ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΚΑΙ ΤΕΧΝΙΚΕΣ ΑΝΑΛΥΣΗΣ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΣΕ ΣΥΣΤΗΜΑΤΑ IoT

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Λένος Κ. Τσοκκής

Επιβλέπουσα : Βασιλική Καντερέ
Επίκουρη Καθηγήτρια Ε.Μ.Π

Αθήνα, Ιούλιος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΚΑΙ ΤΕΧΝΙΚΕΣ ΑΝΑΛΥΣΗΣ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΣΕ ΣΥΣΤΗΜΑΤΑ IoT

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Λένος Κ. Τσοκκής

Επιβλέπουσα : Βασιλική Καντερέ
Επίκουρη Καθηγήτρια Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Ιουλίου 2020.

.....	.....	.....
Β. Καντερέ	Ν. Κοζύρης	Σ. Παπαβασιλείου
Επίκουρη Καθηγήτρια Ε.Μ.Π	Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2020

.....

Λένος Κ. Τσοκκής

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

Περίληψη

Ζούμε σε μια εποχή όπου η τεχνολογία αναπτύσσεται με ραγδαίους ρυθμούς. Εκατομμύρια συνδεδεμένες συσκευές αλληλοεπιδρούν και ανταλλάζουν τεράστιο όγκο δεδομένων μεταξύ τους καθημερινά, με στόχο την αύξηση του βιοτικού επιπέδου συνεισφέροντας σε πολλούς τομείς της καθημερινότητας όπως η υγεία, επικοινωνία, εκπαίδευση, διαβίωση, βιομηχανία κτλ. Κινητές συσκευές, smartwatch, κάμερες, αισθητήρες κίνησης, θερμοκρασίας και φωτός είναι μερικά κομμάτια ενός τεράστιου διασυνδεδεμένου συστήματος που μας περιβάλλει το οποίο ονομάζεται Internet of Things (IoT). Μπορεί κανείς να φανταστεί τον τεράστιο όγκο δεδομένων (Big Data) που παράγεται, μεταδίδεται, επεξεργάζεται και αποθηκεύεται καθημερινά έτσι ώστε να εξασφαλιστεί η ομαλή λειτουργία του συστήματος και όλων των επιμέρους κομματιών του. Αυτό δημιουργεί μεγάλο πρόβλημα στην διαδικασία διανομής, επεξεργασίας και αποθήκευσης λόγω του τεράστιου όγκου δεδομένων που καλείται το σύστημα να διαχειριστεί, καθώς και τις ανάγκες των εκατομμυρίων χρηστών καθημερινά. Έτσι κρίνεται απαραίτητη η χρήση βέλτιστων τεχνικών αντιμετώπισης του προβλήματος αυτού. Οι δύο τεχνολογίες που αποτελούν τους κύριους πυλώνες του Internet of Things είναι το cloud computing και το edge computing συνθέτοντας μια αρχιτεκτονική 3 επιπέδων (edge devices – edge computing – cloud computing). Ο συνδυασμός κατάλληλων αρχιτεκτονικών cloud computing και edge computing μπορούν να βελτιώσουν σημαντικά τα προβλήματα που αντιμετωπίζουν τα σημερινά IoT συστήματα. Στην παρούσα διπλωματική εργασία θα εξερευνηθούν αρχιτεκτονικές cloud computing και edge computing, οι οποίες σε συνδυασμό με την μελέτη τεχνικών μετάδοσης δεδομένων - πρωτοκόλλων επικοινωνίας μεταξύ των συσκευών, τεχνικών αποθήκευσης δεδομένων μεγάλης κλίμακας, τεχνικών streaming και επεξεργασίας δεδομένων σε πραγματικό χρόνο, τεχνικών διαμοιρασμού πόρων και παράλληλης επεξεργασίας δεδομένων όπως επίσης και τεχνικές ενορχήστρωσης, συντονισμού συστήματος και απομόνωσης εφαρμογών θα οδηγήσουν σε εύρεση βέλτιστων μεθόδων αντιμετώπισης των προβλημάτων διαχείρισης και επεξεργασίας του τεραστίου όγκου δεδομένων.

