρμογή τους σε κάθε κόμβο.

4. **Ενιαία Διαχειριστική Επιφάνεια**

   Όλη η παραμετροποίηση του EMQX βρίσκεται μέσα στο ίδιο οικοσύστημα του Kubernetes.

5. **Υψηλή Διαθεσιμότητα**

   Σε περίπτωση αστοχίας ενός κόμβου ή ενός ολόκληρου node του Kubernetes, ο Operator εντοπίζει το πρόβλημα και επανεκκινεί τις επιμέρους υπηρεσίες

Διαχείριση μεγάλων όγκων μηνυμάτων σε κεντρικά συστήματα

Εισαγωγή

Οι συμβατικές βάσεις δεδομένων σχεδιάστηκαν για τη διαχείριση δομημένων δεδομένων με σχετικά χαμηλούς ρυθμούς εισαγωγής και ανάκτησης. Η αρχιτεκτονική τους βασίζεται σε κεντρικοποιημένες δομές που δεν μπορούν εύκολα να κλιμακωθούν οριζόντια για να αντιμετωπίσουν τον αυξανόμενο όγκο και την ποικιλία των δεδομένων που παράγονται από σύγχρονες εφαρμογές.

Παρά τις προσπάθειες να αντιμετωπιστούν αυτοί οι περιορισμοί μέσω της ανάπτυξης νέων τεχνολογιών βάσεων δεδομένων, όπως οι κατανεμημένες NoSQL βάσεις δεδομένων ή οι βάσεις χρονοσειρών, εξακολουθούν να υπάρχουν σημαντικές προκλήσεις. Αυτές οι σύγχρονες βάσεις δεδομένων έχουν σχεδιαστεί για να προσφέρουν υψηλότερη απόδοση και κλιμάκωση, επιτρέποντας την οριζόντια επέκταση σε πολλαπλούς κόμβους και την αποτελεσματική διαχείριση μεγάλων όγκων δεδομένων. Βασίζονται σε αρχιτεκτονικές που αξιοποιούν την κατανεμημένη αποθήκευση και επεξεργασία, επιτρέποντας τη γρήγορη ανάκτηση και εισαγωγή δεδομένων.

Ωστόσο, παρά τις βελτιώσεις που προσφέρουν, αυτές οι τεχνολογίες δεν είναι πάντα κατάλληλες για την επεξεργασία δεδομένων σε πραγματικό χρόνο και τη διαχείριση συνεχών ροών δεδομένων μεγάλης κλίμακας. Η φύση τους ως αποθηκευτικά συστήματα, ακόμη και αν είναι βελτιστοποιημένα για υψηλή απόδοση, δεν ανταποκρίνεται πλήρως στις απαιτήσεις εφαρμογών που χρειάζονται άμεση επεξεργασία και μετάδοση δεδομένων με ελάχιστη καθυστέρηση.

Κατανεμημένα συστήματα αποθήκευσης μηνυμάτων.

Τα κατανεμημένα συστήματα αποθήκευσης μηνυμάτων επιτρέπουν την αποσύνδεση των εφαρμογών που παράγουν δεδομένα (producers) από εκείνες που τα καταναλώνουν (consumers). Με αυτόν τον τρόπο, οι διάφορες υπηρεσίες και υπό-συστήματα μπορούν να αναπτυχθούν, να εξελιχθούν και να συντηρηθούν ξεχωριστά, περιορίζοντας τις άμεσες εξαρτήσεις. Επίσης, παρέχουν δυνατότητα οριζόντιας κλιμάκωσης, ώστε να ανταποκρίνονται άμεσα σε αυξημένο φόρτο, ενώ χάρη στη διανομή των δεδομένων σε πολλαπλούς κόμβους, επιτυγχάνεται υψηλή ανθεκτικότητα (fault tolerance) και αξιοπιστία.

Μέσα σε αυτό το ευρύτερο πλαίσιο εντάσσεται και το Apache Kafka, ένα κατανεμημένο σύστημα αποθήκευσης μηνυμάτων που αποτελεί πλέον σημείο αναφοράς στον χώρο των streaming platforms. Χάρη στην αρχιτεκτονική του, το Kafka προσφέρει ταχύτατη και αξιόπιστη ανταλλαγή μηνυμάτων, η οποία συνδυάζεται με απρόσκοπτη επεκτασιμότητα. Η δυνατότητά του να επεξεργάζεται σημαντικές ποσότητες δεδομένων σε πραγματικό χρόνο το έχει καταστήσει ουσιώδες συστατικό σε περιβάλλοντα με μικροϋπηρεσίες (microservices), όπου η επικοινωνία μεταξύ αποσυνδεδεμένων υπηρεσιών βασίζεται όλο και περισσότερο σε γεγονότα (event-driven architecture).

Το σύστημα Apache Kafka

Το Apache Kafka αποτελεί μια κατανεμημένη πλατφόρμα ροής δεδομένων (distributed streaming platform), σχεδιασμένη να επιτρέπει την ταχύτατη και αξιόπιστη ανταλλαγή και επεξεργασία μηνυμάτων μεταξύ συστημάτων. Η βάση της λειτουργίας του έγκειται σε μια αρχιτεκτονική που εστιάζει στην ιδέα των *γεγονότων (events),* τα οποία αναπαρίστανται ως μηνύματα που αποστέλλονται από διάφορες πηγές και καταναλώνονται από ετερογενείς προορισμούς.

Στην καρδιά του Kafka βρίσκονται τα λεγόμενα «θέματα» (topics). Το κάθε θέμα οργανώνεται σε πολλαπλά τμήματα (partitions), ώστε να διασφαλίζεται υψηλός βαθμός παράλληλης επεξεργασίας και αποθήκευσης. Κάθε partition λειτουργεί ουσιαστικά σαν ένα κατανεμημένο αρχείο καταγραφής (log), στο οποίο τα μηνύματα εγγράφονται σειριακά και φέρουν ένα μοναδικό offset, προσφέροντας έτσι ξεκάθαρη δομή ιστορικού. Αυτή η προσέγγιση προσδίδει σταθερότητα στην πλατφόρμα, καθώς διευκολύνει τη διαχείριση μεγάλων ροών δεδομένων και επιτρέπει την εύκολη ανάκτηση των μηνυμάτων όταν χρειάζεται αναδρομική επεξεργασία ή επανεπεξεργασία.

Οι ρόλοι των οντοτήτων εντός του οικοσυστήματος του Kafka είναι ξεκάθαροι: οι παραγωγοί (producers) συνθέτουν και αποστέλλουν μηνύματα, οι καταναλωτές (consumers) τα λαμβάνουν, ενώ οι διαχειριστές του συστήματος, δηλαδή οι «μεσολαβητές» (Brokers), αναλαμβάνουν την αποθήκευση και τη διακίνηση των μηνυμάτων

Σημαντικό πυλώνα στη σχεδίαση του Kafka αποτελεί η αξιοπιστία μέσω αναπαραγωγής (replication). Τα partitions ενός θέματος αντιγράφονται σε πολλαπλούς κόμβους του Kafka, δημιουργώντας αντίγραφα που εξυπηρετούνται από ένα εκάστοτε «ηγέτη» (leader). Όταν ένας κόμβος που φιλοξενεί το αντίγραφο-leader καταστεί μη διαθέσιμος, ένα άλλο αντίγραφο αναλαμβάνει αυτόματα τον ρόλο του ηγέτη, γεγονός που διασφαλίζει την απρόσκοπτη λειτουργία του συνόλου. Αυτή η διαδικασία στηρίζεται παραδοσιακά στο Apache ZooKeeper, το οποίο συγχρονίζει τις πληροφορίες για την κατάσταση του cluster, αν και στα νεότερα releases προωθείται η μετάβαση σε μια πιο αυτόνομη ενσωμάτωση των υπηρεσιών συντονισμού, γνωστή και ως KIP-500 (Kafka without ZooKeeper).

Η κλιμάκωση (scalability) επιτυγχάνεται κυρίως με την ευελιξία στον διαμοιρασμό των δεδομένων σε πολλά partitions και με την προσθήκη νέων Brokers στο cluster. Έτσι, όταν ένα σύστημα αντιμετωπίζει αυξανόμενο φορτίο, μπορούμε να επεκτείνουμε το Kafka οριζόντια, προσθέτοντας επιπλέον κόμβους, και το ίδιο το cluster προσαρμόζεται ανακατανέμοντας τις ευθύνες αποθήκευσης και διακίνησης μηνυμάτων. Αυτή η ικανότητα καθιστά το Kafka εξαιρετικά ελκυστικό για συστήματα που χρειάζονται υψηλή απόδοση σε συνθήκες με πολλές ταυτόχρονες ροές και μεγάλους όγκους δεδομένων.

Ένα ακόμη χαρακτηριστικό που έχει προσδώσει στο Kafka τη φήμη του ως *backbone* για event-driven συστήματα είναι η υποστήριξη για σύνθετα μοτίβα επεξεργασίας ροής (stream processing). Εργαλεία όπως το Kafka Streams API ή και λύσεις τρίτων (Apache Flink, Apache Spark Streaming) αξιοποιούν την υποδομή του Kafka για να προσφέρουν σε πραγματικό χρόνο επεξεργασία, μετασχηματισμό και ανάλυση δεδομένων. Έτσι, η πλατφόρμα δεν περιορίζεται απλώς στη μεταφορά γεγονότων, αλλά επεκτείνεται σε μια ολοκληρωμένη ροή, όπου τα γεγονότα μπορούν να επεξεργαστούν ακαριαία και να τροφοδοτήσουν άλλα συστήματα με επεξεργασμένα, συμπυκνωμένα ή εμπλουτισμένα δεδομένα.

Επιπλέον, στο πεδίο της παράδοσης και του συγχρονισμού, το Kafka προσφέρει διαφορετικά επίπεδα *εγγυήσεων (delivery guarantees),* από *τουλάχιστον μία φορά (at-least-once)* έως *τουλάχιστον μία φορά με σπάνια διπλοτυπία* ή και *ακριβώς μία φορά (exactly-once)*, ανάλογα με τον τρόπο που έχουν ρυθμιστεί οι παραγωγοί, οι καταναλωτές και τα transactions. Αυτά τα χαρακτηριστικά επιτρέπουν να χτιστούν πολύπλοκες λύσεις, ακόμη και σε επιχειρησιακά σενάρια όπου η ακρίβεια των εγγραφών είναι καθοριστική.

Τέλος, ο σχεδιασμός του Kafka ευνοεί την ανεξαρτησία των συστημάτων που ανταλλάσσουν δεδομένα. Με το να παρεμβάλλεται ως ένας αρθρωτός ενδιάμεσος χώρος (buffer) μεταξύ εφαρμογών και υπηρεσιών, μειώνονται οι άμεσες εξαρτήσεις και η πολυπλοκότητα που προκύπτει από τις σημειακές διασυνδέσεις. Με αυτόν τον τρόπο, το Kafka διατηρεί μια κεντρική θέση σε αρχιτεκτονικά σχήματα που βασίζονται στη φιλοσοφία των μικροϋπηρεσιών (microservices). Κάθε υπηρεσία μπορεί να δημοσιεύει ή να καταναλώνει γεγονότα χωρίς να χρειάζεται να επικοινωνεί ευθέως με τις υπόλοιπες, συνθέτοντας έτσι ένα πιο ευέλικτο, επεκτάσιμο και ανθεκτικό οικοσύστημα.[19]

Kafka bridge

Το Kafka Bridge χρησιμοποιείται για να επιτρέπει την επικοινωνία μεταξύ συστημάτων που δεν υποστηρίζουν απευθείας το πρωτόκολλο επικοινωνίας του, όπως ο EMQX Broker, και του συστήματος Kafka. Η βασική λειτουργία του είναι να γεφυρώνει τη μετάδοση των μηνυμάτων από τον EMQX (ή άλλα συστήματα) προς το Kafka με τρόπο απλό και αποδοτικό.

Οι λόγοι για να χρησιμοποιήσουμε τον Kafka Bridge περιλαμβάνουν:

1. Ευκολία Ενσωμάτωσης: Το Kafka Bridge επιτρέπει στον EMQX Broker, που χρησιμοποιεί το πρωτόκολλο MQTT, να στείλει τα δεδομένα του στο Kafka χωρίς να χρειάζεται να μιλάει απευθείας το πρωτόκολλο του Kafka.
2. Υποστήριξη HTTP: Ο Kafka Bridge χρησιμοποιεί HTTP APIs, κάτι που διευκολύνει τη σύνδεση του EMQX (που υποστηρίζει HTTP) με το Kafka. Αυτό επιτρέπει τη μετάδοση μηνυμάτων μέσω ενός γνωστού και ευέλικτου πρωτοκόλλου όπως το HTTP.
3. Απλοποίηση Διαχείρισης: Η χρήση του Kafka Bridge απλοποιεί τη διαδικασία ενσωμάτωσης συστημάτων που έχουν διαφορετικά πρωτόκολλα επικοινωνίας, χωρίς να χρειάζεται να κάνουμε σημαντικές αλλαγές στη δομή του EMQX ή του Kafka.
4. Μεταφορά Δεδομένων σε Κλίμακα: Ο Kafka Bridge είναι σχεδιασμένος να μεταφέρει μεγάλες ποσότητες δεδομένων από συστήματα όπως ο EMQX στο Kafka με σταθερό και αξιόπιστο τρόπο, διασφαλίζοντας ότι η ροή δεδομένων είναι ομαλή και συνεχής.[20]

Apache Kafka & K8s

Η ενσωμάτωση του Kubernetes με πλατφόρμες διαχείρισης δεδομένων, όπως το Apache Kafka, προσφέρει σημαντικά πλεονεκτήματα στην ανάπτυξη και διαχείριση εφαρμογών μεγάλης κλίμακας. Το Kubernetes επιτρέπει την αυτοματοποίηση της ανάπτυξης των Kafka clusters, διευκολύνοντας την κλιμάκωση και τη διαχείριση των πόρων.

Με τη χρήση Deployments και StatefulSets, τα Kafka Brokers μπορούν να αναπτυχθούν και να διαχειριστούν αποτελεσματικά, εξασφαλίζοντας την ακεραιότητα των δεδομένων και τη διαθεσιμότητα των υπηρεσιών. Τα ConfigMaps και Secrets διευκολύνουν τη διαχείριση των ρυθμίσεων και των ευαίσθητων πληροφοριών που απαιτούνται για τη λειτουργία του Kafka, επιτρέποντας την ασφαλή και ευέλικτη διαχείριση του συστήματος.

Η χρήση του Kubernetes σε συνδυασμό με το Apache Kafka επιτρέπει την επεξεργασία μεγάλων ροών δεδομένων σε πραγματικό χρόνο. Αυτό είναι ιδιαίτερα σημαντικό στην ενεργειακή κατανομή φορτίων, όπου η ταχεία επεξεργασία και ανάλυση δεδομένων μπορεί να οδηγήσει σε βελτιώσεις στην απόδοση και την αποδοτικότητα των συστημάτων..

Strimzi Operator

Ο Strimzi είναι ένα open-source έργο που παρέχει Operators και εργαλεία για την ανάπτυξη και τη διαχείριση συμπλεγμάτων (clusters) του Apache Kafka μέσα σε ένα Kubernetes cluster. Στόχος του είναι να απλοποιήσει τη διαδικασία ρύθμισης, κλιμάκωσης, αναβάθμισης και παρακολούθησης της υποδομής Kafka, εκμεταλλευόμενο τις εγγενείς δυνατότητες του Kubernetes για ανθεκτικότητα, οριζόντια κλιμάκωση και αυτοματοποίηση.

Το έργο Strimzi εστιάζει σε δύο κύριες πλευρές:

1. Εφαρμογή της δηλωτικής (declarative) φιλοσοφίας του Kubernetes, όπου ο χρήστης ορίζει την επιθυμητή κατάσταση του Kafka cluster μέσω Custom Resource Definitions (CRDs).
2. Ανάπτυξη Operators που αναλαμβάνουν να συγκρίνουν την τρέχουσα κατάσταση του συστήματος με την επιθυμητή κατάσταση και να εκτελούν αυτόματα τις απαιτούμενες ενέργειες (π.χ. δημιουργία, κλιμάκωση, αναβάθμιση κόμβων, διαχείριση topics κ.λπ.).

Τα βασικά συστατικά του Strimzi

Custom Resource Definitions (CRDs)

Το Kubernetes επιτρέπει τη δημιουργία νέων ειδών πόρων (resources) μέσω των λεγόμενων Custom Resource Definitions. Ο Strimzi εισάγει συγκεκριμένα CRDs για το Kafka, όπως:

- Kafka: Δηλώνει πώς πρέπει να διαμορφωθεί ένα Kafka cluster (συμπεριλαμβανομένων των ρυθμίσεων για ZooKeeper, Kafka Brokers κ.λπ.).

- KafkaTopic: Επιτρέπει στον χρήστη να περιγράψει με δηλωτικό τρόπο (YAML spec) ένα ή περισσότερα Kafka topics.
- KafkaUser: Διαχειρίζεται χρήστες και δικαιώματα πρόσβασης (ACLs).
- KafkaConnect: Καθορίζει πώς να γίνει εγκατάσταση και διαχείριση του Kafka Connect στο Kubernetes.

Κάθε CRD περιλαμβάνει τα πεδία που χρειάζονται για τη ρύθμιση του cluster ή του αντίστοιχου πόρου (π.χ. αριθμός replicas των Brokers, παραμετροποίηση topics). Όταν ο χρήστης δημιουργεί ή τροποποιεί ένα CRD αρχείο YAML, το Kubernetes ενημερώνει τον Operator για να ευθυγραμμίσει την πραγματική κατάσταση του συστήματος με αυτήν που περιγράφεται στο CRD.

Operators

Ο Strimzi Operator αποτελεί το βασικό κομμάτι του συστήματος: παρακολουθεί τα CRDs και προβαίνει στις αναγκαίες αλλαγές, ώστε το Kafka cluster να διατηρείται στην επιθυμητή κατάσταση. Αυτοματοποιεί πολύπλοκες διαδικασίες όπως:

- Δημιουργία του ZooKeeper cluster που απαιτείται για τον συντονισμό του Kafka.
- Δημιουργία/ρύθμιση των Kafka Brokers με βάση τις προδιαγραφές (π.χ. επιθυμητό replication, persistence, παράμετροι για τη βελτιστοποίηση της απόδοσης).
- Rolling updates: Όταν αλλάζει μια ρύθμιση σε έναν Broker ή στο ZooKeeper, ο Operator εκτελεί μια ομαλή αναβάθμιση των pods, χωρίς να προκαλεί υπερβολική διακοπή υπηρεσίας.
- Συνεχή παρακολούθηση (monitoring): Ελέγχει την κατάσταση των pods, τα logs κ.λπ., επεμβαίνοντας αυτόματα σε περιπτώσεις αστοχίας (π.χ. επανεκκίνηση ενός Broker).

Υπάρχουν επιπλέον υπο-operators που αναλαμβάνουν άλλες λειτουργίες, όπως ο Topic Operator (για αυτόματη δημιουργία/διαγραφή Kafka topics) και ο User Operator (για διαχείριση χρηστών και ACLs).

Βασικές Λειτουργίες και Οφέλη

Δηλωτική ανάπτυξη (Declarative approach)

Ένα από τα μεγαλύτερα πλεονεκτήματα του Strimzi Operator είναι το δηλωτικό μοντέλο (YAML-based). Αντί ο διαχειριστής να διαχειρίζεται χειροκίνητα pods, services και άλλες ρυθμίσεις, καθορίζει ένα YAML αρχείο όπου ορίζει τις βασικές προδιαγραφές (π.χ. replicas: 3), και ο Operator αυτοματοποιεί τη δημιουργία και τη συντήρηση αυτής της αρχιτεκτονικής.

Κλιμακούμενη και ανθεκτική αρχιτεκτονική

Χάρη στις εγγενείς δυνατότητες του Kubernetes, ο Operator διασφαλίζει ότι τα pods που σχετίζονται με το Kafka θα αναπτυχθούν και θα αντικατασταθούν σωστά σε περίπτωση αστοχίας. Επιπλέον, το Strimzi υποστηρίζει την κλιμάκωση (scale up/down) των Brokers για να ανταποκρίνεται στις μεταβολές του φόρτου.

Αυτοματοποίηση αναβαθμίσεων

Η διαδικασία αναβαθμίσεων και ρυθμιστικών αλλαγών στο Kafka cluster γίνεται ομαλά, εφαρμόζοντας μία-μία τις αλλαγές στους Brokers, εξαλείφοντας στο μέγιστο δυνατό βαθμό τις διακοπές υπηρεσίας. Αυτό είναι ιδιαίτερα χρήσιμο σε παραγωγικά περιβάλλοντα που απαιτούν συνεχή λειτουργία.

Διαχείριση Topics και Χρηστών μέσω CRDs

Με τον Topic Operator, ορισμένα Kafka topics μπορούν να δημιουργούνται απλώς τροποποιώντας ένα CRD (KafkaTopic). Το ίδιο ισχύει και για τους χρήστες, μέσω του User Operator (KafkaUser). Αυτές οι λειτουργίες καθιστούν το Kubernetes ένα κεντρικό σημείο ελέγχου, προσφέροντας ακόμη και πολιτικές RBAC (Role-Based Access Control) για τη διαχείριση δικαιωμάτων σε επίπεδο namespace.

Ενοποίηση με άλλα εργαλεία και Καλές Πρακτικές

Παρακολούθηση και Logging

Συνήθως, εγκαθίσταται υποστήριξη για Prometheus και Grafana, ώστε να συλλέγονται metrics (CPU, memory usage, I/O, καθυστερήσεις) και να απεικονίζονται γραφικά. Ο Operator παρέχει επίσης ενσωματωμένα dashboards που βοηθούν στη γρήγορη διάγνωση προβλημάτων.

Διαχείριση Αποθήκευσης (Storage)

Σε περιβάλλον Kubernetes, οι Brokers χρειάζονται έγκυρο τρόπο για την αποθήκευση των μηνυμάτων (π.χ. Persistent Volumes). Το Strimzi Operator υποστηρίζει διάφορες επιλογές (π.χ. ephemeral ή persistent storage), δίνοντας ευελιξία ανάλογα με το αν επιδιώκεται επίμονο ιστορικό μηνυμάτων ή γρηγορότερη προσωρινή αποθήκευση.

Ασφάλεια και Κρυπτογράφηση

Το Strimzi διευκολύνει τη ρύθμιση ενδιάμεσων πιστοποιητικών TLS και την ενεργοποίηση της επικοινωνίας μέσω SSL μεταξύ Brokers και εφαρμογών-πελατών, όπως και την ενεργοποίηση ACLs για έλεγχο πρόσβασης σε topics. Έτσι, είναι εφικτή η ανάπτυξη ενός προστατευμένου Kafka cluster χωρίς επίπονες χειροκίνητες ρυθμίσεις.[21]

Ροές δεδομένων

Εισαγωγή

Η σύγχρονη επεξεργασία δεδομένων έχει εξελιχθεί σε ένα πολυσύνθετο πεδίο, το οποίο διαχειρίζεται τεράστιους όγκους και διαφορετικές μορφές πληροφορίας σε πραγματικό ή σχεδόν πραγματικό χρόνο. Από τα παραδοσιακά συστήματα μαζικής (batch) επεξεργασίας, που επιτρέπουν τη μετασχηματισμένη αποθήκευση και ανάλυση μετά τη συγκέντρωση όλων των δεδομένων, μέχρι τα πιο σύγχρονα συστήματα ροής (streaming), τα οποία παρέχουν τη δυνατότητα συνεχούς ανάλυσης με χαμηλή καθυστέρηση, η πρόκληση παραμένει να επιτευχθεί ένας συνδυασμός υψηλής απόδοσης, επεκτασιμότητας, και ορθότητας των αποτελεσμάτων.

Ωστόσο, παρά τη ραγδαία πρόοδο και την πληθώρα εργαλείων που έχουν αναπτυχθεί, παρατηρούνται ακόμη περιορισμοί σε διάφορα επίπεδα. Συχνά, η παραγωγή "σωστών" και συνεπών αποτελεσμάτων απαιτεί την επεξεργασία συμβάντων που καταφθάνουν με αταξία (out-of-order arrival) ή με σημαντικές καθυστερήσεις ως προς τον πραγματικό χρόνο δημιουργίας τους (event time). Επιπλέον, οι απαιτήσεις σε υπολογιστικούς πόρους, οι ανάγκες για συνεχή επικαιροποίηση αποτελεσμάτων και οι διαφορετικές μορφές παραθύρων (windowing), όπως χρονικά, ολισθαίνοντα ή δυναμικά ανά συνεδρία, περιπλέκουν τη διαχείριση των ροών. Σε πραγματικά περιβάλλοντα, όπως για παράδειγμα σε πλατφόρμες βίντεο με ζωντανή και ασύγχρονη θέαση, η ορθότητα έχει κρίσιμη σημασία, ειδικά όταν εμπλέκονται οικονομικές συναλλαγές ή στατιστικά στοιχεία που επηρεάζουν επιχειρηματικές αποφάσεις.

Παραδοσιακά συστήματα μαζικής επεξεργασίας δυσκολεύονται να ανταποκριθούν σε απαιτήσεις χαμηλής καθυστέρησης, ενώ αρκετά συστήματα ροής δεν παρέχουν επαρκή μηχανισμό ανοχής σε σφάλματα ή δεν μπορούν να εξασφαλίσουν επεξεργασία με *ακριβώς μία φορά (exactly-once semantics)*. Κάποια άλλα συστήματα αδυνατούν να χειριστούν σύνθετους ορισμούς παραθύρων και εξαρτώνται υπερβολικά είτε από την ώρα επεξεργασίας (processing time), είτε από την υπόθεση ότι τα δεδομένα καταφθάνουν ταξινομημένα κατά χρονική σειρά. Αυτοί οι περιορισμοί δημιουργούν κενά σε πραγματικά σενάρια, όπου οι ροές δεδομένων καταφθάνουν συχνά χαοτικά ή από διαφορετικές γεωγραφικές τοποθεσίες, και όπου η δυνατότητα επιλογής σωστού σημείου ισορροπίας μεταξύ κόστους, καθυστέρησης και ορθότητας των αποτελεσμάτων είναι κομβική.[22]

Για την αντιμετώπιση αυτών των προκλήσεων, προτείνεται ένα ενιαίο μοντέλο επεξεργασίας που δεν εξαρτάται από τον εκάστοτε μηχανισμό εκτέλεσης (batch, micro-batch, streaming). Η κεντρική ιδέα είναι η ενσωμάτωση τεσσάρων βασικών διαστάσεων:

1. Ο καθορισμός των συναρτήσεων ή μετασχηματισμών που παράγουν τα τελικά αποτελέσματα.
2. Η χρήση του event time, έτσι ώστε τα δεδομένα να ομαδοποιούνται με βάση την πραγματική στιγμή που συνέβησαν, και όχι απλώς σύμφωνα με την ώρα άφιξης στο σύστημα.

3. Η υιοθέτηση ενός μηχανισμού έναυσης (triggering) που επιτρέπει την ευέλικτη απόφαση για το πότε θα προκύπτει έξοδος από ένα *παράθυρο* δεδομένων.
4. Η δυνατότητα αναθεώρησης αποτελεσμάτων όταν καταφθάνουν νέα δεδομένα ή διορθώσεις, ώστε να παραμένει η συνολική επεξεργασία συνεπής.

Με αυτόν τον τρόπο, το μοντέλο επιτρέπει τη διατήρηση υψηλής πιστότητας στα αποτελέσματα (λόγο της δυνατότητα ορθής διαχείρισης δεδομένων που φτάνουν εκτός σειράς), ενώ αφήνει ανοιχτή την επιλογή ως προς το υποκείμενο υλικό εκτέλεσης και τους επιμέρους συμβιβασμούς σε επιδόσεις και κόστος. Σε ένα τόσο ποικιλόμορφο οικοσύστημα εφαρμογών, όπου συστήματα εκτελούν διαφορετικούς φόρτους εργασίας και έχουν διαφορετικές ανάγκες σε χρόνο απόκρισης και ακρίβεια υπολογισμών, η ύπαρξη ενός ενοποιημένου τρόπου σκέψης επιτρέπει την καλύτερη προσαρμογή της αρχιτεκτονικής στις απαιτήσεις κάθε σεναρίου.

Τέλος, ακόμα κι αν το εν λόγο μοντέλο δεν επιλύει αυτομάτως όλες τις υπολογιστικές δυσκολίες προσφέρει ένα συνεκτικό πλαίσιο που καθιστά δυνατή την απλή, αλλά και ευέλικτη, ανάπτυξη παρόμοιων ροών επεξεργασίας. Με αυτό τον τρόπο, κάθε οργανισμός ή προγραμματιστής μπορεί να επιλέγει τον κατάλληλο συνδυασμό παράθυρων (windows), χρόνου εξαγωγής (triggering), και τρόπων ενημέρωσης (incremental processing), προσαρμόζοντας το σύστημα σε ρεαλιστικές συνθήκες και στηρίζοντας καινοτόμες εφαρμογές που επωφελούνται από τα μεγάλα, πολύπλοκα και δυναμικά δεδομένα του σύγχρονου κόσμου.

Η σουίτα Apache Flink

Το Apache Flink είναι ένα ανοικτού κώδικα σύστημα επεξεργασίας δεδομένων που εστιάζει κυρίως στην επεξεργασία ροών σε πραγματικό χρόνο, προσφέροντας ωστόσο δυνατότητες και για batch επεξεργασία. Αποτελεί μια επιλογή που στηρίζεται σε αρχιτεκτονική η οποία επιτρέπει την παράλληλη και κατανεμημένη διαχείριση μεγάλων όγκων δεδομένων, υποσχόμενη υψηλή απόδοση και χαμηλές καθυστερήσεις. Η σχεδίασή του βασίζεται στην ιδέα ότι τα δεδομένα μπορούν να ρέουν συνεχώς, ενώ παράλληλα διατηρείται η δυνατότητα να αντιμετωπίζονται αυτές οι ροές με τρόπο που μοιάζει πολύ με τον παραδοσιακό τρόπο batch επεξεργασίας, διατηρώντας ομοιότητες στον τρόπο γραφής του κώδικα και στη λογική υλοποίησης.

Πέρα από το γεγονός ότι εστιάζει πρωτίστως στην επεξεργασία ροών, το Flink ενσωματώνει τεχνικές που επιτρέπουν την ανθεκτικότητα σε σφάλματα, τη διαχείριση κατάστασης σε πραγματικό χρόνο και την ευελιξία προσαρμογής σε διαφορετικά σενάρια χρήσης. Με την αξιοποίηση της τεχνολογίας του, μπορεί να αναπτυχθεί ένα οικοσύστημα αναλύσεων και εφαρμογών που καλύπτουν από την ανάγνωση και φιλτράρισμα δεδομένων μέχρι την εκτέλεση αλγορίθμων σε πραγματικό χρόνο. Το σύστημα ενσωματώνεται καλά με άλλα εργαλεία, επιτρέποντας έτσι τη δημιουργία ολοκληρωμένων ροών εργασίας σε εταιρικά περιβάλλοντα, όπου συχνά απαιτείται η σύνδεση με διαχειριστές μηνυμάτων, βάσεις δεδομένων και πλατφόρμες ανάλυσης.

Σε γενικές γραμμές, το Flink έρχεται να καλύψει την ανάγκη για ένα κεντρικό και ευέλικτο πλαίσιο που θα δίνει τη δυνατότητα άμεσης επεξεργασίας δεδομένων με ανθεκτικότητα και δυνατότητα κλιμάκωσης. Λόγω του τρόπου με τον οποίο είναι σχεδιασμένο, προσφέρει μια συγκροτημένη προσέγγιση για τη δημιουργία εφαρμογών πραγματικού χρόνου που πρέπει να

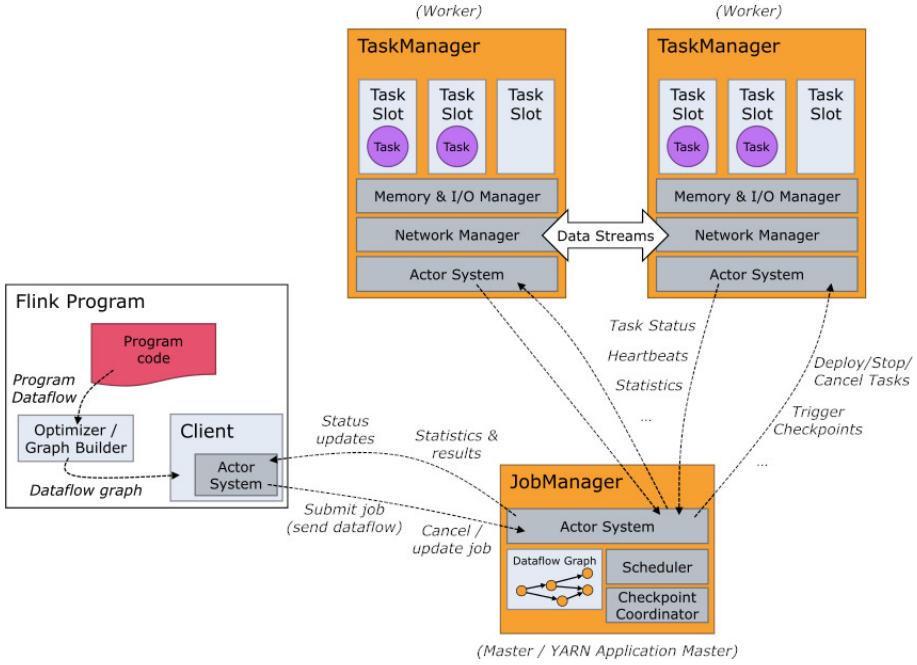
λειτουργούν αδιάλειπτα, ανεξάρτητα από την κλίμακα και τον ρυθμό μεταβολής των δεδομένων. Όλα αυτά το καθιστούν μια ιδιαίτερα δημοφιλή επιλογή στον χώρο του big data, χωρίς να αποκλείει τη χρήση του σε απλούστερα ή πιο συμβατικά περιβάλλοντα όπου υπάρχει ανάγκη διαρκούς ανάλυσης και επεξεργασίας.[23]

Η αρχιτεκτονική του Apache Flink

Βασικές μονάδες

Το Flink έχει σχεδιαστεί ώστε να ανταποκρίνεται στις απαιτήσεις μεγάλων συστημάτων, όπου η παραλληλοποίηση, η διαχείριση πόρων και η αξιοπιστία είναι κρίσιμες.

Σε ολιστικό επίπεδο η λειτουργία του Flink βασίζεται στη συνεργασία δύο βασικών οντοτήτων, του JobManager και των TaskManagers.



Εικόνα 6: Αρχιτεκτονική Flink in Standalone mode

Job Manager

Ο JobManager αναλαμβάνει το ρόλο του κεντρικού συντονιστή του cluster. Όταν υποβάλλεται μια διεργασία τυπου Flink. ο JobManager αναλαμβάνει την ανάλυση και τη μετατροπή της σε έναν γράφο ροής εργασίας (DAG – Directed Acyclic Graph), όπου κάθε κόμβος αντιπροσωπεύει ένα συγκεκριμένο task. Μέσα από αυτόν τον γράφο, καθορίζονται οι εξαρτήσεις και η ροή δεδομένων, γεγονός που επιτρέπει την ομαλή κατανομή και εκτέλεση των εργασιών. Πιο αναλυτικά, ο ρολος του JobManager επεκτείνεται σε τρία βασικά υποσυστήματα: ResourceManager, Dispatcher και τον JobMaster.

ResourceManager

Ο ResourceManager είναι υπεύθυνος για την κατανομή και διαχείριση των πόρων σε ένα cluster Flink.

Συγκεκριμένα είναι υπεύθυνος για την:

- **Διαχείριση Task Slots:** Διαχειρίζεται τα task slots, που αποτελούν τις βασικές μονάδες προγραμματισμού πόρων στο Flink. Κάθε TaskManager παρέχει έναν αριθμό slots, όπου κάθε slot είναι μια μονάδα επεξεργασίας για τα tasks.
- **Υποστήριξη για Διάφορα Περιβάλλοντα:** Το Flink διαθέτει διαφορετικές υλοποιήσεις του ResourceManager για περιβάλλοντα όπως το YARN, το Kubernetes και τις standalone εγκαταστάσεις. Σε περιβάλλοντα standalone, ο ResourceManager λειτουργεί αποκλειστικά με τους υπάρχοντες TaskManagers, διανέμοντας τα διαθέσιμα slots, χωρίς δυνατότητα αυτόματης εκκίνησης νέων TaskManagers.

Με αυτόν τον τρόπο, ο ResourceManager εξασφαλίζει ότι οι διαθέσιμοι πόροι χρησιμοποιούνται αποδοτικά και ότι κάθε εργασία έχει πρόσβαση στις απαραίτητες μονάδες επεξεργασίας.

Dispatcher

Το Dispatcher λειτουργεί ως το κεντρικό σημείο εισόδου για τις εφαρμογές Flink. Οι κύριες λειτουργίες του περιλαμβάνουν την:

- **Υποβολή Εφαρμογών:** Παρέχει μια REST διεπαφή μέσω της οποίας οι χρήστες μπορούν να υποβάλουν τις εφαρμογές τους για εκτέλεση.
- **Εκκίνηση του JobMaster:** Για κάθε υποβληθείσα εργασία, το Dispatcher ξεκινά έναν νέο JobMaster, ο οποίος θα αναλάβει τη διαχείριση της εκτέλεσης αυτής της συγκεκριμένης εργασίας.
- **Παροχή Πληροφοριών:** Τρέχει την διεπαφή χρήστη του Flink, το οποίο δίνει πρόσβαση σε πληροφορίες για την κατάσταση και την πρόοδο των εργασιών.

Με τον τρόπο αυτό, το Dispatcher καθιστά τη διαδικασία υποβολής και παρακολούθησης των εφαρμογών διαφανή και προσβάσιμη, επιτρέποντας την εύκολη διαχείριση του συστήματος.

JobMaster

Ο JobMaster (παλαιότερα γνωστός ως JobManager) είναι υπεύθυνος για τη διαχείριση της εκτέλεσης μιας μεμονωμένης εργασίας, η οποία αναπαρίσταται από έναν JobGraph. Κάθε JobMaster έχει ως καθήκοντα:

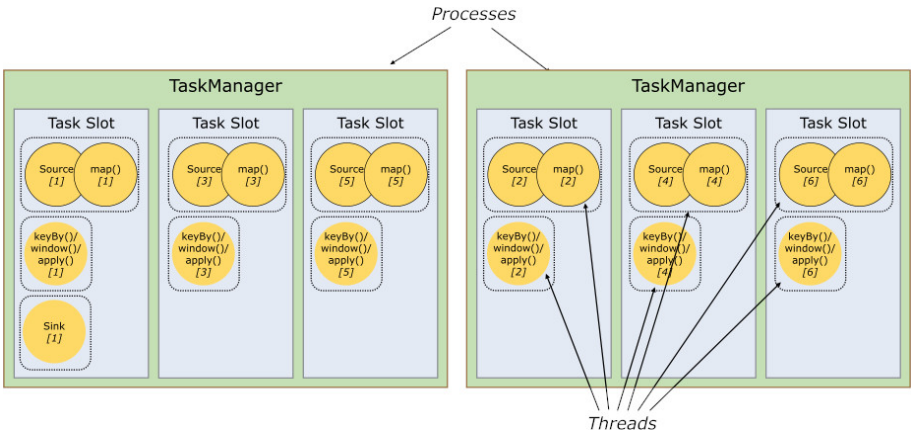
- **Χρονοπρογραμματισμό Εργασιών:** Αποφασίζει πότε θα προγραμματιστεί το επόμενο task ή ομάδα tasks, λαμβάνοντας υπόψη τις εξαρτήσεις που υπάρχουν μεταξύ τους.
- **Παρακολούθηση Εκτέλεσης:** Αντιδρά στις ολοκληρωμένες εργασίες ή στις αποτυχίες κατά την εκτέλεση, διασφαλίζοντας ότι κάθε μέρος της εφαρμογής λειτουργεί σωστά.