Λέξεις - κλειδιά: Internet of Things, cloud computing, edge computing, big data, data streaming, batch processing, real time processing, τεχνικές αποθήκευσης δεδομένων, πρωτόκολλα επικοινωνίας συσκευών, distributed systems

Abstract

We live in an age where technology is growing rapidly. Millions of connected devices interact and exchange huge amounts of data with each other on a daily basis, aiming to increase living standards by contributing to many areas of daily life such as health, communication, education, living, industry, etc. Mobile devices, smartwatch, cameras, sensors and light are some of the parts of a huge interconnected system that surrounds us called the Internet of Things (IoT). One can imagine the huge amount of data (Big Data) being generated, transmitted, processed and stored on a daily basis to ensure the smooth operation of the system and all its individual parts. This creates a big problem in the process of distribution, processing and storage due to the huge amount of data that the system is required to manage as well as the needs of millions of users daily. Thus, it is considered necessary to use optimal techniques to deal with this problem. The two technologies that make up the main pillars of the Internet of Things are cloud computing and edge computing, composing a 3-level architecture (edge devices - edge computing - cloud computing). The combination of appropriate cloud computing and edge computing architecture can significantly improve the problems facing today's IoT systems. In this dissertation we will explore the architectures of cloud computing and edge computing, which in combination with the study of data transmission techniques - communication protocols between devices, large-scale data storage techniques, real-time streaming and data processing techniques, resource sharing techniques and data processing as well as orchestration techniques, system coordination and application containment will lead to finding optimal methods for dealing with the problems of managing and processing huge data volumes.

Keywords: Internet of Things, cloud computing, edge computing, big data, data streaming, batch processing, real time processing, data storage techniques, device communication protocols, distributed systems

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια Βασιλική Καντερέ για την ανάθεση της συγκεκριμένη διπλωματικής εργασίας, η οποία μου έδωσε την ευκαιρία να έρθω σε επαφή με καινοτόμες τεχνολογίες και αρχιτεκτονικές σε έναν σημαντικό τομέα της σύγχρονης τεχνολογίας όπως είναι αυτός του Internet of Things.
Ακόμη θα ήθελα να ευχαριστήσω θερμά τον υποψήφιο διδάκτορα του Ε.Μ.Π Πάρη Κερασιώτη, για τις οδηγίες, τις κατευθύνσεις και την βοήθεια που μου προσέφερε κατά την διάρκεια των σταδίων εκπόνηση της παρούσας διπλωματικής εργασίας.
Ένα μεγάλο ευχαριστώ στους συμφοιτητές και φίλους μου Σάββα, Νεόφυτο, Μηνά, Μαρία, Ειρήνη, Σταύρο, Γιώργο, Θωμά, Ανδρέα, Χριστόφορο για τα όμορφα φοιτητικά χρόνια και τις αξέχαστες στιγμές που μου προσέφεραν.
Τέλος ένα τεράστιο ευχαριστώ θα ήθελα να αφιερώσω στην οικογένεια μου για την στήριξη και την βοήθεια που μου προσέφεραν στην μέχρι τώρα σταδιοδρομία μου.