- **Συντονισμός Checkpoints και Recovery:** Διαχειρίζεται τη διαδικασία δημιουργίας checkpoints για την αποθήκευση της κατάστασης της εφαρμογής και τον συντονισμό της επαναφοράς σε περίπτωση σφαλμάτων.

Καθώς σε ένα cluster μπορούν να εκτελούνται πολλαπλές εργασίες ταυτόχρονα, κάθε εφαρμογή έχει τον δικό της JobMaster, ο οποίος παρακολουθεί και διαχειρίζεται ανεξάρτητα την εξέλιξη της συγκεκριμένης εργασίας.

Task Managers

Οι TaskManagers αποτελούν τους κόμβους εκτέλεσης στο Apache Flink και είναι υπεύθυνοι για την πραγματική επεξεργασία των δεδομένων. Κάθε TaskManager εκτελεί τα tasks που του ανατίθενται από τον JobManager, σύμφωνα με τον καθορισμένο βαθμό παραλληλισμού της εφαρμογής. Κάθε TaskManager διαθέτει έναν αριθμό από execution slots (ή task slots), οι οποίοι αντιπροσωπεύουν τις ανεξάρτητες μονάδες υπολογισμού εντός του ίδιου κόμβου. Κάθε slot μπορεί να εκτελέσει ένα ή περισσότερα υπο-τμήματα (subtasks) ενός Flink job, επιτρέποντας την ταυτόχρονη εκτέλεση πολλαπλών ροών επεξεργασίας στον ίδιο φυσικό πόρο. Μέσω αυτής της κατανεμημένης και παραλληλοποιημένης δομής, το Flink επιτυγχάνει υψηλή απόδοση, αποδοτική χρήση πόρων και δυνατότητα οριζόντιας επεκτασιμότητας για την επεξεργασία μεγάλου όγκου δεδομένων σε πραγματικό χρόνο.



Εικόνα 7: Οπτική αναπαράσταση επιμέρους μονάδων ενός TaskManager

Επιπλέον, οι TaskManagers είναι υπεύθυνοι για τη διαχείριση της κατάστασης μια ροής (state), το οποίο είναι κρίσιμο για εφαρμογές που απαιτούν stateful επεξεργασία. Αυτό το state μπορεί να αποθηκεύεται και να ανακτηθεί σε περιπτώσεις σφαλμάτων, διασφαλίζοντας την αξιοπιστία της εφαρμογής.

Ένα από τα χαρακτηριστικά που καθιστούν το Apache Flink ιδιαίτερα αξιόπιστο είναι η ενσωμάτωση μηχανισμών για την αντιμετώπιση σφαλμάτων και την αποκατάσταση της λειτουργίας μετά από διακοπές. Μέσω των checkpoints, το σύστημα δημιουργεί περιοδικά στιγμιότυπα της κατάστασης των εφαρμογών, ώστε, σε περίπτωση αποτυχίας, να μπορεί να επαναφέρει το state από το τελευταίο ασφαλές σημείο. Παράλληλα, τα checkpoints παρέχουν

τη δυνατότητα χειροκίνητης λήψης στιγμιότυπων, κάτι που είναι ιδιαίτερα χρήσιμο κατά τις διαδικασίες αναβάθμισης ή συντήρησης.

Για την ορθή διαχείριση των ροών δεδομένων και την αντιμετώπιση πιθανών καθυστερήσεων (lateness), το Flink υιοθετεί το concept των watermarks. Τα watermarks λειτουργούν ως χρονικοί δείκτες που καθορίζουν πότε είναι ασφαλές να ολοκληρωθεί η επεξεργασία ενός συγκεκριμένου χρονικού παραθύρου, επιτρέποντας έτσι την ακριβή διαχείριση του event time και την σωστή συγχώνευση των δεδομένων που φτάνουν με καθυστέρηση.[24]

Παράλληλη Εκτέλεση και Job Graph

Το Flink έχει σχεδιαστεί ώστε να εκμεταλλεύεται πλήρως ένα κατανεμημένο περιβάλλον. Κάθε operator μπορεί να εκτελείται σε πολλαπλά αντίγραφα (parallel instances), κατανεμημένα σε διάφορους κόμβους, επιτυγχάνοντας υψηλή απόδοση ακόμη και σε μεγάλα φορτία. Από τη στιγμή που θα υποβληθεί μια εφαρμογή για εκτέλεση, το Flink δημιουργεί το λεγόμενο job graph, στο οποίο φαίνονται οι operators (πηγές, μετασχηματισμοί, sinks) και οι σχέσεις τους. Αυτή η αναπαράσταση επιτρέπει στο σύστημα να κατανείμει αρμονικά τα υπο-εργασίες (subtasks) μεταξύ των διαθέσιμων κόμβων και να κλιμακώνεται οριζόντια, καλύπτοντας πιο απαιτητικά σενάρια χρήσης χωρίς να υπονομεύεται η απόδοση ή η συνέχεια της επεξεργασίας.

Διασύνδεση (Connectors)

Το Flink ενσωματώνεται με εξωτερικά συστήματα αποθήκευσης και μηνυμάτων μέσω connectors. Στην δική μας περίπτωση, χρησιμοποιούνται συγκεκριμένα οι Kafka connectors για να λαμβάνουμε δεδομένα από τα Kafka topics και να επιστρέφουμε τα αποτελέσματά μας σε άλλα ή στα ίδια topics. Αυτή η στενή ολοκλήρωση με το Kafka διευκολύνει εξαιρετικά τη συνεχή λήψη γεγονότων και την αποστολή συμπερασμάτων ή ενδιάμεσων αποτελεσμάτων, επιτυγχάνοντας μία πλήρως *stream-oriented* αρχιτεκτονική.

Τρόποι επεξεργασίας ροών

Η επεξεργασία ροών στο Flink βασίζεται σε μια συνεχή ροή δεδομένων όπου τα γεγονότα επεξεργάζονται καθώς εμφανίζονται. Σε αντίθεση με την παραδοσιακή επεξεργασία παρτίδων, η οποία λειτουργεί σε περιορισμένα σύνολα δεδομένων, η επεξεργασία ροών λειτουργεί σε δυνητικά άπειρες ροές δεδομένων.

Η επεξεργασία ροών στο Flink υποστηρίζει δύο λειτουργίες:

1. **Χρόνος Συμβάντων (Event Time)**: Τα γεγονότα επεξεργάζονται με βάση χρονικές σημάνσεις που αποδίδονται από τον παραγωγό. Αυτή η λειτουργία εξασφαλίζει ότι τα γεγονότα επεξεργάζονται με σωστή χρονική σειρά, ανεξάρτητα από το πότε φτάνουν.
2. **Χρόνος Επεξεργασίας (Processing Time)**: Σε αυτή τη λειτουργία, το Flink επεξεργάζεται τα γεγονότα με βάση την ώρα του συστήματος όταν λαμβάνεται το

γεγονός, το οποίο είναι χρήσιμο σε περιπτώσεις όπου η χρονική σήμανση του γεγονότος δεν είναι διαθέσιμη ή δεν έχει σημασία.

Μέσω των δυνατοτήτων του στην επεξεργασία ροών, το Flink προσφέρει προηγμένες λειτουργίες όπως **παραθυροποίηση (windowing)**, **επεξεργασία με κατάσταση (stateful processing)**, και **watermarking**, τα οποία διαχειρίζονται αργοπορημένα γεγονότα και δεδομένα εκτός σειράς με αποδοτικό τρόπο.

Watermarking

Η έννοια του **watermark** αποτελεί καθοριστικό μηχανισμό για την ορθή διαχείριση γεγονότων (events) που ενδέχεται να καταφτάνουν σε τυχαία ή καθυστερημένη σειρά (out-of-order ή late arrivals). Ειδικότερα, όταν μια εφαρμογή βασίζεται στο **event-time**, επιθυμούμε να πραγματοποιούμε υπολογισμούς (όπως παράθυρα, αθροίσεις) βάσει της πραγματικής χρονικής στιγμής δημιουργίας κάθε γεγονότος, και όχι βάσει της χρονικής στιγμής που φτάνει το γεγονός στο σύστημα.

Σε μια ιδανική περίπτωση, όλα τα events ενός συγκεκριμένου χρονικού διαστήματος θα έφταναν έγκαιρα και με σωστή σειρά, οπότε δεν θα υπήρχε δυσκολία. Στην πράξη, όμως, συχνά υπάρχουν δικτυακές καθυστερήσεις ή άλλοι παράγοντες που κάνουν τα δεδομένα να έρχονται ακανόνιστα: μπορεί ένα συμβάν που δημιουργήθηκε νωρίς να φτάσει αργότερα από ένα συμβάν που δημιουργήθηκε αργότερα. Σε αυτήν την περίπτωση, αν ο αλγόριθμος βασιζόταν αποκλειστικά στη σειρά παραλαβής των γεγονότων (processing time), οι υπολογισμοί στα χρονικά παράθυρα θα αποδεικνύονταν ανακριβείς.[25]

Τα **watermarks** είναι, ουσιαστικά, *δήλωση* ή *σήμα* προς το σύστημα επεξεργασίας ότι *έχει περάσει το χρονικό διάστημα t, και θεωρούμε πως έχουμε λάβει όλα τα σημαντικά γεγονότα ως αυτό το σημείο*. Μέσω αυτής της διαδικασίας πραγματοποιείται:

1. **Ορισμός χρονικής ανοχής (out-of-orderness bound):** Όταν ορίζουμε έναν κανόνα για τα watermarks (π.χ. Watermark = eventTime - 2 δευτερόλεπτα), ουσιαστικά λέμε στο σύστημα ότι επιτρέπουμε στα γεγονότα να έρθουν έως και δύο δευτερόλεπτα αργοπορημένα σε σχέση με τη χρονική σήμανσή τους, προτού συμπεράνουμε ότι έχουμε λάβει όσα γεγονότα ανήκουν σε εκείνο το χρονικό πλαίσιο.
2. **Κλείσιμο των παραθύρων:** Αν για παράδειγμα διαθέτουμε ένα παράθυρο (window) που ξεκινά στις 10:00:00 και διαρκεί 1 λεπτό, το σύστημα δεν θα βγάλει τελικό αποτέλεσμα ακριβώς στη λήξη του παραθύρου, αλλά θα περιμένει μέχρι το watermark να *ξεπεράσει* το 10:01:00 (συν την όποια ανοχή έχουμε ορίσει). Έτσι διασφαλίζεται ότι ακόμη κι αν φτάσει ένα γεγονός με timestamp 10:00:30 αλλά με κάποια καθυστέρηση, δεν θα χαθεί από τον υπολογισμό.
3. **Λογική καθυστερημένων γεγονότων:** Σε ορισμένες περιπτώσεις, μπορεί να εμφανιστούν γεγονότα που φτάνουν πολύ αργότερα από το watermark. Αυτά θεωρούνται *late arrivals* και, ανάλογα με την πολιτική που έχει επιλέξει ο σχεδιαστής της εφαρμογής (π.χ. απόρριψη ή επανεκτίμηση των υπολογισμών), μπορεί να υποστούν ξεχωριστή μεταχείριση.

Σε ένα πιο θεωρητικό πλαίσιο, τα watermarks επιλύουν το πρόβλημα της **έλλειψης κεντρικού ρολογιού** σε κατανεμημένα συστήματα. Επειδή τα γεγονότα δημιουργούνται και διακινούνται με διαφορετικές καθυστερήσεις, η ύπαρξη ενός μηχανισμού που «υπολογίζει» πόσο χρόνο περιμένουμε για τυχόν αργοπορημένα δεδομένα είναι κρίσιμη. Με άλλα λόγια, τα watermarks επιτρέπουν στον αλγόριθμο να έχει μια συνεπή αντίληψη του χρονικού συνεχούς, χωρίς να απαιτείται σκληρός συγχρονισμός μεταξύ όλων των κόμβων ή συσκευών.

Η επίτευξη της κατάλληλης ισορροπίας, λοιπόν, γίνεται μελετώντας τα χαρακτηριστικά των ροών (μέγιστη πιθανή καθυστέρηση άφιξης) και τις απαιτήσεις της εφαρμογής (πόση καθυστέρηση επιτρέπεται πριν παραχθεί ένα αποτέλεσμα). Από τη στιγμή που οριστεί η στρατηγική δημιουργίας watermarks (bounded out-of-orderness, punctuated κ.λπ.), το σύστημα θα είναι σε θέση να *κλείνει* εγκαίρως τα χρονικά παράθυρα, να παράγει ασφαλή υπολογιστικά αποτελέσματα και ταυτόχρονα να απορροφά ένα λογικό επίπεδο καθυστερημένων γεγονότων χωρίς να διαταράσσεται η συνοχή της επεξεργασίας.

Επεξεργασίας ροών που διακρίνουν εσωτερική κατάσταση (Stateful stream processing)

Ένα από τα κύρια χαρακτηριστικά που διαφοροποιούν το Apache Flink είναι η δυνατότητά του για κατάσταση επεξεργασίας ροών. Η κατάσταση επιτρέπει στο Flink να διατηρεί πληροφορίες για τα γεγονότα ή τις ακολουθίες γεγονότων διαχρονικά. Αυτό είναι ιδιαίτερα χρήσιμο σε σενάρια όπως η αναγνώριση προτύπων γεγονότων, η σύνθετη επεξεργασία γεγονότων, ή ακόμη και η μηχανική μάθηση, όπου το σύστημα πρέπει να θυμάται προηγούμενα γεγονότα για να λάβει μελλοντικές αποφάσεις.

Η κατάσταση στο Flink είναι ανθεκτική στις αστοχίες και διαχειρίζεται μέσω κατανεμημένων στιγμιότυπων (distributed snapshots) για να καταγράφει μια συνεπή εικόνα του συστήματος σε κάθε χρονική στιγμή. Αυτά τα στιγμιότυπα επιτρέπουν στο Flink να ανακάμψει από αποτυχίες, αποκαθιστώντας την κατάσταση της εφαρμογής και συνεχίζοντας την επεξεργασία από το ακριβές σημείο όπου διακόπηκε. Επιπλέον, η κατάσταση του Flink διαχειρίζεται αποτελεσματικά και μπορεί να κλιμακωθεί απρόσκοπτα με το μέγεθος της εφαρμογής.

Keyed State και Stateful Stream Processing

Σε ένα πλαίσιο όπου οι εφαρμογές επεξεργασίας ροών απαιτούν όχι μόνο την απλή ανάγνωση και μετασχηματισμό δεδομένων, αλλά και την αξιοποίηση της ιστορικής πληροφορίας ανά οντότητα, η αρχιτεκτονική της Flink ενσωματώνει την έννοια του keyed state. Συγκεκριμένα, κάθε γεγονός (event) που φέρει μια πληροφορία κλειδί μπορεί να εκτρέπει την εκτέλεση σε έναν ξεχωριστό νοητό χώρο επεξεργασίας που αφορά αποκλειστικά αυτό το κλειδί. Αυτός ο χώρος επεξεργασίας, ο οποίος ονομάζεται και αλλιώς state, μπορεί να πάρει διάφορες μορφές/ παραλλαγές, όπως:

- **ValueState**: Κρατάει μία μοναδική τιμή, χρήσιμη για αποθήκευση του πιο πρόσφατου γεγονότος ή μιας υπολογισμένης στατιστικής (π.χ. τρέχον μέσος όρος).
- **ListState**: Αποθηκεύει πολλαπλές τιμές ή γεγονότα, επιτρέποντας τη σωρευτική συλλογή τους πριν από την περαιτέρω επεξεργασία, κάτι ιδιαίτερα χρήσιμο σε σενάρια που απαιτούν παράθυρα ή μικρά *batch* δεδομένων.

- **MapState**: Διατηρεί ζεύγη κλειδιού-τιμής, ιδανικό σε περιπτώσεις όπου χρειάζεται δυναμική αναζήτηση (lookup) με διαφορετικά κλειδιά εντός της ίδιας ροής ή πλήρη ευελιξία στη δομή των δεδομένων.

Με αυτό τον τρόπο, οι εφαρμογές μπορούν να συσχετίζουν γεγονότα που αφορούν την ίδια οντότητα, ανεξαρτήτως του χρονικού διαστήματος στο οποίο συμβαίνουν, διαμορφώνοντας μια πολυπλοκότερη και πιο *επίμονη* μνήμη των όσων έχουν συμβεί. Είναι ιδιαίτερα χρήσιμο, για παράδειγμα, σε σενάρια εκπαίδευσης μοντέλων μηχανικής μάθησης, όπου κάθε client_id συγκεντρώνει τα δικά του δεδομένα και ενημερώνει ξεχωριστά το μοντέλο του, ή σε εφαρμογές όπου απαιτείται η ανίχνευση μοτίβων (patterns) στη ροή, λαμβάνοντας υπόψη το ιστορικό γεγονότων για κάθε κλειδί.

Η αξία αυτής της *κατάστασης ανά κλειδί*, είναι εμφανής σε κάθε ροή που έχει πολυάριθμα κλειδιά, καθώς το Flink διαχειρίζεται αυτόματα την κατανομή των κλειδιών στους κόμβους του cluster, φροντίζοντας να διατηρείται τόσο η απόδοση όσο και η συνεκτικότητα των υπολογισμών. Επιπλέον, χάρη στον ενσωματωμένο μηχανισμό checkpointing, η κατάσταση που είναι αποθηκευμένη σε καθέναν από αυτούς τους κόμβους μπορεί να ανακτάται σε περίπτωση σφάλματος, εξασφαλίζοντας ακεραιότητα και συνέχεια στην επεξεργασία.[26]

Επεξεργασία Παρτίδων (Batch Processing) ως επέκταση της Ροής

Παρότι η Flink διακρίνεται πρωτίστως για την επεξεργασία ροών, υποστηρίζει παράλληλα και την επεξεργασία δεδομένων που έχουν πεπερασμένο μέγεθος. Στην πραγματικότητα, η αρχιτεκτονική του Flink μοντελοποιεί την επεξεργασία παρτίδων ως μια *ειδική περίπτωση* επεξεργασίας ροών, στην οποία η πηγή δεδομένων (source) δεν παράγει γεγονότα επ' αόριστόν, αλλά σταματά όταν φτάσει στο τέλος του διαθέσιμου συνόλου. Αυτή η ενοποιημένη προσέγγιση επιτρέπει στους χρήστες να αξιοποιούν το ίδιο μοντέλο προγραμματισμού, ανεξαρτήτως αν το πρόβλημα απαιτεί real-time ανάλυση ή επεξεργασία δεδομένων ιστορικού.

Η Flink διαθέτει συγκεκριμένο API αποκλειστικά για επεξεργασία σε παρτίδες, αυτό συνδυάζει τις βελτιστοποιήσεις της ροής με τεχνικές που παραδοσιακά συναντώνται σε parallel batch frameworks, ώστε να μπορεί να διαχειριστεί συγκεντρώσεις (aggregations), συνενώσεις (joins) και διάφορους μετασχηματισμούς σε μεγάλης κλίμακας σύνολα δεδομένων. Το αποτέλεσμα είναι ένα ενιαίο περιβάλλον ανάπτυξης, στο οποίο η ίδια λογική επεξεργασίας μπορεί να εφαρμοστεί είτε σε ένα αέναο data stream είτε σε μια αποθηκευμένη συλλογή αρχείων ή εγγραφών.

Windowing

Μια κεντρική έννοια στην επεξεργασία ροών είναι η έννοια των **παραθύρων**, τα οποία επιτρέπουν την ομαδοποίηση των ροών δεδομένων με βάση το χρόνο ή άλλα κριτήρια. Τα παράθυρα επιτρέπουν στο Flink να χωρίζει μια άπειρη ροή δεδομένων σε πεπερασμένα κομμάτια, τα οποία στη συνέχεια μπορούν να υποστούν επεξεργασία ξεχωριστά.

Το Flink υποστηρίζει διάφορες στρατηγικές παραθυροποίησης, όπως:

- **Παράθυρα Σταθερού Μεγέθους (Tumbling Windows)**: Παράθυρα με σταθερό μέγεθος που δεν αλληλεπικαλύπτονται, όπου κάθε γεγονός ανήκει ακριβώς σε ένα παράθυρο.

- **Παράθυρα Ολίσθησης (Sliding Windows)**: Παράθυρα σταθερού μεγέθους με επικαλυπτόμενες περιόδους. Ένα γεγονός μπορεί να ανήκει σε πολλά παράθυρα με βάση τη χρονική του σήμανση.
- **Παράθυρα Συνεδρίας (Session Windows)**: Παράθυρα που καθορίζονται από περιόδους αδράνειας ή κενά μεταξύ των γεγονότων, ιδανικά για ανάλυση συνεδριών χρηστών.

Checkpoint

Το **checkpointing** είναι ο μηχανισμός που επιτρέπει στο σύστημα να εξασφαλίζει ανθεκτικότητα σε σφάλματα (fault tolerance) και να εγγυάται την ακεραιότητα της επεξεργασίας, ακόμη και σε κατανεμημένο περιβάλλον με πολλούς κόμβους. Στην πράξη, η Flink περιοδικά *παγώνει* τη ροή και καταγράφει την τρέχουσα κατάσταση (state) όλων των operators, καθώς και τη θέση ανάγνωσης των πηγών δεδομένων (όπως offsets σε Kafka). Αυτό το *στιγμιότυπο (snapshot)* αποθηκεύεται σε έναν ανθεκτικό αποθηκευτικό χώρο μαζί με τις απαραίτητες πληροφορίες για την επαναφορά (restore) της εκτέλεσης σε περίπτωση σφάλματος.

Όταν ένας κόμβος ή ολόκληρο το cluster αντιμετωπίσει αστοχία, η Flink μπορεί να ανακτήσει τη ροή δεδομένων από το πιο πρόσφατο checkpoint, συνεχίζοντας από εκεί την επεξεργασία χωρίς να χάνεται ή να υπολογίζεται διπλά κανένα γεγονός. Η στρατηγική αυτή υποστηρίζει το μοντέλο **exactly-once** semantics, που διασφαλίζει ότι όλα τα μηνύματα θα επεξεργαστούν ακριβώς μία φορά, ακόμη κι αν μεσολαβήσει επανεκκίνηση. Με το checkpointing, επομένως, επιτυγχάνεται αξιόπιστη συνέχεια της επεξεργασίας ροών, κάτι κομβικό για εφαρμογές που απαιτούν σταθερή λειτουργία και ακρίβεια στα αποτελέσματα, όπως χρηματοοικονομικές πλατφόρμες, ανάλυση σε πραγματικό χρόνο και βιομηχανικά περιβάλλοντα IoT.[27]

Backpressure

Το backpressure είναι ο μηχανισμός αυτορρύθμισης με τον οποίο το Flink αποτρέπει την υπερφόρτωση της ροής όταν κάποιο στάδιο επεξεργασίας ή εξόδου δεν μπορεί να καταναλώσει δεδομένα με τον ίδιο ρυθμό που παράγονται. Πρακτικά, όταν ένας operator *«στενέψει»*, γεμίζουν οι ενδιάμεσοι καταχωρητές και ο ίδιος σταματά προσωρινά να δέχεται νέες εγγραφές. Το σήμα αυτό διαδίδεται προς τα πίσω tasks (upstream) καθυστερώντας αναλογικά τους προηγούμενους operators και τελικά τις πηγές (π.χ. αναγνώσεις από Kafka). Έτσι διατηρείται η σταθερότητα του συστήματος. Ο ρυθμός παραγωγής προσαρμόζεται στον ρυθμό κατανάλωσης αντί να συσσωρεύονται ανεξέλεγκτα δεδομένα ή να καταρρεύσει ο κόμβος. Το φαινόμενο συνοδεύεται από αύξηση καθυστέρησης (latency), καθυστέρηση υδατοσήμων (watermark lag) και μείωση της διαμεταγωγής (throughput), και αποτυπώνεται ρητά σε μετρικές που το αφορούν με ομώνυμες ονομασίες. Οι αιτίες μπορεί να είναι σημεία συμφόρησης σε δίσκο/δίκτυο, δύσκολος μετασχηματισμός, μη ισομερής κατανομή κλειδιών ή ανεπαρκής παραλληλισμός/partitioning. Η αντιμετώπιση γίνεται στοχευμένα, με αύξηση βαθμού παραλληλισμού εκεί όπου εντοπίζεται η συμφόρηση, καλύτερο partitioning (περισσότερα Kafka partitions), περιορισμός περιττών ανακατανομών κλειδιών, βελτίωση serialization/I/O ή ασύγχρονοι sinks. Με αυτόν τον τρόπο το backpressure λειτουργεί ως

«φρένο ασφαλείας», θυσιάζοντας προσωρινά καθυστέρηση για να διατηρήσει την ορθότητα και τη διαθεσιμότητα της ροής.

Skew

Το skew (ανισοκατανομή φόρτου) εμφανίζεται όταν τα εισερχόμενα δεδομένα δεν μοιράζονται ομοιόμορφα στις παράλληλες υπο-εργασίες ενός operator, λίγα "καυτά" κλειδιά, τμήματα ροής ή partitions συγκεντρώνουν δυσανάλογο όγκο, με αποτέλεσμα ορισμένα subtasks να υπερφορτώνονται ενώ άλλα μένουν υποαπασχολημένα. Ως αποτέλεσμα, παρατηρείται αυξημένη καθυστέρηση και backpressure σε λίγους κόμβους, χαμηλή συνολική αξιοποίηση πόρων και σε stateful στάδια (π.χ. παράθυρα ή aggregations), μεγάλα, άνισα states που επιβαρύνουν τα checkpoints και αυξάνουν τον κίνδυνο να βγεις εκτός μνήμης. Ο εντοπισμός τέτοιων περιπτώσεων γίνεται μέσω μετρικών ανά subtask (records in/out, busy time, heap/state size, consumer lag, watermark lag) που δείχνουν αυτή την ετερογένεια. Η αντιμετώπιση στοχεύει στην εξισορρόπηση μέσω της αύξησης και καλύτερης επιλογής partitioning στην πηγή (περισσότερα Kafka partitions), πιο διάσπαρτα κλειδιά ή σύνθετα κλειδιά, key salting με δίφασες aggregations (σπάμε το "καυτό" κλειδί σε πολλά ψευδοκλειδιά και επανασυνθέτουμε downstream), χρήση rebalance/rescale πριν από "βαριά" στάδια για ομοιόμορφη ροή, broadcast join όταν η μία πλευρά είναι μικρή, καθώς και στοχευμένο scale-out (αύξηση του βαθμού παραλληλισμού και επαρκή slots) μόνο όπου υπάρχει πραγματικό bottleneck.. Με τις τεχνικές που αναφέραμε παραπάνω, το skew μετατρέπεται σε ελεγχόμενο φαινόμενο, διατηρώντας υψηλό throughput και προβλέψιμο latency.[28]

Flink & K8s

Ο **Flink Kubernetes Operator** αποτελεί μια ολοκληρωμένη λύση για την ανάπτυξη και τη διαχείριση εφαρμογών Apache Flink σε περιβάλλοντα Kubernetes, αξιοποιώντας το καθιερωμένο "Operator pattern". Αντί ο χρήστης να επιμελείται χειροκίνητα τις ρυθμίσεις και την αρχιτεκτονική του Flink (JobManagers, TaskManagers), ο Operator αναλαμβάνει αυτόνομα τη δημιουργία, την κλιμάκωση και την αναβάθμιση του σχετικού υποσυστήματος.

Η λειτουργία του βασίζεται σε Custom Resource Definitions (CRDs) οι οποίες περιγράφουν με δηλωτικό τρόπο (declarative approach) την επιθυμητή κατάσταση του συστήματος, για παράδειγμα, τον αριθμό των TaskManagers, τη σύσταση των pods, την παραλληλία εκτέλεσης κ.λπ. Ο Operator παρακολουθεί διαρκώς την τρέχουσα κατάσταση του cluster Kubernetes, συγκρίνει τις απαραίτητες ρυθμίσεις με τις δηλωμένες στο CRD και εκτελεί τις απαιτούμενες ενέργειες (όπως η δημιουργία ή καταστροφή pods) ώστε να επιτευχθεί η επιθυμητή κατάσταση. Με αυτόν τον τρόπο, ο κύκλος ζωής μιας εφαρμογής Flink (ανάπτυξη, επανεκκίνηση, κλιμάκωση, αναβάθμιση) ενσωματώνεται αρμονικά στις δυνατότητες που παρέχει το Kubernetes.

Επιπλέον, ο Flink Kubernetes Operator επιτρέπει την ομαλή ενσωμάτωση μηχανισμών όπως τα checkpoints και τα savepoints, διασφαλίζοντας έτσι την ακεραιότητα και τη συνέχιση της επεξεργασίας ροών σε απαιτητικές συνθήκες, όπως η αλλαγή εκδοχής μιας εφαρμογής ή η αντιμετώπιση σφαλμάτων. Ως αποτέλεσμα, οι προγραμματιστές μπορούν να επικεντρωθούν

στην ανάπτυξη των Flink jobs και των υποκείμενων αλγορίθμων επεξεργασίας, ενώ ο Operator αναλαμβάνει τη σύγχρονη, κλιμάκωση και αξιόπιστη εκτέλεσή τους στο Kubernetes.[29]

2. Σχεδίαση & Υλοποίηση

Στην παρούσα διπλωματική υλοποιείται ένα σύστημα λήψης και επεξεργασίας μετρήσεων ολικής κατανάλωσης για ομάδες συσκευών (οντότητες). Για λόγους σαφήνειας εστιάζουμε σε πολλά οικίες και θεωρούμε ότι, για ορισμένα χρονικά διαστήματα, διαθέτουμε μια περιστασιακή παρατήρηση της κατάστασης των επί μέρους συσκευών (μέσω μιας υποθετικής διεπαφής χρήστη). Σε αυτή τη ροή, ο στόχος μας δεν είναι η ίδια η πρόβλεψη ή η εκπαίδευση μοντέλων, αλλά η κατασκευή των χρονικών παραθύρων και των συνόλων δεδομένων που θα χρησιμοποιηθούν απευθείας σε αλγορίθμους μηχανικής μάθησης. Συγκεκριμένα, σε πραγματικό χρόνο δημιουργούμε:

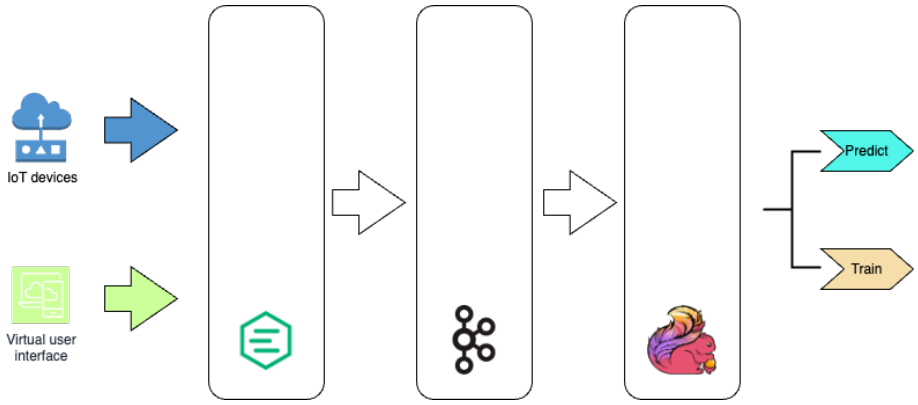
1. Παράθυρα πρόβλεψης και σύνολα χαρακτηριστικών ανά οντότητα για online inference.
2. Σύνολα εκπαίδευσης/επαλήθευσης/δοκιμής όταν διατίθενται δεδομένα κατάστασης συσκευών, ώστε να παράγεται έγκυρο ground truth για μάθηση.

Σε αυτό το κεφάλαιο παρουσιάζουμε την αρχιτεκτονική, τις τεχνολογίες, τους αλγόριθμους και τις βασικές λειτουργίες της πλατφόρμας, με ανάλυση από την συλλογή, την αποθήκευση έως τις στρατηγικές για την δημιουργία των datasets. Το αποτέλεσμα είναι ένα ουδέτερο ως προς το μοντέλο (model-agnostic) σύνολο δεδομένων, ικανό να υποστηρίξει τόσο την πρόβλεψη κατανομής φορτίων όσο και εκπαίδευση μοντέλων μηχανικής μάθησης

Αρχιτεκτονική Υποδομής

Η λύση μας έχει αρκετά επιμέρους στάδια. Ακολουθεί ανάλυση όλων των επιμέρους μονάδων με μια κατηγοριοποίηση ως προς την λειτουργική τους συμπεριφορά. Η λύση έχει ενοποιηθεί σε ένα cluster υπολογιστικών μονάδων που έχει ως κεντρική λειτουργική μονάδα το σύστημα Kubernetes και συγκεκριμένα το MicroK8s (open source). Στη διάταξη μας χρησιμοποιούμε 2 εικονικά μηχανήματα (8c/24GB) αλλά έχουμε την δυνατότητα να αυξήσουμε τον αριθμό των μηχανημάτων ανάλογα με τις απαιτήσεις του φόρτου εργασίας. Για την παραγωγική λειτουργία της λύσης μας συνοπτικά χρησιμοποιήθηκαν οι τεχνολογίες

- Apache Kafka Cluster (Strimzi Operator*, Open source*)
- Apache Kafka Bridge (Strimzi Operator*, Open source*)
- Kafka UI (Open Source project)
- EMQX (Helm, Emqx/Operator, *Open Source variant*)
- Apache Flink (Flink operator, *Open source*)
- Java 21 για το κώδικα προσομοίωσης των clients



Εικόνα 8:Οπτική αναπαράσταση της προτεινόμενης αρχιτεκτονικής

Παραμετροποίηση Συστήματος

Σε κάθε node εγκαταστήσαμε την σουίτα MicroK8s. Ταυτόχρονα προσαρμόζεται το περιβάλλον χρήσης ώστε να καταστεί δυνατή η απρόσκοπτη διαχείριση του Kubernetes. Ο πρώτος κόμβος αναλαμβάνει τον ρόλο του κεντρικού κόμβου και δημιουργεί ένα διακριτό κλειδί συμμετοχής, το οποίο αποτελεί τον μηχανισμό πιστοποίησης για την ένταξη πρόσθετων κόμβων στο σύστημα. Ο δεύτερος κόμβος, αξιοποιώντας το κλειδί αυτό, συνδέεται στο cluster και γίνεται μέλος του. Η επιτυχής ολοκλήρωση της διαδικασίας επιβεβαιώνεται από την ορατότητα και των δύο κόμβων στο ενιαίο περιβάλλον διαχείρισης.

Η σουίτα του Microk8s, υποστηρίζει addons, έτοιμα πακέτα υπηρεσιών για την διαχείριση της υποδομής. Ενεργοποιήσαμε τα παρακάτω addons:[30]

- cert-manager
- helm
- hostpath-storage
- dns
- Grafana
- Prometheus
- MetalLB

Ακολούθως, καθορίζεται ένα εύρος διαθέσιμων IP διευθύνσεων του τοπικού δικτύου, το οποίο δεν χρησιμοποιείται από άλλες συσκευές και μπορεί να δεσμευτεί αποκλειστικά για τις ανάγκες του Cluster. Το εύρος αυτό δηλώνεται ως pool διευθύνσεων στο αρχείο παραμετροποίησης του MetalLB, ώστε κάθε νέα υπηρεσία που ορίζεται με τύπο LoadBalancer να μπορεί να λάβει αυτόματα μια διεύθυνση από το προκαθορισμένο σύνολο.

Για την παραμετροποίηση της υποδομής EMQX cluster, χρησιμοποιήθηκε ο EMQX operator για την open source έκδοση του EMQX v5.7 μέσω Helm.

Η αρχιτεκτονική του EMQX διαρθρώνεται σε δύο επίπεδα, τον κεντρικό πυρήνα, ο οποίος αποτελείται από πολλαπλούς κόμβους που διαχειρίζονται την κύρια λογική του συστήματος,

και μια δευτερεύουσα ομάδα κόμβων που λειτουργούν ως επικουρικοί αναμεταδότες για την αποτελεσματική κατανομή του φόρτου. Ο ακριβής αριθμός των pods μπορεί να προσαρμόζεται δυναμικά βάσει των επιχειρησιακών αναγκών. Αυτή η προσέγγιση επιτρέπει στο σύστημα να εξυπηρετεί ταυτόχρονα αυξημένο αριθμό χρηστών, να επεκτείνεται ευέλικτα και να διατηρεί υψηλή διαθεσιμότητα ακόμη και σε περιπτώσεις αστοχίας μεμονωμένων κόμβων του Kubernetes Cluster.

Για τη διασφάλιση της σταθερής λειτουργίας, εφαρμόζουμε αυστηρούς περιορισμούς στη χρήση πόρων, όπως η μνήμη και επεξεργαστική ισχύ, ώστε να εξασφαλίζεται η προβλεψιμότητα κάθε κόμβου και η βέλτιστη κατανομή φόρτου στο cluster.

Στον τομέα της επικοινωνίας, το σύστημα διαμορφώνεται ώστε να υποστηρίζει τρία βασικά σημεία εισόδου: επικοινωνία μέσω TCP, WebSocket, καθώς και ένα κεντρικό web περιβάλλον διαχείρισης. Η αξιοποίηση του MetalLB διασφαλίζει ότι οι υπηρεσίες δημοσιοποιούνται με σταθερές διευθύνσεις IP, παρέχοντας στους πελάτες αξιόπιστη και απρόσκοπτη συνδεσιμότητα. [27]

Για την υλοποίηση του κεντρικού συστήματος αποθήκευσης της διάταξής μας επιλέχθηκε η τεχνολογία Apache Kafka, με χρήση της open source έκδοσης 3.6.0. Για τη βελτιστοποίηση της διαχείρισης αυτής της λειτουργικής μονάδας αξιοποιήθηκε ο Strimzi Operator (open source), ο οποίος χρησιμοποιήθηκε τόσο για τη διαχείριση του Kafka cluster όσο και για τη δημιουργία του Kafka bridge. Το Kafka Bridge επιτρέπει την αποτελεσματική επικοινωνία του EMQX με το Kafka Cluster μέσω του πρωτοκόλλου HTTP.

Όσον αφορά το υποσύστημα του Kafka cluster, παραμετροποιήσαμε το σύστημα ώστε σε κατάσταση ηρεμίας να έχουμε 3 κόμβους Kafka (listeners) και 3 κόμβους Zookeeper, ενεργοποιώντας μόνο τις δυνατότητες για επικοινωνία μέσα στο Kubernetes Cluster, όχι δηλαδή με εξωτερικά συστήματα.

Σε αυτό το Kafka Cluster ορίσαμε 4 βασικά topics τα οποία έχουν κατά κύριο λόγο από 10 έως 20 partitions το καθένα. Ο αριθμός αυτός μπορεί να αυξηθεί ή να μειωθεί κατά το δοκούν, ανάλογα πάντα με τις απαιτήσεις των δεδομένων που λαμβάνουμε. Τα topic αυτά έχουν τις εξής ονομασίες:

1. *power:* για την αποθήκευση των μετρήσεων ανά δευτερόλεπτο κάθε οντότητας
2. *device-state:* για την αποθήκευση της κατάστασης των συσκευών ανά δευτερόλεπτο
3. *predictions*: για την αποθήκευση των προβλέψεων ή σύνολα δεδομένων έτοιμα για πρόβλεψη για κάθε οντότητα.
4. *dataset2train*: σύνολα δεδομένων έτοιμα για εκπαίδευση μοντέλων μηχανικής μάθησης

Η συγκεκριμένη διάταξη προσφέρει ευελιξία στις ροές εισόδου, ανεξάρτητα από τον αριθμό των οντοτήτων που περιλαμβάνονται στην ανάλυση. Μια εναλλακτική μέθοδος θα προέβλεπε τη δημιουργία ενός topic παρόμοιου με τα προηγούμενα, αλλά ξεχωριστού για κάθε οντότητα. Παρόλο που δεν υφίσταται εμφανής περιορισμός στον αριθμό των topics σε ένα σύστημα Kafka, η εν λόγω προσέγγιση θα οδηγούσε σε αυξημένο διαχειριστικό κόστος, ιδιαίτερα ως προς τη δημιουργία και διαγραφή επιμέρους topics για οντότητες που είτε μόλις έχουν προστεθεί είτε δεν συμμετέχουν πλέον στην υπηρεσία.