Λένος Κ. Τσοκκής
Αθήνα, Ιούλιος 2020

Πίνακας περιεχομένων

Πίνακας Σχημάτων

1. Εισαγωγή

1.1 Internet of Things

Με τον όρο Internet of Things (IoT), ορίζουμε ένα σύστημα-δίκτυο επικοινωνίας το οποίο αποτελείται από διασυνδεδεμένες ηλεκτρονικές συσκευές (things), οι οποίες ενσωματώνουν μέσα τους λογισμικό, αισθητήρες και συνδεσιμότητα στο διαδίκτυο με σκοπό την ανταλλαγή δεδομένων χωρίς την απαίτηση αλληλεπίδρασης μεταξύ ανθρώπων ή ανθρώπων και υπολογιστών. Συγκεκριμένα με τον όρο "Thing" μπορεί να αναφερόμαστε σε κάποια έξυπνη συσκευή όπως smartphone, smartwatch, σε συσκευές με αισθητήρες κίνησης, θερμοκρασίας και φωτός αλλά ακόμη και σε ανθρώπους με μοσχεύματα (βηματοδότες κτλ.) ή ζώα με ενσωματωμένα ολοκληρωμένα (biochip transponder).
Η ιδέα τοποθέτησης αισθητήρων σε συσκευές βρισκόταν ήδη υπό συζήτηση την περίοδο 1980 – 1990 αλλά η πρόοδος καθυστερούσε λόγω του ότι η υπάρχουσα τεχνολογία δεν ήταν ικανή να υλοποιήσει την ανάγκη αυτή. Η έλλειψη φθηνών επεξεργαστών που να έχουν τις κατάλληλες ενεργειακές δυνατότητες καθιστούσε την συνδεσιμότητα εκατομμυρίων συσκευών δύσκολη και μη συμφέρουσα ως προς το κόστος. Με την πάροδο του χρόνου όμως η εμφάνιση των RFID tags (ενσωματωμένα μικρής ενεργειακής απαίτησης με δυνατότητες ασύρματης επικοινωνίας) καθώς και η υιοθέτηση του πρωτοκόλλου IPV6 ήταν ένα σημαντικό βήμα στην κλιμάκωση του IoT. Το 1999 ο όρος Internet of Things αποδόθηκε από τον επιχειρηματία Kevin Ashton, έναν από τους ιδρυτές του Auto-ID center στο MIT και μέλος της ομάδας που ανακάλυψε τον τρόπο να συνδέει αντικείμενα στο διαδίκτυο με την βοήθεια της ετικέτας RFID [1].
Μια από τις πρώτες εφαρμογές του IoT ήταν απλά η τοποθέτηση των RFID tags σε ακριβές συσκευές για τον εντοπισμό της τοποθεσίας τους. Από τότε όμως το κόστος τοποθέτησης αισθητήρων και δυνατότητας συνδεσιμότητας στο διαδίκτυο σε συσκευές, συνεχίζει να μειώνεται δραματικά μέχρι σήμερα, με τους ειδικούς να προβλέπουν ότι η βασική λειτουργικότητα μπορεί μια μέρα να κοστίζει μόνο 10 σεντς, με αποτέλεσμα να μπορούμε να συνδέσουμε σχεδόν τα πάντα στο διαδίκτυο. Με την βοήθεια των φθηνών ενσωματωμένων κυκλωμάτων και της κλιμάκωσης των ασύρματων δικτύων, υπάρχει ένα τεράστιο πεδίο εφαρμογής στα συστήματα IoT.

Μερικές από τις σύγχρονες εφαρμογές του IoT είναι :

- **Wearables (smartwatches, fit bands):** Διάφορα αξεσουάρ όπως ρολόγια ή έξυπνα βραχιόλια τα οποία συλλέγουν και απεικονίζουν στον χρήστη διάφορα δεδομένα όπως βήματα, παλμούς καρδιάς, ώρες ύπνου, διαθέτοντας επίσης και δυνατότητες τηλεφωνικών κλήσεων, λήψη μηνυμάτων, email και σύνδεση σε κοινωνικά δίκτυα.

- **Smart Homes:** Διασύνδεση διάφορων συσκευών και λειτουργικοτήτων του σπιτιού όπως ηλεκτρικές συσκευές, φώτα, συναγερμός με το κινητό για την αυτοματοποίηση διαφόρων λειτουργειών από μακριά.

- **Γεωργία:** Η χρήση διαφόρων συσκευών με αισθητήρες θερμοκρασίας, υγρασίας και φωτός βοηθά στον έλεγχο διαφόρων περιβαλλοντικών παραμέτρων οι οποίες αναλύονται και επεξεργάζονται οδηγώντας στην εφαρμογή των κατάλληλων ενεργειών για την εξασφάλιση καλύτερων σοδειών.

- **Υγεία:** Διασύνδεση του ιατρικού εξοπλισμού και μηχανημάτων (παλμογράφοι, MRI, CT scans) για την ανταλλαγή δεδομένων ασθενών σε πραγματικό χρόνο, με σκοπό την ανάλυση και επεξεργασία τους για την ενίσχυση του ερευνητικού τομέα της ιατρικής καθώς και την βελτίωση παροχής ιατρικής περίθαλψης.

- **Βιομηχανία:** Η χρήση διασυνδεδεμένων συσκευών μπορεί να βοηθήσει σε πολλά στάδια της βιομηχανίας όπως έλεγχος αποθέματος, η αυτοματοποίηση της παραγωγής, αποστολή αναφορών κατά την διάρκεια χρήσης ενός προϊόντος για την αντιμετώπιση τυχών βλαβών ή προβλημάτων καθώς και βελτίωσης του σε επερχόμενη έκδοση, ασφάλεια και προστασία.

Ενώνοντας συσκευές, ανθρώπους και περιβάλλοντα μέσω των συστημάτων IoT δημιουργούνται αναμφίβολα πλεονεκτήματα όπως η δυνατότητα ανάλυσης τεράστιου όγκου δεδομένων με στόχο την πρόβλεψη μοτίβων για καλύτερη λήψη στρατηγικών αποφάσεων, μείωση των ανθρωπίνων λαθών, παρακολούθηση των συστημάτων απομακρυσμένα, σε πραγματικό χρόνο καθώς και βελτίωση της παραγωγικότητας μέσω της συνεχούς παρακολούθησης και ελέγχου διαφόρων διεργασιών που βελτιστοποιούν την παραγωγικότητα και αποδοτικότητα.
Τα συστήματα IoT αν και αποτελούν την τελευταία λέξη της τεχνολογίας, βρίσκονται σε ένα στάδιο συνεχούς ανάπτυξης και βελτίωσης, για αυτό και αντιμετωπίζουν κάποια σημαντικά προβλήματα. Ένα σημαντικό πρόβλημα των συστημάτων αυτών είναι η διαχείριση και η επεξεργασία του τεράστιου όγκου δεδομένων που παράγουν οι συσκευές οι οποίες διατηρούν συνεχή επικοινωνία με το δίκτυο (data streaming). Αυτό οδηγεί στην απαίτηση τεράστιων χώρων αποθήκευσης καθώς και εξειδικευμένες τεχνικές επεξεργασίας τους. Ακόμη υπάρχουν προβλήματα που αφορούν την ασφάλεια τόσο των δεδομένων, όσο και ολόκληρου του συστήματος, αφού κάποιος κακόβουλος χρήστης μπορεί είτε να υποκλέψει στοιχεία που διακινούνται στο δίκτυο ή ακόμη και να αποκτήσει έλεγχο σε μέρη του συστήματος. Η έλλειψη ενός διεθνούς προτύπου συμβατικότητας στα συστήματα IoT αποτελεί άλλη μία πρόκληση για τα συστήματα IoT αφού καθιστά δύσκολη την επικοινωνία μεταξύ συσκευών από διαφορετικούς κατασκευαστές.
Στην παρούσα διπλωματική εργασία θα εξερευνηθούν αρχιτεκτονικές edge computing και αρχιτεκτονικές cloud computing με σκοπό την εύρεση βέλτιστων τεχνικών για την αντιμετώπιση του προβλήματος της επεξεργασίας και αποθήκευσης του τεράστιου όγκου δεδομένων.