Παρόλο που ο EMQX έχει την δυνατότητα να αποθηκεύσει προσωρινά τα δεδομένα, δεν αποθηκεύουμε τίποτα στους Brokers. Όλα τα δεδομένα προωθούνται στο υποσύστημα του Kafka. Θα περιγράψουμε τους μετασχηματισμούς που πραγματοποιούμε ώστε να προετοιμάσουμε τα δεδομένα μας για το επόμενο στάδιο, την αποθήκευσης για μεγαλύτερο χρονικό διάστημα. Κύριος στόχος είναι η διασφάλιση ότι τα δεδομένα μας είναι ομοιόμορφα και με όσο το δυνατόν ακριβέστερες χρονοσημάνσεις.

Για να μπορεί να επικοινωνήσει το υποσύστημα του EMQX με το υποσύστημα του Kafka, αξιοποιήθηκε ένας μηχανισμός γεφύρωσης (Kafka bridge), ο οποίος επιτρέπει τη μετατροπή των μηνυμάτων σε κατάλληλη μορφή και τη διοχέτευσή τους στο Kafka μέσω HTTP, εξασφαλίζοντας έτσι ομαλή διασύνδεση και διαλειτουργικότητα μεταξύ των δύο υποσυστημάτων.

Η Open source έκδοση του EMQX δεν μας δίνει την δυνατότητα να ορίσουμε αποδέκτη μηνυμάτων ένα Kafka topic, συνεπώς ορίσαμε κανόνες όπου διαβάζουν τα εισερχόμενα μηνύματα από αυτά τα δυο topic και φιλτράρουν τα δεδομένα μετασχηματίζοντας τα έτσι σε μια δομή (*application/vnd.Kafka.json.v2+json)* που είναι πλήρως συμβατή με το επόμενο στάδιο.

Τα δεδομένα μετατρέπονται από μορφή JSON ενσωματωμένη στο πρωτόκολλο MQTT σε HTTP, και τελικά διαμορφώνονται κατάλληλα ώστε να είναι συμβατά με την τεχνολογία Kafka. Ο Broker, για κάθε εισερχόμενο μήνυμα, εκτελεί POST αίτημα προς τη διεπαφή Kafka Bridge, διασφαλίζοντας την ομαλή ροή πληροφοριών μεταξύ των συστημάτων.

Για το υποσύστημα του Flink, αξιοποιήσαμε το Helm chart του Flink Operator για να εγκαταστήσουμε τον operator στο cluster μας. Με αυτόν τον τρόπο, έχουμε τη δυνατότητα να χρησιμοποιούμε τα custom resource definitions που προσφέρονται, διευκολύνοντας την εγγενή διαχείριση του Flink εντός του Kubernetes περιβάλλοντος. Με αυτόν τον τρόπο αποκτήσαμε τη δυνατότητα να περιγράψουμε και να αναπτύξουμε ένα session cluster του Flink απευθείας ως αντικείμενο του Kubernetes, απλοποιώντας σημαντικά τη διαδικασία. Η συγκεκριμένη διάταξη περιλαμβάνει έναν JobManager, ο οποίος έχει την ευθύνη του συντονισμού και της οργάνωσης των εργασιών καθώς και 2 TaskManagers, οι οποίοι εκτελούν τις ίδιες τις διεργασίες επεξεργασίας δεδομένων, αξιοποιώντας τα διαθέσιμα task slots για παράλληλη εκτέλεση.

Κατά την υλοποίηση διαπιστώσαμε ότι οι βασικές εικόνες του Flink δεν περιλάμβαναν τις απαραίτητες βιβλιοθήκες για άμεση διασύνδεση με το Kafka. Για το λόγο αυτό δημιουργήσαμε ένα προσαρμοσμένο container image, βασισμένο στην έκδοση 1.20 του Flink, στο οποίο ενσωματώσαμε τους σχετικούς connectors και εξαρτήσεις. Έτσι, το cluster μας ήταν σε θέση να επικοινωνεί απευθείας με το Kafka, να καταναλώνει ροές δεδομένων και να τις επεξεργάζεται χωρίς πρόσθετες παρεμβάσεις.

Μοντελοποίηση δεδομένων χρηστών

Κατηγοριοποίηση δεδομένων

Για την επίτευξη του στόχου μας συλλέγουμε τα παρακάτω δεδομένα:

1. Δεδομένα μέτρησης ολικής κατανάλωσης ισχύος ανά οντότητα.
2. Δεδομένα κατάστασης των επιμέρους συσκευών ανά οντότητα.

Στην δική μας μελέτη μια οντότητα αποτελεί μια οικία.

Δεδομένα μέτρησης ολικής κατανάλωσης ισχύος ανά οντότητα:

Ως δεδομένα μέτρησης ολικής κατανάλωσης ισχύος ανά οντότητα ορίζουμε την συνολική κατανάλωση ισχύος που περιλαμβάνει ένα υποσύνολο συσκευών που μπορούν να ομαδοποιηθούν. Συνεπώς με αυτή την μέτρηση συλλέγουμε την στιγμιαία κατανάλωση ισχύος για όλες τις συσκευές της οικίας.

Στο σημείο αυτό θα σημειώσουμε πως διακρίνουμε και μια ευρύτερη ομάδα, αυτή του χρήστη. Ένας χρήστης μπορεί να δίνει αναφορά/δεδομένα στο σύστημα μας για πολλαπλές οντότητες (οικίες στην προκυμμένη περίπτωση).

Δεδομένα κατάστασης των επιμέρους συσκευών ανά οντότητα:

Ως δεδομένα κατάστασης των επιμέρους συσκευών ανά οντότητα ορίζουμε μια αναπαράσταση της κατάστασης (διακριτή, ενεργοποιημένη/απενεργοποιημένη) κάθε επιμέρους συσκευής που μας είναι χρήσιμη για την οντότητα μας. Η μέθοδος αυτή μας δίνει την δυνατότητα να προσομοιώσουμε μια εφαρμογής χρήστη όπου, ο χρήστης παρέχει στο σύστημα μας την πληροφορία της κατάστασης των συσκευών του.

Αλγόριθμος για παραγωγή δεδομένων

Η παραγωγή δεδομένων γίνεται συνθετικά ανά οντότητα (οικία) ως εξής. Σε κάθε χρονικό βήμα δημιουργείται ή ενημερώνεται ένα δυαδικό διάνυσμα κατάστασης για ένα σταθερό σύνολο συσκευών (on/off), το οποίο ανανεώνεται περιοδικά με στοχαστικό ρυθμό, ώστε να σχηματίζονται τμήματα τυχαίας συμπεριφοράς. Η ολική κατανάλωση ισχύος προκύπτει ως γραμμικός συνδυασμός των ενεργών συσκευών (εσωτερικό γινόμενο του διανύσματος κατάστασης με συντελεστές κατανάλωσης ανά συσκευή) και δημοσιεύεται συνεχώς στο power topic. Παράλληλα, για την εικονική διεπαφή χρήστη που μας δίνει την κατάσταση των συσκευών δημιουργούμε εναλλάσσονται στοχαστικά παράθυρα με ή χωρίς δημοσίευση κατάστασης ώστε να δημιουργούνται labeled και unlabeled διαστήματα για εκπαίδευση/πρόβλεψη. Ο χρονισμός των δημοσιεύσεων ενσωματώνει τυχαία καθυστέρηση μεταξύ μετρήσεων και λαμβάνει υπόψιν μια μεταβλητή καθυστέρηση σαν χρόνο αποστολή, προσομοιώνοντας ασύγχρονες καθυστερήσεις δικτύου και ετεροχρονισμό συσκευών.

Πώς επιτυγχάνουμε την τυχαιότητα στα δεδομένα

Κάθε χρήστης πραγματοποιεί δημοσιεύσεις με τυχαία καθυστέρηση 1–3 δευτερολέπτων, ενώ κάθε μεμονωμένη ενέργεια δημοσίευσης συνοδεύεται από πρόσθετη μικροκαθυστέρηση 1–100 ms. Η συγκεκριμένη προσέγγιση προσομοιώνει ασύγχρονη λειτουργία καθώς και καθυστερήσεις του δικτύου.

1. Διαστήματα με ή χωρίς διαλείπουσα γνώση της κατάστασης των συσκευών: Εναλλασσόμενα χρονικά διαστήματα όπου δημοσιεύεται και το state μαζί με το power αλλά και τμήματα όπου δημοσιεύεται μόνο η συνολική κατανάλωση ισχύος. Η εναλλαγή γίνεται ανά 16–50 κύκλους (τυχαία).
2. Κατάσταση συσκευών: Το διάνυσμα κατάστασης ανανεώνεται στοχαστικά ανά 15 κύκλους (~30 s κατά μέσο όρο). Αυτό δίνει κομμάτια σχεδόν σταθερής συμπεριφοράς πάνω στα οποία παράγεται η ολική κατανάλωση, προσφέροντας χρήσιμες σταθερές περιόδους για εκτίμηση/εκπαίδευση.
3. Κλιμάκωση χρηστών: Νέοι χρήστες προστίθενται σταδιακά (batch) αύξηση με βάση τριών παραμέτρων (*increase_in_seconds*, *increase_users και total_users*), ώστε να ελέγχεται πώς συμπεριφέρεται η συσκευή μας με αύξηση του φόρου
4. Εισαγάγουμε τεχνική καθυστερημένων δημοσιεύσεων: 1 ζευγάρι τυχαίων διαδοχικών τιμών δημοσιεύονται στο EMQX σε αντίστροφη σειρά.

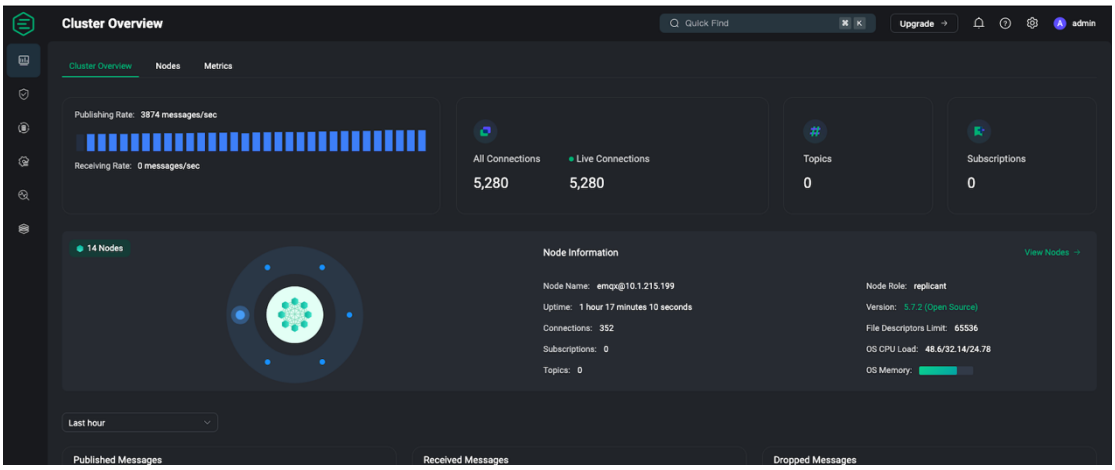
Συλλογή Δεδομένων

Διασύνδεση με τον Broker

Σε αυτό το σημείο θα εστιάσουμε στις λειτουργίες της αυθεντικοποίησης και της αποστολής δεδομένων από τους πελάτες στο κεντρικό σημείο διαχείρισης των μηνυμάτων (EMQX Broker). Το σύστημα υποστηρίζει πολλαπλούς ξεχωριστούς πελάτες (clients), οι οποίοι

συνδέονται ταυτόχρονα στον Broker και επικοινωνούν μέσω MQTT για να δημοσιεύσουν δεδομένα σε προκαθορισμένα θέματα (topics). Οι πελάτες αυτοί λειτουργούν ανεξάρτητα και αλληλοεπιδρούνε με συγκεκριμένα topics που σχετίζονται με την κατανάλωση ενέργειας και την κατάσταση λειτουργίας συσκευών.

Όπως αναφέραμε, τα μηνύματα που αποστέλλονται στον Broker δεν διαχωρίζονταν όσον αφορά την παρουσία ή όχι του πεδίου χρονοσήμανσης (timestamp) της μέτρησης. Αυτό σημαίνει ότι ο Broker έπρεπε να διαχειρίζεται όλα τα μηνύματα με τον ίδιο τρόπο, ανεξάρτητα από το αν περιείχαν ή όχι χρονική σήμανση.



Εικόνα 9: EMQX Overview dashboard

Αυθεντικοποίηση στον EMQX Broker

Ο EMQX Broker διατηρεί μια βάση δεδομένων για να αποθηκεύει τους χρήστες και τους κωδικούς πρόσβασης. Όταν ένας πελάτης (user) προσπαθεί να συνδεθεί, ο Broker ελέγχει τα στοιχεία του (username και password) στη βάση δεδομένων. Αν τα στοιχεία είναι σωστά, ο χρήστης συνδέεται και μπορεί να δημοσιεύει ή να λαμβάνει μηνύματα. Επιπλέον, ο μηχανισμός του Broker περιλαμβάνει και έλεγχο εξουσιοδότησης (authorization), μέσω του οποίου καθορίζεται σε ποια topics έχει πρόσβαση κάθε χρήστης, στη δική μας υλοποίηση όλοι οι χρήστες έχουν εξουσιοδοτηθεί να εκτελούν πράξεις publish στα topics *power* και *device-state*.

Οι κωδικοί αποθηκεύονται με ασφάλεια (κρυπτογραφημένοι) και ο Broker μπορεί να ορίσει δικαιώματα πρόσβασης για κάθε χρήστη, καθορίζοντας ποια topics μπορεί να χρησιμοποιήσει. Επίσης, μέσω του EMQX dashboard, ο διαχειριστής μπορεί να προσθέτει ή να αφαιρεί χρήστες και να βλέπει ποιοι είναι ενεργοί.

Αποστολή δεδομένων

Οι clients δημοσιεύουν δεδομένα σε προκαθορισμένα topics του EMQX που αφορούν την κατανάλωση ενέργειας και την κατάσταση των συσκευών τους. Τα δεδομένα δημοσιεύονται σε μορφή JSON, με την κατανάλωση να αποστέλλεται ως τιμή του κλειδιού *consumption* απο το topic *power* και την κατάσταση των συσκευών μέσω μιας λίστας τιμών αληθείας *state_bitmap* απο το topic *state*.

Κάθε πελάτης κατά τη δημοσίευση ενός συμβάντος του, ορίζει μεταξύ άλλων τα εξής πεδία που αξίζει να σημειωθούν (τα οποία ενσωματώνονται στα μεταδεδομένα του μηνύματος)

- client_id (το όνομα της συσκευής)
- username (το όνομα του χρήστη)
- timestamp (την χρονική στιγμή της μετρησης)
- consumption | state_bitmap (key-value, αναλογα με το topic)

Η επικοινωνία πραγματοποιείται μέσω του πρωτοκόλλου MQTT v5, το οποίο εξασφαλίζει χαμηλό latency, συνεχή σύνδεση και ελάχιστη κατανάλωση πόρων, καθιστώντας το ιδανικό για εφαρμογές που απαιτούν άμεση μετάδοση δεδομένων σε πραγματικό χρόνο. Για τη μετάδοση χρησιμοποιείται το επίπεδο QoS 0 (Quality of Service 0), το οποίο αντιστοιχεί σε μετάδοση τύπου “at most once”. Σε αυτήν τη ρύθμιση, τα μηνύματα αποστέλλονται χωρίς μηχανισμό επιβεβαίωσης παραλαβής, επιτυγχάνοντας τη μέγιστη ταχύτητα επικοινωνίας με ελάχιστο δικτυακό overhead. Αν και ενδέχεται να υπάρξουν απώλειες μεμονωμένων πακέτων σε συνθήκες αστάθειας, το QoS 0 αποτελεί την πιο αποδοτική επιλογή για συνεχή ροή τηλεμετρικών δεδομένων, όπου η έμφαση δίνεται στη διαμεταγωγή (throughput) και όχι στην απόλυτη αξιοπιστία παράδοσης.

Επεξεργασία στον Broker

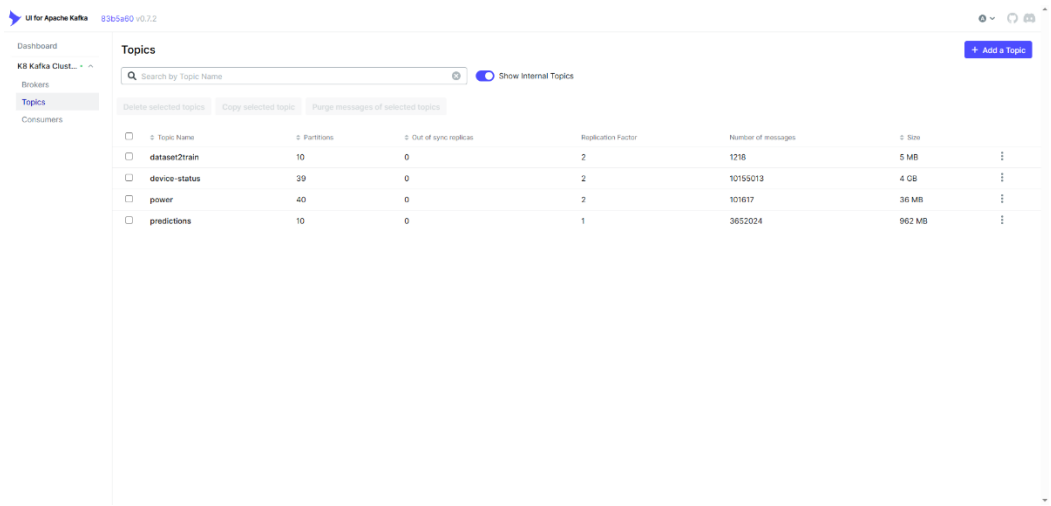
Μία από τις παθογένειες πολλών συσκευών IOT είναι ότι πολλές φορές δεν συντηρούν σύστημα RTC. Αυτό έχει ως αποτέλεσμα να φτάνουν δεδομένα στον Broker μας με ή χωρίς καθορισμένο το πεδίο timestamp. Η λύση που προτείνουμε την υποδομή μας, κατά την επεξεργασία στον Broker μέσω διαδικασιών ελέγχουμε αν το timestamp value είναι κενό και σε αυτή την περίπτωση και χάριν απλότητας, χρησιμοποιούμε και ενημερώνουμε το event με την χρονοσήμανση που το μήνυμα έφτασε στον Broker

Σε αυτό το στάδιο τα δεδομένα κάθε χρήστη για κάθε οντότητα δημοσιεύονται στα δυο επιμέρους topics

Αποθήκευση στο Kafka

Πριν την αποθήκευσης στο Kafka μεσολαβεί το στάδιο του Kafka Bridge, Αυτό το στάδιο δεν είναι άλλο από έναν μεσολαβητή που ξέρει να πάρει ένα request μέσω του πρωτοκόλλου HTTP και χωρίς εμείς να γνωρίζουμε πως πρέπει να μιλήσουμε στο Kafka, να μπορέσει να το μετατρέψει σε ένα event μέσα σε αυτό.

Συνεπώς πολλαπλοί EMQX Brokers στέλνουν τα μηνύματα τους σε πολλαπλούς HTTP server του Kafka Bridge και αυτοί με τη σειρά τους τα μεταφέρουν στα topics του Kafka.



Εικόνα 10:Kafka UI dashboard - list topics

Για τις ανάγκες της παρούσας εργασίας όπως έχουμε ήδη αναφέρει, έχουμε δημιουργήσει 4 topic, όπου τα 2 πρώτα (power, predictions) κρατούν την πληροφορία για την ολική κατανάλωση ισχύος και την κατάσταση των συσκευών για όλους τους πελάτες. Ο αριθμός των partition του κάθε topic είναι ενδεικτικός αλλά άρρηκτα συνδεδεμένος με το επόμενο στάδιο του Flink, ώστε να εξετάσουμε την κλιμακωσιμότητα του συστήματος μας.

Τα δύο επόμενα Topics (dataset2train, predictions) είναι βοηθητικά για να μας δώσει την εικόνα των δεδομένων που λαμβάνουμε από το Flink αφού εκτελέσουμε τις διαδικασίες μας.