2. Αρχιτεκτονικές επεξεργασίας δεδομένων μεγάλης κλίμακας (Big Data)

Καθώς το διαδίκτυο γίνεται όλο και πιο προσβάσιμο σε μεγάλο ποσοστό ανθρώπων, το πλήθος των δεδομένων που παράγονται καθημερινά και είναι διαθέσιμα στο διαδίκτυο αυξάνονται με ραγδαίους ρυθμούς. Υπολογίζεται ότι το 2020 περίπου 50 δισεκατομμύρια συσκευές θα είναι συνδεδεμένες στο διαδίκτυο μέσω συστημάτων IoT. Τα δεδομένα που παράγονται συνολικά ανά έτος είναι της τάξης μεγέθους των zettabytes, δηλαδή ενός τρισεκατομμυρίου gigabytes. Έτσι η ανάγκη για κλιμακούμενες, γρήγορες και ανθεκτικές σε σφάλματα αρχιτεκτονικές επεξεργασίας και αποθήκευσης μεγάλου όγκου δεδομένων αυξάνεται. Η επεξεργασία των δεδομένων που παράγονται μπορεί να γίνεται τοπικά (edge computing) καθώς και απομακρυσμένα στο cloud (cloud computing) στις 3-tier IoT αρχιτεκτονικές για αυτό και θα μελετηθούν και τα 2 ήδη αρχιτεκτονικών. Οι αρχιτεκτονικές αυτές είναι σχεδιασμένες ώστε να χειρίζονται την διαδικασία ανάλυσης και επεξεργασίας δεδομένων που είναι πολύ μεγάλα και πολύπλοκα για τις συνηθισμένες βάσεις δεδομένων. Είναι κατάλληλες για επεξεργασία των δεδομένων σε παρτίδες (batch processing), επεξεργασία σε πραγματικό χρόνο (real time processing), προγνωστική ανάλυση (predictive analytics) καθώς και εφαρμογή μηχανικής μάθησης (machine learning) πάνω σε αυτά.

2.1 Cloud Computing

Με τον όρο cloud computing αναφερόμαστε στην δυνατότητα παροχής υπολογιστικών πόρων απομακρυσμένα, μέσω διαδικτύου για την αυτοματοποίηση διαδικασιών, εκτέλεση υπολογισμών, επεξεργασία και αποθήκευση δεδομένων. Η έννοια του cloud computing εμφανίστηκε την δεκαετία του 1950 σε εταιρείες και εκπαιδευτικά ιδρύματα όπου χρήστες μέσω τερματικών, είχαν πρόσβαση σε κεντρικά συστήματα υπολογιστών με μεγάλη υπολογιστική ισχύ και δυνατότητες αποθήκευσης.

Οι cloud αρχιτεκτονικές επεξεργασίας μεγάλου όγκου δεδομένων αποτελούνται συνήθως από συνδυασμό των παρακάτω λειτουργιών [2]:

- **Data sources**: Οι πηγές δεδομένων όπως βάσεις δεδομένων, αρχεία όπως log files καθώς και δεδομένα που παράγονται σε πραγματικό χρόνο από συσκευές IoT.

- **Batch processing**: Αρχική επεξεργασία των δεδομένων, όπου λόγω του μεγάλου μεγέθους τους, φιλτράρονται, αθροίζονται και προετοιμάζονται σε παρτίδες για ανάλυση.

- **Real-time message ingestion:** Λειτουργεί ως buffer για δεδομένα συνεχούς ροής τα οποία αποθηκεύονται για περαιτέρω επεξεργασία.

- **Stream processing:** Αφού τα δεδομένα συνεχούς ροής αποθηκευτούν φιλτράρονται, αθροίζονται και προετοιμάζονται για ανάλυση.

- **Analytical data store:** Τα επεξεργασμένα δεδομένα σερβίρονται σε μια δομημένη μορφή ώστε να μπορούν τα αναλυτικά εργαλεία να εφαρμόζουν ερωτήματα σε αυτά.

- **Analysis and reporting:** Ανάλυση, έκθεση αναφορών καθώς και οπτικοποίηση των αποτελεσμάτων.

- **Orchestration:** Αυτοματοποίηση των επανειλημμένων διεργασιών μετατροπής και μετακίνησης δεδομένων στα διάφορα μέρη της αρχιτεκτονικής.

Στην παρακάτω ενότητα θα μελετηθούν διάφορες αρχιτεκτονικές cloud computing.

2.1.1 Αρχιτεκτονική λ (Lambda architecture)

Η αρχιτεκτονική λ είναι μια αρχιτεκτονική επεξεργασίας μεγάλου όγκου δεδομένων η οποία είναι ικανή να διαχειριστεί τεράστιες ποσότητες δεδομένων με αποδοτικό τρόπο, η οποία πήρε το όνομα της από τον Nathan Marz. Είναι μια γενική και κλιμακώσιμη αρχιτεκτονική επεξεργασίας δεδομένων που μπορεί να χρησιμοποιηθεί και σε ροές δεδομένων πραγματικού χρόνου με ανεκτικότητα στα σφάλματα. Ένα από τα κύρια χαρακτηριστικά της είναι η υψηλή διακίνηση δεδομένων (throughput) καθώς και ο μικρός χρόνος απόκρισης.

Η αρχιτεκτονική λ αποτελείται από τα εξής 3 επίπεδα επεξεργασίας:

- **Batch layer:** Αποτελεί το cold path της ροή δεδομένων στην αρχιτεκτονική. Αποθηκεύει τα δεδομένα στην αρχική τους μορφή και εκτελεί επεξεργασία ανά παρτίδες πάνω σε αυτά. Δεν υπάρχουν απαιτήσεις για μικρό χρόνο απόκρισης από το επίπεδο αυτό, για αυτό και τα δεδομένα αυτά χαρακτηρίζονται από μεγάλη ακρίβεια. Το αποτέλεσμα της επεξεργασίας των δεδομένων αυτών αποθηκεύεται σε batch views. [3]

- **Speed layer (Stream layer):** Αποτελεί το hot path της ροή δεδομένων στην αρχιτεκτονική. Τα δεδομένα αναλύονται σε πραγματικό χρόνο με αντίτιμο την ακρίβεια τους. Χαρακτηρίζεται από μικρό χρόνο απόκρισης, με βάσει τις απαιτήσεις του serving layer από αυτό, καθώς διαχειρίζεται δεδομένα σε πραγματικό χρόνο και έχει μικρότερο υπολογιστικό φόρτο από το batch layer.

- **Serving layer:** Τα δεδομένα από το cold path και hot path φτάνουν στο serving layer σε μορφές batch views και real-time views αντίστοιχα. Από τα views αυτά το serving layer εξυπηρετεί τα εισερχόμενα ερωτήματα των πελατών. [4]

Λειτουργία

Καθώς νέα δεδομένα εισάγονται στο σύστημα, αυτά διανέμονται παράλληλα στο batch layer (cold path) και speed layer (hot path) αντίστοιχα. Τα δεδομένα που εισάγονται στο batch layer είναι αμετάβλητα και προσθέτονται συνεχώς πάνω στα παλιά δεδομένα μέσα στις μεγάλες βάσεις δεδομένων (data lakes), χωρίς να τα αντικαθιστούν. Οι αλλαγές στα δεδομένα αποθηκεύονται με χρονοσφραγίδες επιτρέποντας έτσι την ανασύνθεση των batch views ανά πάσα χρονική στιγμή. Τα δεδομένα που εισάγονται στο speed layer είτε θα επεξεργαστούν σε μικρά χρονικά παράθυρα και θα διατεθούν σε πελάτες που χρειάζονται δεδομένα σε πραγματικό χρόνο με πιθανώς λιγότερη ακρίβεια, είτε θα χρησιμοποιηθούν για να ενημερώσουν το serving layer με βάση τα πιο πρόσφατα δεδομένα.