Εδώ θα γίνει πλήρως αντιληπτό γιατί εισαγάγουμε το στάδιο του Flink. Όπως αναφέραμε στην προηγούμενη υποενότητα, έχουμε 2 ροές δεδομένων τα οποία έχουν τα εξής χαρακτηριστικά:

1. Lateness
   Τα γεγονότα δεν φτάνουν στον Broker απαραίτητα με την ακριβή χρονική σειρά παραγωγής τους, λόγω δικτυακών καθυστερήσεων ή άλλων παραγόντων, μπορεί ένα γεγονός που δημιουργήθηκε νωρίτερα να φτάσει μετά από ένα νεότερο.

2. Out-of-orderness
   Προκύπτει κυρίως από το γεγονός ότι η ανάγνωση από πολλά partitions οδηγεί σε ανάμειξη της σειράς των γεγονότων

3. Multitenancy
   Τα δεδομένα στα topic μας προέρχονται από πολλαπλές οντότητες. Αυτό απαιτεί την ομαδοποίηση και διαχωρισμό των δεδομένων με βάση το αναγνωριστικό κάθε οντότητας (client_id), έτσι ώστε κάθε πελάτης να επεξεργάζεται και να αναλύεται ανεξάρτητα.

4. Ειναι διακοπτόμενες (*device-state)*
   Τα δεδομένα σχετικά με την κατάσταση των συσκευών (device-state) δεν παράγονται συνεχώς, αλλά εμφανίζονται σε διαστήματα – είναι διακοπτόμενα. Δηλαδή, κάποια χρονικά διαστήματα υπάρχουν δεδομένα που περιγράφουν την κατάσταση ενώ διαστήματα δεν υπάρχει τέτοια πληροφορία.

Σκοπός μας είναι μέσω των δυο ροών δεδομένων που θα αναπτύξουμε στις επόμενες παραγράφους να δημιουργήσουμε πλούσια δεδομένα για την πρόβλεψη και για την εκπαίδευση μοντέλων μηχανικής μάθησης για τον διαχωρισμό συσκευών.

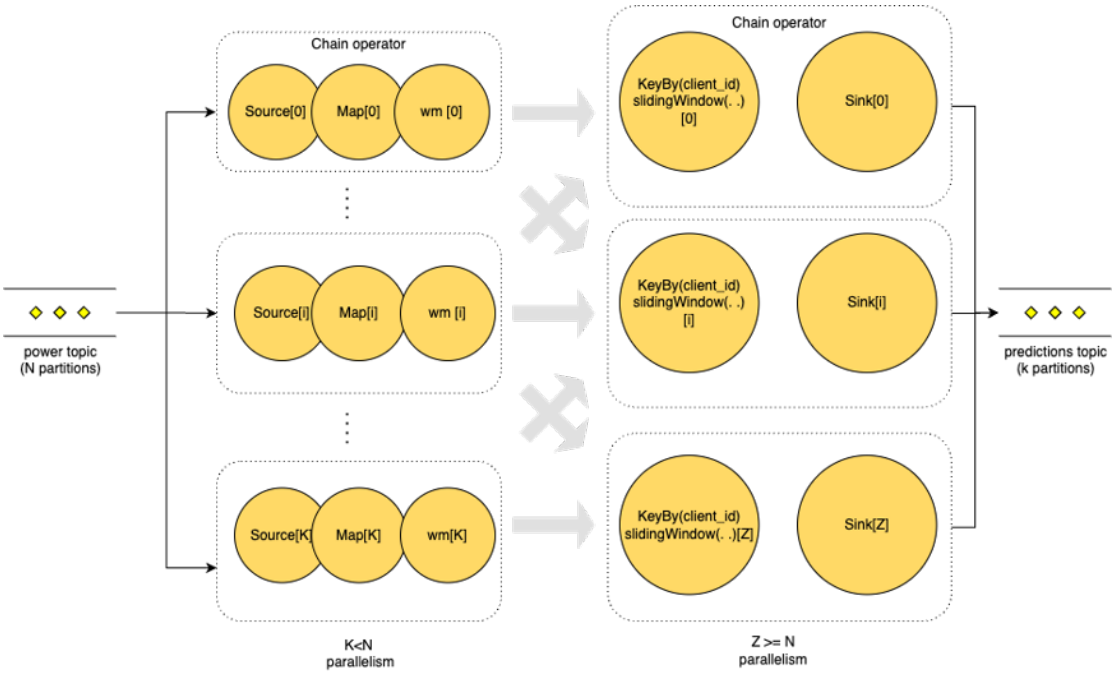
Ροές δεδομένων

Ροή δεδομένων για πρόβλεψη.

Στόχος μας είναι η ζωντανή πρόβλεψη κατανάλωσης ανά πελάτη και για αυτό υλοποιήσαμε έναν αλγόριθμο σε Java που τρέχει ως job στο Flink, λαμβάνει τα γεγονότα κατανάλωσης, τα ευθυγραμμίζει χρονικά με βάση τον χρόνο συμβάντος (χειριζόμενο lateness/out-of-orderness με μικρή ανοχή) και τα ομαδοποιεί ανά client_id ώστε κάθε πελάτης να επεξεργάζεται ανεξάρτητα και παράλληλα πάνω σε αυτό το stream. Χρησιμοποιούμε παράθυρα ολίσθησης (sliding windows) διάρκεις 15s που έχουν βήμα 1s για να παράγουμε συνεκτικά, βραχυπρόθεσμα στιγμιότυπα ώστε να τρέξουν πάνω τους αλγόριθμοι μηχανικής μάθησης, τα οποία στη δική μας περίπτωση τα επιστρέφουμε πίσω στην διάταξη του Kafka.

Αλγόριθμος:

- Διαβάζουμε γεγονότα κατανάλωσης και τα φέρνουμε όλα στην ίδια χρονική κλίμακα (sec), δουλεύοντας σε χρόνο γεγονότος.
- Δίνουμε μικρή ανοχή καθυστέρησης για late/out-of-order αφίξεις ώστε κάθε γεγονός να συμπεριλαμβάνεται στο σωστό σημείο της χρονοσειράς.
- Ομαδοποιούμε ανά οντότητα (client_id) ώστε κάθε πελάτης να επεξεργάζεται ανεξάρτητα.
- Δημιουργούμε παράθυρα ολίσθησης διάρκειας 15s με βήμα 1s και συνθέτουμε σύντομα, συνεκτικά στιγμιότυπα των τελευταίων T δευτερολέπτων.
- Αποθηκεύουμε το αποτέλεσμα (σε μορφή JSON) πίσω στο Kafka με κλειδί το client_id.

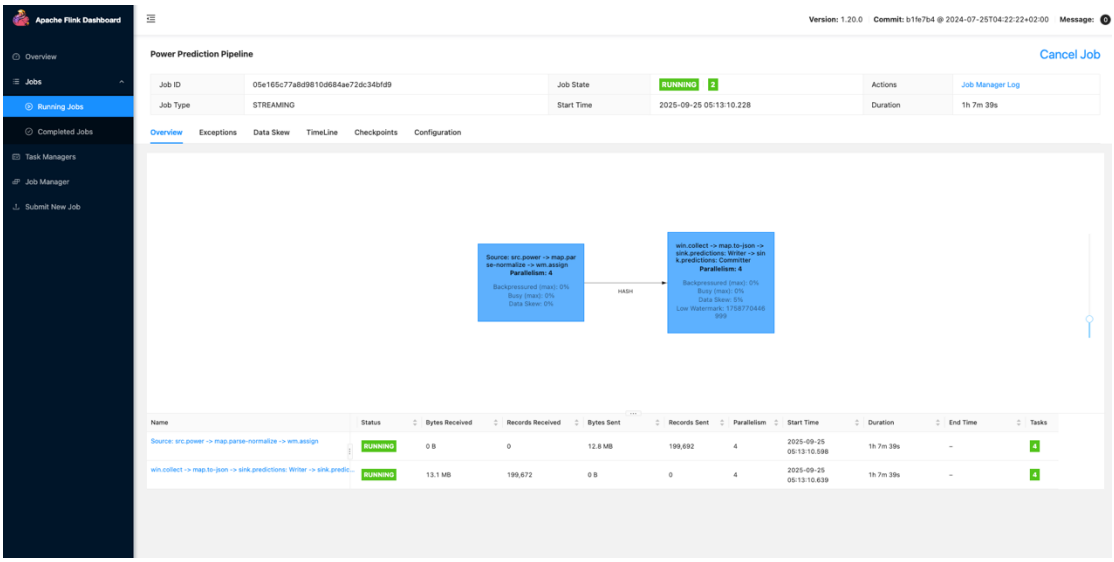


Εικόνα 11: Σχεδιάγραμμα ροής (Ροή πρόβλεψης)

Εδώ πρέπει να σημειωθεί ότι σκοπίμως χρησιμοποιήσαμε τον ίδιο βαθμό παραλληλισμού σε όλα τα στάδια της ροής. Αν αποφασίζαμε να αυξήσουμε τον βαθμό παραλληλισμού μόνο στο στάδιο του Sink, τότε ο συγκεκριμένος operator θα εκτελούνταν με περισσότερα subtasks, που ναι μεν θα μας έδινε μεγαλύτερο βαθμό παραλληλισμού, αλλά δεν θα μπορούσε πλέον να συνενωθεί (chain) με το προηγούμενο στάδιο, καθώς η διαφορά στον parallelism θα ανάγκαζαν το Flink να τα εκτελέσει ως ξεχωριστά task groups.

Η επιλογή ίδιου βαθμού παραλληλισμού σε όλα τα επίπεδα έχει δύο βασικά πλεονεκτήματα:

1. Απλοποιεί τον σχεδιασμό και την παρακολούθηση του job, η εκτέλεση παραμένει συμμετρική και εύκολα αναγνώσιμη στο UI.
2. Επιτρέπει στο Flink να εφαρμόσει operator chaining, δηλαδή να ενώσει διαδοχικούς operators σε έναν κοινό task. Έτσι μειώνονται οι περιττές επικοινωνίες μέσω δικτύου, με αποτέλεσμα καλύτερη απόδοση και χαμηλότερη καθυστέρηση.



Εικόνα 12: Flink UI - εκτέλεση ροής πρόβλεψης

Παραλληλισμός και πώς τον επιτυγχάνεται:

- Τα δεδομένα μοιράζονται και διαβάζονται από και σε πολλαπλά Kafka partitions ώστε η ροή να κλιμακώνει οριζόντια.
- Το κλειδί client_id κατευθύνει *όλα* τα γεγονότα του ίδιου πελάτη στο ίδιο υπό-task, διατηρώντας σειρά και την τοπικότητα της κατάστασης, διαφορετικοί πελάτες εκτελούνται ταυτόχρονα σε άλλα υπό-tasks.
- Τα sliding windows υπολογίζονται ανά κλειδί και έτσι κατανέμονται φυσικά σε όλα τα υπό-tasks, χωρίς συγχρονισμούς μεταξύ πελατών.
- Η έξοδος με μεσω του κλειδιού client_id κατανέμει τις προβλέψεις σε πολλαπλά Kafka partitions, επιτρέποντας παράλληλες εγγραφές.
- Αυξάνοντας τον βαθμό παραλληλισμού (και τα αντίστοιχα partitions) παίρνουμε σχεδόν γραμμική κλιμάκωση, αρκεί τα client_id να είναι αρκετά και ομοιόμορφα κατανεμημένα. Σε περίπτωση που έχουμε “hot” keys (συγκεκριμένοι και ίδιοι πελάτες χρησιμοποιούν συνέχεια την υποδομή μας), εφαρμόζουμε τεχνικές εξισορρόπησης (όπως το hash-bucketing του client_id).

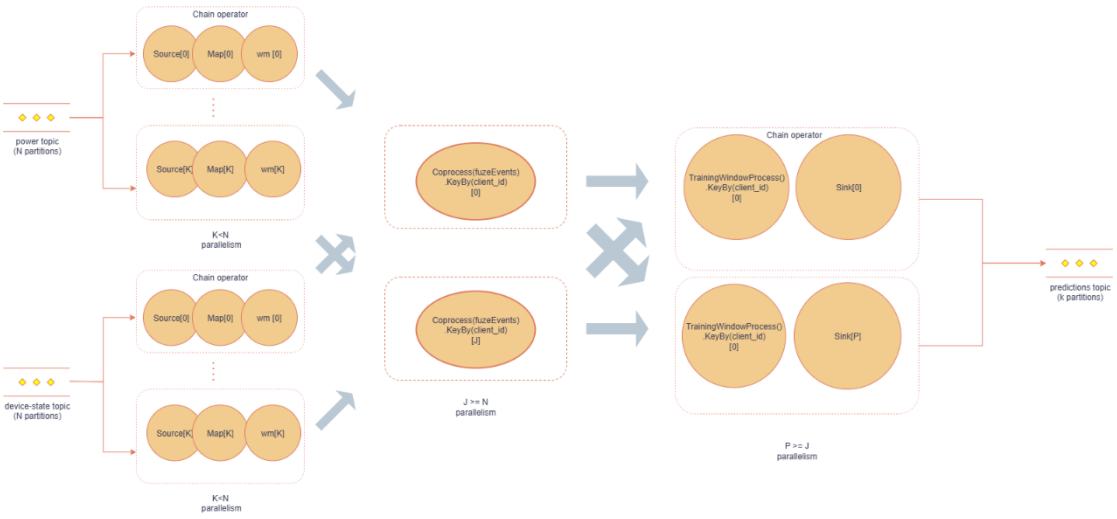
Ροή δεδομένων για εκπαίδευση μοντέλων.

Σκοπός μας είναι η δημιουργία καθαρών training datasets διαβάζοντας τα δύο topics του Kafka (κατανάλωση και κατάσταση συσκευών). Για το λόγο αυτό ευθυγραμμίζουμε τα γεγονότα στον πραγματικό χρόνο και ενοποιούμε τις δύο ροές για κάθε πελάτη. Εντοπίζουμε διαστήματα όπου συνυπάρχουν και οι δύο πληροφορίες και σε αυτά με ένα session-based window που ανοίγει με το πρώτο σχετικό event (δεδομένα κατάστασης) και κλείνει όταν υπάρξει μικρή αδράνεια δεδομένων κατάστασης. Συνοπτικά, συγκεντρώνουμε συνεχόμενα γεγονότα και αν το τελικό απόσπασμα είναι ≥30s επιδιώκουμε να κρατήσουμε το πλουσιότερο δυνατό

συνεκτικό κομμάτι (μεγαλύτερη διάρκεια/πληρότητα), το *«κλειδώνουμε»* ως window και το προωθούμε ως υποψήφιο για εκπαίδευση (πίσω στο Kafka).

Αλγόριθμος:

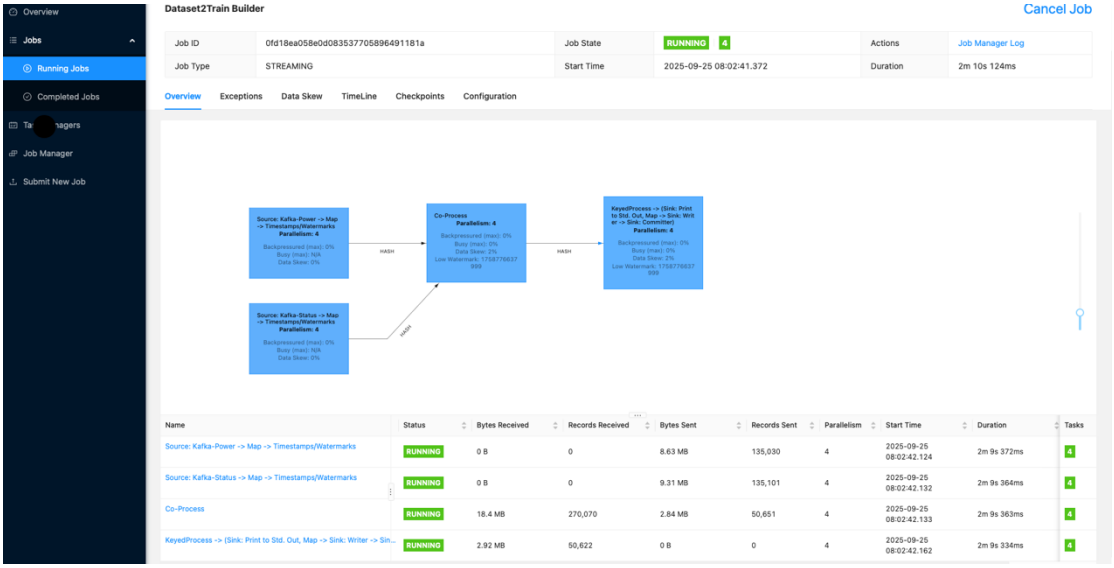
- Διαβάζουμε δύο topic από το Kafka (power & device-status) και φέρνουμε τα timestamps στην ίδια κλίμακα (δευτερόλεπτα), δουλεύοντας σε event time με μικρή ανοχή για late/out-of-order αφίξεις.
- Ομαδοποιούμε ανά πελάτη (client_id) ώστε κάθε πελάτης να επεξεργάζεται ανεξάρτητα.
- Συγχωνεύουμε ανά πελάτη τα δύο ρεύματα στο ίδιο δευτερόλεπτο (fused event: power + state), κρατώντας τοπικό state για τα ταιριάσματα.
- Χτίζουμε session-based windows, ανοίγει με το πρώτο fused event και κλείνει μετά από ~5s αδράνειας. Αν η διάρκεια του window είναι ≥30s, το κρατάμε
- Τα παραγόμενα Dataset Windows τα στέλνουμε ως JSON πίσω στο Kafka στο topic datasets2train με κλειδί το client_id.



Εικόνα 13: Σχεδιάγραμμα ροής (Ροή εκπαίδευσης)

Παραλληλισμός και πώς επιτυγχάνεται:

- Τα δεδομένα μοιράζονται σε πολλαπλά Kafka partitions, δίνοντας πολλαπλά παράλληλα tasks σε source/sink.
- Το client_id κατευθύνει όλα τα γεγονότα του ίδιου πελάτη στον ίδιο subtask, ενώ διαφορετικοί πελάτες εκτελούνται ταυτόχρονα σε αλλα subtasks.
- Το στάδιο συνένωσης των ροών (Co-Process) και τα session windows δουλεύουν per key με τοπικό state/timers, χωρίς συγχρονισμούς μεταξύ πελατών.
- Η έξοδος (με key το εκάστοτε client_id) κατανέμει τα windows σε πολλές Kafka partitions, επιτρέποντας παράλληλες εγγραφές και σταθερή σειρά per client.
- Αυξάνοντας τον παραλληλισμό στα στάδια του Flink και τα αντίστοιχα partitions πετυχαίνουμε σχεδόν γραμμική κλιμάκωση. Για τυχόν hot keys εφαρμόζουμε τεχνικές εξισορρόπησης (π.χ. bucketing του client_id).



Εικόνα 14: Flink UI - εκτέλεση ροής εκπαίδευσης

Παρατηρησιμότητα συστήματος

Για την παρακολούθηση της λειτουργίας της πλατφόρμας, αναπτύξαμε έναν μηχανισμό βασισμένο στον συνδυασμό *Prometheus* για τη συλλογή μετρικών και *Grafana* για την οπτικοποίησή τους. Οι μετρικές αποθηκεύονται, επεξεργάζονται και παρουσιάζονται σε ένα ενιαίο περιβάλλον παρακολούθησης, ανάλυσης απόδοσης και εντοπισμού προβλημάτων σε πραγματικό χρόνο, προσφέροντας πλήρη εικόνα της λειτουργικής κατάστασης του συστήματος.