Η γενική λειτουργία της αρχιτεκτονικής λ χαρακτηρίζεται από την σχέση:

Query = λ (Complete data) = λ (Live Streaming Data) * λ (Stored Data)

η οποία εκφράζει τις απαντήσεις στα εισερχόμενα ερωτήματα (queries) στο σύστημα ως τον συνδυασμό των αποτελεσμάτων από την επεξεργασία (filters, aggregators, αλγόριθμοι, στατιστική) των δεδομένων σε μορφή batch view από το batch layer και real time view από το speed layer.

Εφαρμογές

Η αρχιτεκτονική λ εφαρμόζεται σε μοντέλα επεξεργασίας δεδομένων όπου τα ερωτήματα των χρηστών πρέπει να ικανοποιούνται κατ' απαίτηση από το batch layer. Είναι κατάλληλη για συστήματα τα οποία εξυπηρετούν εφαρμογές οι οποίες είναι απαιτητικές ως προς την επεξεργασία των δεδομένων και την ακρίβεια των αποτελεσμάτων. Ακόμη είναι κατάλληλη για μοντέλα τα οποία χρειάζονται γρήγορες απαντήσεις σε ερωτήματα καθώς και πολλές ενημερώσεις των στοιχείων μέσω ροής δεδομένων σε πραγματικό χρόνο (data streaming) όπως επίσης και σε συστήματα όπου δεν πρέπει να διαγράφονται προηγούμενες εγγραφές στοιχείων αλλά να προστίθενται καινούριες ενημερώσεις αυτών πάνω σε αυτά στην βάση δεδομένων.