Η μετρικές που παρακολουθούμε ενδεικτικά είναι:

1. Ο αριθμός των event που φτάνουν στους Broker
2. Ο αριθμός των event που φτάνουν από τους Broker στα input topics τους Kafka

3. Ανισοκατανομή στα δεδομένα των partition (Kafka) και των επιμέρους task των Ροών (Flink)
4. Καθυστέρηση επικοινωνίας μεταξύ των Kafka Nodes
5. Ποσοστό χρήσης σε πόρους όπως η μνήμη και ο χρόνος επεξεργασίας (σε όλα τα στάδια, EMQX cluster, Kafka Cluster, Kafka Bridge)
6. Καθυστέρηση για εγγραφή συμβάντων στο Kafka
7. Ισοτιμία αριθμού συμβάντων τόσο μεταξύ των επιμέρους task του flink όσο και συμβάντων που παράχθηκαν από τα τελικά στάδια του Flink αλλά δεν έφτασαν ποτέ στο Kafka

3. Πειραματική διάταξη και αποτελέσματα

Χρησιμοποιώντας τη μοντελοποίηση σε επίπεδο μεμονωμένου χρήστη, σχεδιάστηκε και υλοποιήθηκε ένα σύνολο πειραμάτων με στόχο την αξιολόγηση της συνολικής απόδοσης και αξιοπιστίας του συστήματος υπό συνθήκες αυξανόμενου φόρτου. Το πείραμα εστιάζει στη μελέτη δύο κρίσιμων πτυχών της λειτουργίας της πλατφόρμας:

1. *Έλεγχος κλιμακωσιμότητας*

   Σκοπός του ελέγχου είναι να εκτιμηθεί η ικανότητα του συστήματος να διατηρεί αποδεκτά επίπεδα απόδοσης καθώς αυξάνεται ο αριθμός των ταυτόχρονων χρηστών και η ροή των δεδομένων. Η αξιολόγηση πραγματοποιείται μέσω σταδιακής αύξησης του αριθμού ενεργών χρηστών, προσομοιώνοντας σενάρια επέκτασης. Κατά τη διάρκεια του πειράματος παρακολουθούνται δείκτες όπως ο χρόνος απόκρισης, η κατανάλωση πόρων και η σταθερότητα των ροών δεδομένων, με στόχο τον εντοπισμό πιθανών σημείων συμφόρησης (bottlenecks).

2. *Έλεγχος ορθότητας*

   Ο δεύτερος άξονας της αξιολόγησης αφορά την ακρίβεια και συνοχή των παραγόμενων δεδομένων. Για τον σκοπό αυτό επιλέχθηκαν δύο τυχαίοι χρήστες, επί των οποίων πραγματοποιείται λεπτομερής έλεγχος των εξής παραμέτρων:

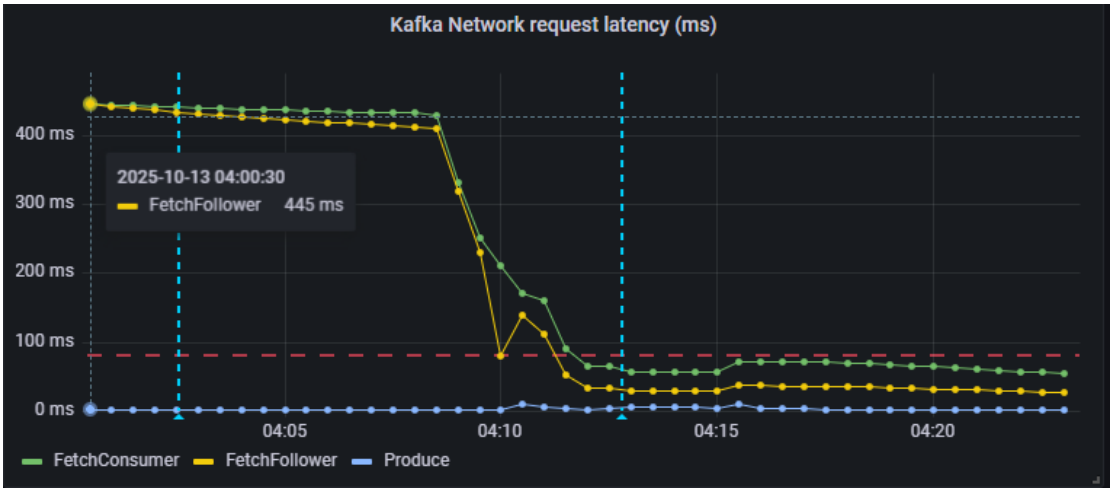
   - Η αντιστοίχιση των μετρήσεων ενεργειακής κατανάλωσης και της κατάστασης των συσκευών εντός κάθε παραθύρου εκπαίδευσης
   - Ο αριθμός και η πληρότητα των παραγόμενων συνόλων δεδομένων.
   - Η χρονική ορθότητα της ακολουθίας των δεδομένων, διασφαλίζοντας ότι οι χρονικές σημάνσεις παραμένουν σε αύξουσα σειρά χωρίς απώλειες ή επικάλυψη.

Η συνδυασμένη αξιολόγηση των δύο αυτών παραμέτρων επιτρέπει τη συνολική αποτίμηση της πλατφόρμας τόσο σε επίπεδο λειτουργικής επάρκειας και επεκτασιμότητας, όσο και σε επίπεδο αξιοπιστίας και ορθότητας των αποτελεσμάτων. Μέσω της πειραματικής αυτής διαδικασίας, επιδιώκεται η αναγνώριση των περιορισμών του συστήματος και η τεκμηρίωση της συμπεριφοράς του σε συνθήκες πραγματικού χρόνου.

Έλεγχος Κλιμακωσιμότητας

Η διαδικασία αξιολόγησης της κλιμακωσιμότητας ξεκινά από μία βάση αναφοράς με 100 ενεργούς χρήστες και εξελίσσεται με γραμμική προοδευτική αύξησης του πλήθους των χρηστών κατά 100 άτομα ανά 5 δευτερόλεπτα. Ορίζουμε ως ανώτατο κατώφλι χρηστών τους 4000 και το συνολικό διάστημα παρατήρησης τη μία ώρα. Κάθε στάδιο αύξησης συνοδεύεται από διάστημα παρατήρησης, κατά το οποίο καταγράφονται οι βασικοί δείκτες απόδοσης του συστήματος (latency, throughput, CPU και memory utilization, parity εισόδου/εξόδου). Η διαδικασία αυτή επιτρέπει τη λεπτομερή παρακολούθηση της συμπεριφοράς της πλατφόρμας σε συνθήκες αυξανόμενου φόρτου και την ταυτοποίηση κρίσιμων σημείων συμφόρησης.

Καθώς ο αριθμός των χρηστών αυξάνεται, η πρώτη σημαντική επιβράδυνση εντοπίζεται στο υποσύστημα μηνυμάτων Kafka. Παρατηρείται αύξηση του latency μεταξύ των nodes του Kafka, η οποία υποδηλώνει ότι οι διαθέσιμοι πόροι επεξεργασίας και επικοινωνίας δεν επαρκούν για τον ρυθμό εισερχόμενων δεδομένων.



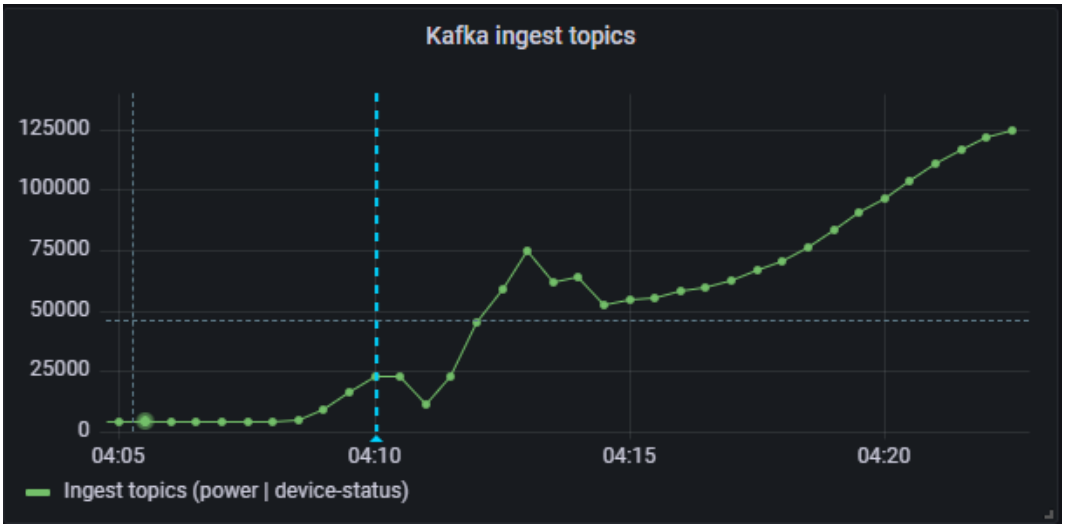
Εικόνα 15:Kafka Nodes network latency before and after the scale up

Για την αντιμετώπιση του φαινομένου, προχωρήσαμε σε διπλασιασμό των υπολογιστικών πόρων που διατίθενται στα Kafka Nodes (CPU και memory). Μετά την κλιμάκωση αυτή, το latency σταθεροποιήθηκε σε αποδεκτά επίπεδα και το throughput επέστρεψε στα αρχικά επίπεδα απόδοσης.

Σε επόμενο στάδιο φόρτου, παρατηρείται μείωση του αριθμού των γεγονότων (events) που καταλήγουν από τον EMQX Broker στο Kafka. Η συμπεριφορά αυτή υποδεικνύει απώλεια

μηνυμάτων κατά τη διαδικασία μεταφοράς, πιθανότατα λόγω συμφόρησης στο Kafka Bridge, το οποίο λειτουργεί ως ενδιάμεσος μηχανισμός δρομολόγησης.

Εικόνα 16:EMQX Published message, before and after the Kafka bridge scale-out

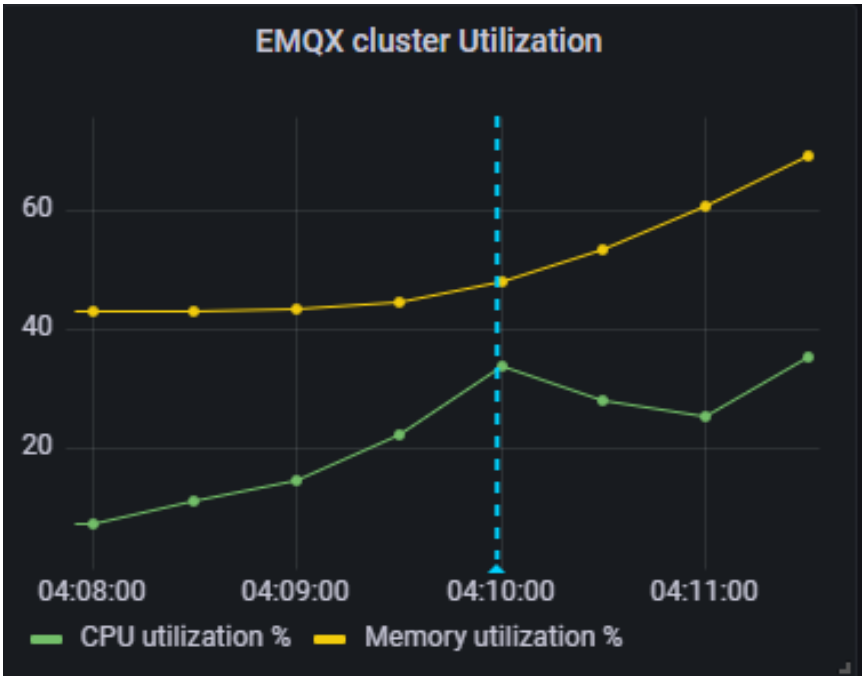


Εικόνα 17: Kafka input topics, before and after the Kafka bridge scale-out

Η ανάλυση των logs έδειξε επίσης ότι τα pods του Kafka Bridge εκτελούσαν επανεκκινήσεις, γεγονός που επιβεβαίωσε την ύπαρξη πίεσης στους διαθέσιμους πόρους. Για την αποκατάσταση της σταθερότητας, πραγματοποιήθηκε διπλασιασμός των πόρων του Kafka

Bridge, με αποτέλεσμα την εξάλειψη των απωλειών και τη σταθεροποίηση της ροής δεδομένων.

Με περαιτέρω αύξηση του αριθμού χρηστών, καταγράφηκε έντονη αύξηση της χρήσης CPU και μνήμης στον EMQX Broker, γεγονός που υποδήλωνε ότι πλησίαζε τα λειτουργικά του όρια.



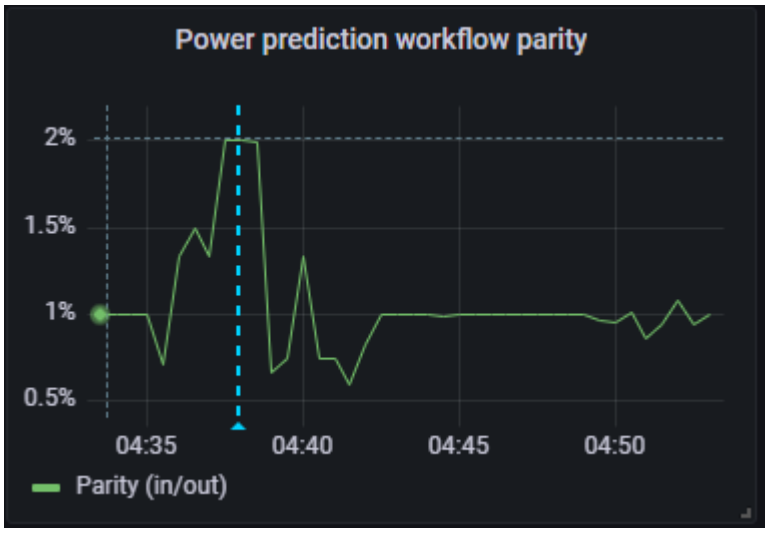
Εικόνα 18: EMQX resource utilization

Προληπτικά, πριν εμφανιστεί πτώση στην απόδοση ή απώλεια συνδέσεων, αποφασίστηκε ο διπλασιασμός του πλήθους των pods του EMQX, εξασφαλίζοντας έτσι καλύτερη κατανομή φόρτου και διατήρηση σταθερής απόδοσης στο σύνολο του πειράματος.

Κατά την περαιτέρω αύξηση του φόρτου, εντοπίστηκε διαφοροποίηση στην ισοτιμία αριθμού συμβάντων (event parity) μεταξύ των ενδιάμεσων σταδίων επεξεργασίας του Flink και του τελικού σταδίου εξόδου προς το Kafka. Συγκεκριμένα, στη ροή επεξεργασίας Power Prediction, παρατηρήθηκε ότι ο αριθμός των εισερχόμενων γεγονότων προς τα ενδιάμεσα tasks του Flink διαφέρει από τον αριθμό των εξερχόμενων, γεγονός που υποδηλώνει απώλειες κατά τη φάση επεξεργασίας ή περιορισμένη δυνατότητα εξυπηρέτησης ορισμένων tasks.

Η ανάλυση των μετρικών έδειξε ότι η διαφορά μεταξύ in/out event rate οφείλεται σε ανισοκατανομή φόρτου μεταξύ των επιμέρους task, καθώς ορισμένα εκτελούν επεξεργασία

σημαντικά βαρύτερων τμημάτων δεδομένων. Το φαινόμενο αυτό οδηγεί σε τοπικές καθυστερήσεις και προσωρινή απώλεια δεδομένων από το stream pipeline.

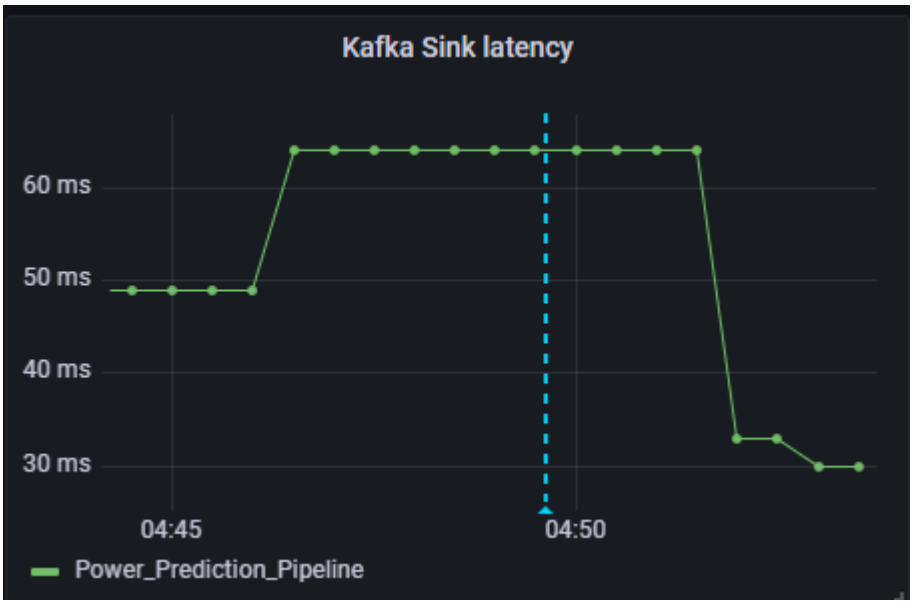


Εικόνα 19: Snapshot of Flink's internal tasks event parity

Για την αντιμετώπιση του προβλήματος, προχωρήσαμε σε διπλασιασμό του επιπέδου παραλληλισμού της ροής Power Prediction, επιτρέποντας την ταυτόχρονη εκτέλεση περισσότερων task και συνεπώς καλύτερη εξισορρόπηση του φόρτου επεξεργασίας. Η παρέμβαση αυτή είχε ως αποτέλεσμα τη σταθεροποίηση του ρυθμού παραγωγής συμβάντων και την αποκατάσταση της ισοτιμίας in/out μεταξύ των ενδιάμεσων σταδίων και του τελικού Kafka sink.

Μετά την αύξηση του επιπέδου παραλληλισμού της ροής Power Prediction, παρατηρήθηκε αξιοσημείωτη βελτίωση του χρόνου απόκρισης στα ενδιάμεσα στάδια επεξεργασίας.

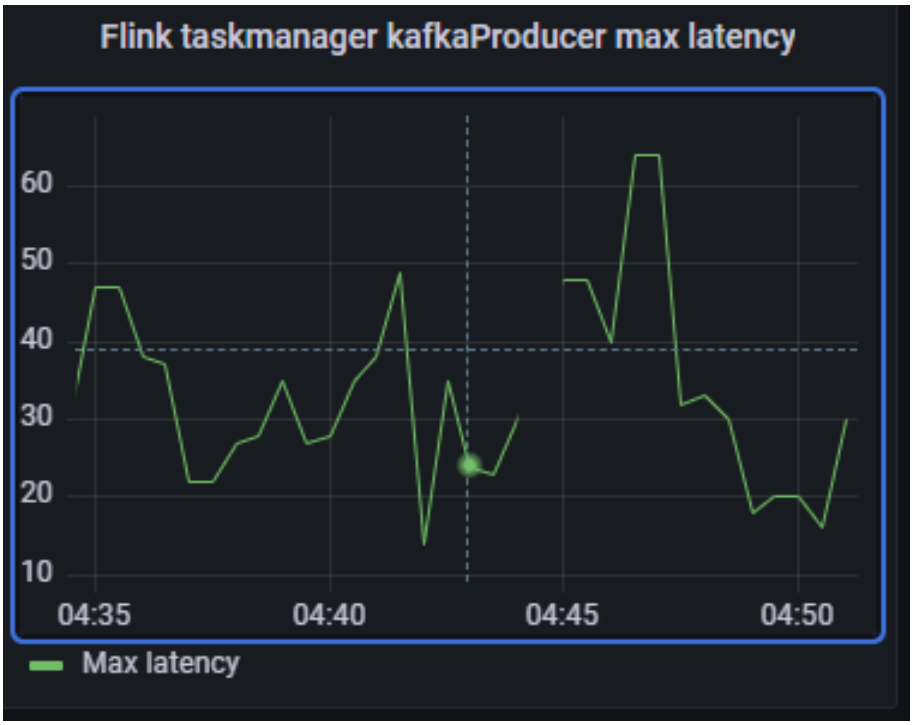
Συγκεκριμένα, ενώ το μέγιστο συνολικό latency του Kafka Sink παρουσίασε αύξηση, το μέγιστο latency ανά Flink Task Producer μειώθηκε σημαντικά.



Εικόνα 20:Flink to Kafka max latency

Η συμπεριφορά αυτή ερμηνεύεται ως εξής:

Η αύξηση του parallelism επέτρεψε την πιο ομοιόμορφη κατανομή του φόρτου στα Flink tasks, μειώνοντας τις καθυστερήσεις εντός του πλαισίου επεξεργασίας.



Εικόνα 21:FlinkProducerTask max latency

Ωστόσο, η αυξημένη ταχύτητα παραγωγής αποτελεσμάτων από τα tasks είχε ως συνέπεια την προσωρινή επιβάρυνση του Kafka Sink, το οποίο λειτουργεί ως κοινό σημείο συγκέντρωσης των εξερχόμενων γεγονότων.

Επομένως, το σύστημα εμφάνισε βελτίωση στην υπολογιστική του απόδοση, αλλά ταυτόχρονα ανέδειξε ένα νέο σημείο συμφόρησης στο τελικό στάδιο εγγραφής. Το εύρημα αυτό επιβεβαιώνει τη σημασία της ολιστικής παρακολούθησης όλων των υποσυστημάτων, καθώς η κλιμάκωση ενός επιμέρους τμήματος μπορεί να μεταφέρει τον περιορισμό σε επόμενο στάδιο της ροής δεδομένων.

Σε διάστημα παρατήρησης δεδομένων διάρκειας μιας ώρας, λάβαμε 6,229,902 events και παράχθηκαν συνολικά 55,897 training dataset και 10,468,098 prediction datasets

Έλεγχος ορθότητας

Για την αξιολόγηση της ορθότητας των δεδομένων και της συνοχής της ροής, επιλέχθηκαν δύο αντιπροσωπευτικοί χρήστες από το βασικό πείραμα (baseline). Στόχος ήταν η επαλήθευση της συνεκτικότητας των παραγόμενων συνόλων δεδομένων (datasets) και της χρονικής ορθότητας των εγγραφών σε σχέση με τα αντίστοιχα παράθυρα εκπαίδευσης (training windows).

Αναπτύχθηκε ένας μηχανισμός ανάλυσης, ο οποίος διάβαζε τα δεδομένα που παρήγαγε κάθε χρήστης και τα δύο τελικά topics της ροής, δηλαδή το power (ροή πρόβλεψης) και το dataset2train (ροή εκπαίδευσης). Ο κώδικας υπολόγιζε για κάθε χρήστη ένα σύνολο δεικτών ορθότητας, με βάση τις ακόλουθες μετρικές:

1. **Ποσοστό τιμών που συνδυάστηκαν στην ενιαία ροή (%).**
   Ο δείκτης αυτός υπολογίζεται σύμφωνα με τη σχέση:

$$\text{ποσοστό} = \frac{E_{\text{dataset2train}}}{E_{\text{κατάστασης}}} \times 100$$

   όπου $E_{\text{dataset2train}}$είναι ο αριθμός των τιμών που ενσωματώθηκαν στα σύνολα εκπαίδευσης, και $E_{\text{κατάστασης}}$ο συνολικός αριθμός των διαθέσιμων δεδομένων κατάστασης των συσκευών. Η μετρική αυτή εκφράζει το ποσοστό των δεδομένων κατάστασης που αξιοποιήθηκαν επιτυχώς στην ενιαία ροή, αντικατοπτρίζοντας τον βαθμό πληρότητας και ευθυγράμμισης μεταξύ των δύο ροών δεδομένων (power και state).

2. **Ποσοστό τιμών σε αύξουσα χρονολογική σειρά (%).**

   Ο δείκτης αυτός αξιολογεί τη χρονική συνέπεια των παραγόμενων dataset, ελέγχοντας εάν οι χρονικές σημάνσεις παραμένουν σε αυστηρά αύξουσα σειρά. Τιμές κοντά στο 100% υποδηλώνουν ορθή χρονική ταξινόμηση και απουσία καθυστερημένων ή αναδρομικών εγγραφών.

Σύμφωνα με τα αποτελέσματα της ανάλυσης:

- Για τον **χρήστη Α**, παρήχθησαν **3.596** σύνολα δεδομένων στη ροή πρόβλεψης και **33** στη ροή εκπαίδευσης, με **100% χρονική συνέπεια** και **89% επιτυχία συνδυασμού τιμών** στην ενιαία ροή.
- Για τον **χρήστη Β**, παρήχθησαν **3.594** σύνολα δεδομένων στη ροή πρόβλεψης και **25** στη ροή εκπαίδευσης, με **100% χρονική συνέπεια** και **91% επιτυχία συνδυασμού τιμών**.

Τα αποτελέσματα αυτά επιβεβαιώνουν την αξιοπιστία της διαδικασίας συγχρονισμού και ενοποίησης των ροών δεδομένων, καθώς και τη σταθερότητα του μηχανισμού παραγωγής training dataset. Η απόλυτη χρονική ορθότητα (100%) υποδηλώνει ορθή διαχείριση των χρονικών δεικτών, ενώ τα υψηλά ποσοστά ενοποίησης (>85%) τεκμηριώνουν την αποτελεσματικότητα του σταδίου της συνένωσης των δύο ροών και την λειτουργική συνέπεια του συστήματος.

4. Συμπεράσματα και Μελλοντική Εργασία

Σύνοψη και πορίσματα

Η υποδομή που φτιάξαμε μπορεί να δεχτεί μεγάλο όγκο δεδομένων από πολλές συσκευές, να τα οργανώσει σωστά και να τα μετατρέψει σε συνεπή κομμάτια πληροφορίας. Η αναγνώριση των διαστημάτων πρόβλεψης γίνεται αυτόματα, χωρίς να χρειάζεται συνεχής παρέμβαση από τον άνθρωπο. Το αποτέλεσμα είναι ότι έχουμε πάντα στη διάθεσή μας καθαρά και αξιόπιστα δεδομένα, τα οποία μπορούν άμεσα να χρησιμοποιηθούν τόσο για εκπαίδευση όσο και για προβλέψεις του ζητήματος του διαχωρισμού κατανομής φορτίων μέσω τεχνικών μηχανικής μάθησης σε πραγματικό χρόνο.

Η κλιμάκωση της υποδομής αξιολογήθηκε ξεχωριστά για κάθε συνιστώσα.

Στο επίπεδο της συλλογής των δεδομένων (EMQX), η δυνατότητα εξυπηρέτησης εξαρτάται κυρίως από τον αριθμό ταυτόχρονων συνδέσεων, τον ρυθμό παραγωγής μηνυμάτων και τους διαθέσιμους πόρους ανά pod. Η αντιμετώπιση πιθανών περιορισμών επιτυγχάνεται με αύξηση του πλήθους των pods, με ρύθμιση των ορίων συνδέσεων και με ενίσχυση των πόρων (CPU, μνήμη). Οι βασικές μετρικές που παρακολουθούμε είναι οι ενεργές συνδέσεις, τα μηνύματα ανά δευτερόλεπτο, η χρήση CPU/μνήμης καθώς και κάποιες φορές πόροι του λειτουργικού όπως οι file descriptors.

Στο επίπεδο της αποθήκευσης (Kafka), η επεκτασιμότητα επηρεάζεται από τον αριθμό των partitions ανά topic, το replication factor, τις δυνατότητες δικτύου και τον αριθμό των διαθέσιμων Brokers ή την ανισοκατανομή των μηνυμάτων στα partition ενός topic. Αντίστοιχα αντιμετωπίζονται με αύξηση των partitions, προσθήκη νέων Brokers ή βελτιστοποίηση των παραμέτρων των producers & consumers του cluster. Οι μετρικές που εξετάζουμε είναι το

throughput (MB/s), το consumer lag, χρήση δίσκου και δικτύου, καθώς και τυχόν ανισοκατανομή φορτίου μεταξύ των partitions (μέσω μετρικών όπως το skew).

Στο επίπεδο επεξεργασίας των δεδομένων (Flink), η κλιμάκωση καθορίζεται από τον βαθμό parallelism και τον αριθμό των task slots τους Cluster, από την ομοιομορφία της κατανομής κλειδιών και από τις παραμέτρους των παραθύρων. Η βελτίωση της απόδοσης γίνεται με αύξηση του βαθμού παραλληλισμού και των taskmanager. Αν κρίνουμε απαραίτητο εφαρμόζουμε αλλαγές στη στρατηγική κατανομής κλειδιών ή στις ρυθμίσεις των παραθύρων. Ενδεικτικές μετρικές είναι η καθυστέρηση επεξεργασίας ανά task, τα φαινόμενα backpressure, ο βαθμός απασχόλησης των subtasks, η διάρκεια των checkpoints και το ποσοστό της ανισοκατανομής (skew) στα δεδομένα.

Πιο σφαιρικά, σε επίπεδο υποδομής (Kubernetes) η κλιμάκωση επηρεάζεται από τα όρια πόρων (requests/limits) ανά pod, τις πολιτικές autoscaling και τον τρόπο κατανομής των pods στους κόμβους. Σε κάθετο επίπεδο, η κλιμακωσιμότητα του συστήματος επιτυγχάνεται με τροποποίηση των ορίων πόρων των επιμέρους και με κατάλληλους κανόνες affinity/anti-affinity για καλύτερη κατανομή το φόρτου εργασίας. Σε οριζόντιο επίπεδο, η κλιμακωσιμότητα επιτυγχάνεται με την προσθήκη επιπλέον υπολογιστικών κόμβων. Οι μετρικές που παρακολουθούμε περιλαμβάνουν τη χρήση CPU/μνήμης ανά pod, τις τυχόν επανεκκινήσεις των pods και τη συνολική αξιοποίηση των κόμβων του cluster.

Προτάσεις για μελλοντική εργασία

Σε επόμενο στάδιο, η παρούσα υποδομή μπορεί να επεκταθεί και να βελτιωθεί σε διάφορες κατευθύνσεις, με σκοπό την αύξηση της λειτουργικότητας, της αποδοτικότητας και της πρακτικής της αξιοποίησης σε πραγματικά περιβάλλοντα. Εκτενέστερα σας επόμενα βήματα θα ορίζαμε:

Εφαρμογή μοντέλων μηχανικής μάθησης πάνω στις ροές δεδομένων

Προβλέπεται η ενσωμάτωση αλγορίθμων μηχανικής μάθησης στις ίδιες τις ροές των δεδομένων, ώστε να επιτυγχάνεται πρόβλεψη κατανομής φορτίων και η εκπαίδευση ή αξιολόγηση μοντέλων μηχανικής μάθησης σε πραγματικό χρόνο. Με αυτόν τον τρόπο, η υποδομή θα μπορεί να λειτουργεί όχι μόνο ως πλατφόρμα συλλογής και επεξεργασίας, αλλά και ως ολοκληρωμένο σύστημα ευφυούς λήψης αποφάσεων. Ιδιαίτερο ενδιαφέρον παρουσιάζει η διερεύνηση τεχνικών online learning ή η ενσωμάτωση υπαρχόντων μοντέλων μέσω Flink ML, TensorFlow ή PyTorch.

Ανάπτυξη σε παραγωγικά περιβάλλοντα μεγάλης κλίμακας

Στο πλαίσιο της επόμενης φάσης ανάπτυξης, θα επιδιωχθεί η εγκατάσταση και λειτουργία του συστήματος σε παραγωγικά περιβάλλοντα με δεκάδες ή εκατοντάδες υπολογιστικούς κόμβους. Η διαδικασία αυτή θα επιτρέψει την πειραματική αξιολόγηση της ανθεκτικότητας, της σταθερότητας και της επεκτασιμότητας του συστήματος υπό μεγαλύτερες συνθήκες φόρτου, καθώς και την αναγνώριση νέων σημείων συμφόρησης.

Εφαρμογή τεχνικών αυτό-κλιμάκωσης

Μια σημαντική μελλοντική βελτίωση αφορά την ενσωμάτωση μηχανισμών αυτόματης κλιμάκωσης των πόρων (auto-scaling). Μέσω εργαλείων όπως το Kubernetes Horizontal/Vertical Pod Autoscaler ή το KEDA, θα είναι δυνατή η δυναμική προσαρμογή των διαθέσιμων υπολογιστικών πόρων με βάση τον φόρτο του συστήματος, διασφαλίζοντας βέλτιστη χρήση πόρων και υψηλή διαθεσιμότητα.

Ενσωμάτωση και εξαγωγή πιο πλούσιων μετρικών

Η περαιτέρω ανάπτυξη των αλγορίθμων των ροών ώστε να παράγουν πιο αξιόπιστες μετρικές για την συμπεριφορά της ροής. Οι μετρικές αυτές θα αξιοποιηθούν για τη συνεχή βελτιστοποίηση του συστήματος, με τη βοήθεια εργαλείων όπως το StatsD, το Prometheus και το Grafana.

Δοκιμή με πραγματικούς αισθητήρες και ανάπτυξη διεπαφής χρήστη

Τέλος, ιδιαίτερη έμφαση θα δοθεί στη διασύνδεση της υποδομής με πραγματικούς αισθητήρες IoT, καθώς και στην ανάπτυξη διεπαφής χρήστη για την δήλωση της κατάστασης των συσκευών σε βραχυπρόθεσμα διαστήματα.

5. Βιβλιογραφία

[1] M. Toledo-Orozco, C. Celi, F. Guartan, A. Peralta, C. Álvarez-Bel, and D. Morales, "Methodology for the disaggregation and forecast of demand flexibility in large consumers with the application of non-intrusive load monitoring techniques," *Energy and AI*, vol. 13, p. 100240, Jul. 2023, doi: 10.1016/J.EGYAI.2023.100240.

[2] S. Dash and N. C. Sahoo, "Electric energy disaggregation via non-intrusive load monitoring: A state-of-the-art systematic review," *Electric Power Systems Research*, vol. 213, p. 108673, Dec. 2022, doi: 10.1016/j.epsr.2022.108673.

[3] C. Nalmpantis, N. Virtsionis Gkalinikis, and D. Vrakas, "Neural Fourier Energy Disaggregation," *Sensors*, vol. 22, no. 2, p. 473, Jan. 2022, doi: 10.3390/s22020473.

[4] A. Moradzadeh, S. Zeinal-Kheiri, B. Mohammadi-Ivatloo, M. Abapour, and A. Anvari-Moghaddam, "Support Vector Machine-Assisted Improvement Residential Load Disaggregation," in *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, IEEE, Aug. 2020, pp. 1–6. doi: 10.1109/ICEE50131.2020.9260869.

[5] H. Bousbiat, Y. Himeur, I. Varlamis, F. Bensaali, and A. Amira, "Neural Load Disaggregation: Meta-Analysis, Federated Learning and Beyond," *Energies (Basel)*, vol. 16, no. 2, p. 991, Jan. 2023, doi: 10.3390/en16020991.

[6] A. Tongta and K. Chooruang, "Long Short-Term Memory (LSTM) Neural Networks Applied to Energy Disaggregation," in *2020 8th International Electrical Engineering Congress (iEECON)*, IEEE, Mar. 2020, pp. 1–4. doi: 10.1109/iEECON48109.2020.229559.

[7] C. Shin, S. Rho, H. Lee, and W. Rhee, "Data Requirements for Applying Machine Learning to Energy Disaggregation," *Energies (Basel)*, vol. 12, no. 9, p. 1696, May 2019, doi: 10.3390/en12091696.

[8] M. Khodayar, J. Wang, and Z. Wang, "Energy Disaggregation via Deep Temporal Dictionary Learning," Sep. 2018.

[9] F. Serepas, I. Papias, N. Bellos, and V. Marinakis, "A Comprehensive Approach to Real-Time and Batch Processing for Energy-Efficient IoT Homes: Leveraging Lambda Architecture and Data Lakes," in *2024 15th International Conference on Information, Intelligence, Systems & Applications (IISA)*, IEEE, Jul. 2024, pp. 1–6. doi: 10.1109/IISA62523.2024.10786715.

[10] I. Papias, V. Michalakopoulos, E. Sarmas, and V. Marinakis, "Aggregated Flexibility Dynamic Forecasting of Residential Energy Prosumers for DR Programs," in *2024 15th International Conference on Information, Intelligence, Systems & Applications (IISA)*, IEEE, Jul. 2024, pp. 1–9. doi: 10.1109/IISA62523.2024.10786677.

[11] F. Serepas, I. Papias, K. Christakis, N. Dimitropoulos, and V. Marinakis, "Lightweight Embedded IoT Gateway for Smart Homes Based on an ESP32 Microcontroller," *Computers*, vol. 14, no. 9, p. 391, Sep. 2025, doi: 10.3390/computers14090391.

[12] E. Mehmood and T. Anees, “Challenges and Solutions for Processing Real-Time Big Data Stream: A Systematic Literature Review,” *IEEE Access*, vol. 8, pp. 119123–119143, 2020, doi: 10.1109/ACCESS.2020.3005268.

[13] E. T. Nordli, S. G. Haugeland, P. H. Nguyen, H. Song, and F. Chauvel, “Migrating monoliths to cloud-native microservices for customizable SaaS,” *Inf Softw Technol*, vol. 160, p. 107230, Aug. 2023, doi: 10.1016/J.INFSOF.2023.107230.

[14] “Kubernetes architecture.” Accessed: Jul. 02, 2025. [Online]. Available: https://kubernetes.io/docs/concepts/architecture/

[15] H. Agrawal, “Advanced Objects In Kubernetes Clusters,” in *Kubernetes Fundamentals*, Berkeley, CA: Apress, 2023, pp. 109–155. doi: 10.1007/978-1-4842-9729-2_4.

[16] “Helm chart fundamentals.” Accessed: Jul. 05, 2025. [Online]. Available: https://helm.sh/docs/topics/charts/

[17] I. Papias, E. Sarrantinopoulos, V. Michalakopoulos, E. Sarmas, and V. Marinakis, “FLEXiT: An Innovative Tool for Estimating Residential Energy Flexibility for Aggregated & Disaggregated Demand-Side Management,” 2025. doi: 10.2139/ssrn.5503416.

[18] “MQTT Essentials The Ultimate Guide to the MQTT Protocol for IoT Messaging.” [Online]. Available: www.hivemq.com

[19] Neha Narkhede, Gwen Shapira, and Todd Palino, *Kafka: The Definitive Guide*. O’Reilly Media, Inc, 2017.

[20] “Kafka Bridge documentation.” Accessed: Aug. 02, 2025. [Online]. Available: https://strimzi.io/docs/bridge/latest/full

[21] “Deploy Kafka resources with Strimzi operator.” Accessed: Aug. 09, 2025. [Online]. Available: https://strimzi.io/docs/operators/latest/overview

[22] A. Trofimov, I. E. Kuralenok, N. Marshalkin, and B. Novikov, “Delivery, consistency, and determinism: rethinking guarantees in distributed stream processing,” Jul. 2019.

[23] P. Carbone *et al.*, “Apache Flink™: Stream and Batch Processing in a Single Engine,” 2015.

[24] “Flink architecture and fundamental components.” Accessed: Aug. 02, 2025. [Online]. Available: https://nightlies.apache.org/flink/flink-docs-release-1.20/docs/concepts/flink-architecture/

[25] “Time in Flink”, Accessed: Sep. 17, 2025. [Online]. Available: https://nightlies.apache.org/flink/flink-docs-release-1.20/docs/concepts/time/

[26] “Stateful Stream Processing in Flink.” Accessed: Aug. 30, 2025. [Online]. Available: https://nightlies.apache.org/flink/flink-docs-release-1.20/docs/concepts/stateful-stream-processing/

[27] “Checkpointing in Flink.” Accessed: Aug. 30, 2025. [Online]. Available: https://nightlies.apache.org/flink/flink-docs-release-1.20/docs/dev/datastream/fault-tolerance/checkpointing/

[28] A. Katsifodimos and S. Schelter, “Apache Flink: Stream Analytics at Scale,” in *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, IEEE, Apr. 2016, pp. 193–193. doi: 10.1109/IC2EW.2016.56.

[29] “Flink Kubernetes operator architecture.” Accessed: Sep. 01, 2025. [Online]. Available: https://nightlies.apache.org/flink/flink-kubernetes-operator-docs-main/docs/concepts/architecture/

[30] “MicroK8s available addons.” Accessed: Sep. 05, 2025. [Online]. Available: https://microk8s.io/docs/addons

Παράρτημα Α

Σύνδεσμοι υλοποίησης

Ο κώδικας της υλοποίησης όλων των στοιχείων αυτής της διπλωματικής είναι διαθέσιμος στην ιστοσελίδα του Github μέσω του συνδέσμου *https://github.com/tkatsoulas/diploma_ece_ntua*