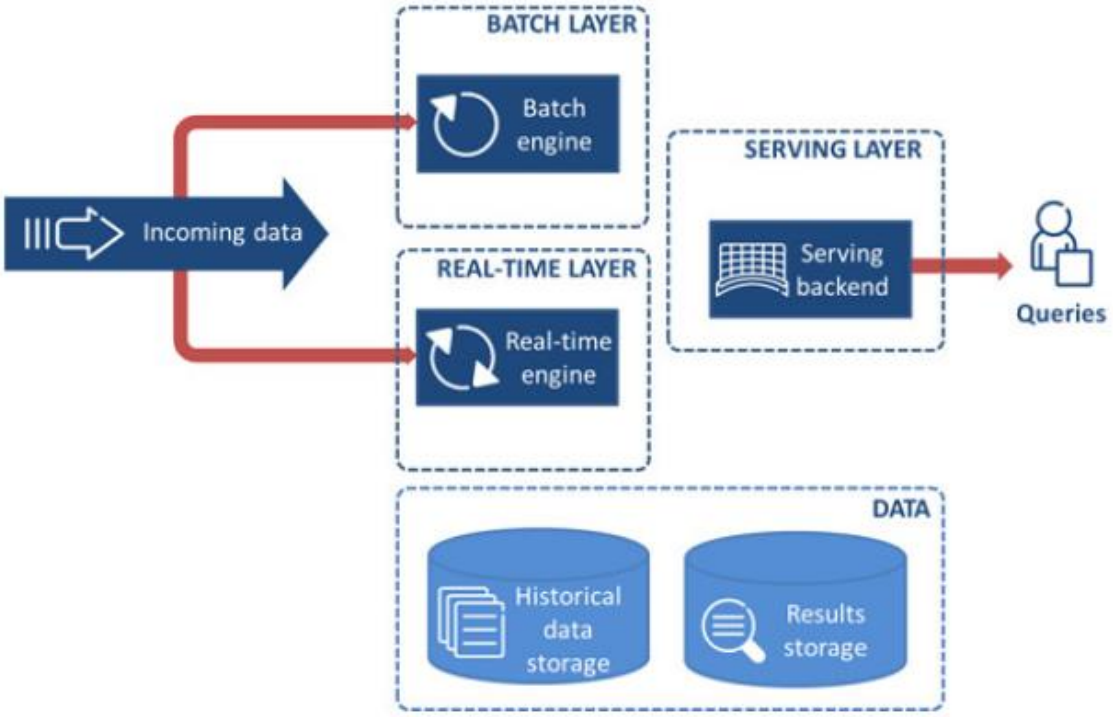
Πλεονεκτήματα

Τα πλεονεκτήματα της αρχιτεκτονικής λ είναι ότι αποτελεί μία κλιμακώσιμη και ανθεκτική σε λάθη αρχιτεκτονική η οποία παρέχει καλή ισορροπία μεταξύ ταχύτητας εξυπηρέτησης και αξιοπιστίας των δεδομένων. Επίσης το batch layer με την βοήθεια του κατανεμημένου συστήματος αποθήκευσης μας εξασφαλίζει μικρή πιθανότητα λάθους ακόμη και μετά από σφάλμα συστήματος.

Μειονεκτήματα

Τα μειονεκτήματα που παρουσιάζει η αρχιτεκτονική αυτή είναι η μη αποδοτική συνεχόμενη επανεπεξεργασία των batch views σε ορισμένα σενάρια χρήσης και το γεγονός ότι οι τεχνολογίες που απαιτούνται για να τρέξουν τα 3 επίπεδα (batch, speed, serving) είναι δύσκολες στη υλοποίηση. Ακόμη ο συγχρονισμός μεταξύ batch και speed layer μπορεί να αποβεί ακριβός σε υπολογιστικό χρόνο, όπως και η συντήρηση των 2 επιπέδων είναι

σχετικά απαιτητική αφού και τα 2 επίπεδα είναι μεταξύ τους διακριτά και εντελώς κατανεμημένα.
Σε γενικές γραμμές η αρχιτεκτονική λ υλοποιεί τους στόχους της αλλά με αυξημένη πολυπλοκότητα σε ορισμένα σενάρια χρήσης, στα οποία πιθανόν να μην χρειάζεται η συνεισφορά και από τα 2 επίπεδα για την εξυπηρέτηση των ερωτημάτων στο σύστημα.



Σχήμα 1.1 Αρχιτεκτονική λ

Πηγή: *https://www.ericsson.com/en/blog/2015/11/data-processing-architectures--lambda-and-kappa*

2.1.2 Αρχιτεκτονική κ (Kappa architecture)

Με βάσει τα μειονεκτήματα που παρουσιάζει η αρχιτεκτονική λ όσον αφορά την πολυπλοκότητα των επιπέδων της καθώς και την διπλή και ίσως περιττή σε κάποιες περιπτώσεις, υλοποίηση της ίδιας υπολογιστικής λογικής, ο Jay Kreps μηχανικός στην εταιρεία LinkedIn το 2014 πρότεινε μια άλλη προσέγγιση στην επεξεργασία δεδομένων, αυτήν της αρχιτεκτονικής κ [5].
Η αρχιτεκτονική κ είναι μια αρχιτεκτονική επεξεργασίας μεγάλου όγκου δεδομένων η οποία είναι κατάλληλη για επεξεργασία σε πραγματικό χρόνο. Δεν αποτελεί υποκατάστατο της αρχιτεκτονικής λ, αλλά αντιμετωπίζεται σαν μία εναλλακτική λύση για τις περιπτώσεις όπου η χρήση του batch layer δεν αναγκαία για την ικανοποίηση των αναγκών του συστήματος. Χαρακτηρίζεται από μεγαλύτερη απλότητα σε σχέση με την αρχιτεκτονική λ όπως επίσης και λιγότερες απαιτήσεις σε κώδικα υλοποίησης.

Η αρχιτεκτονική κ αποτελείται από τα εξής 2 επίπεδα επεξεργασίας:

- **Speed layer (Real time layer):** Τα δεδομένα που εισάγονται σε συνεχή ροή (data streaming) είναι αμετάβλητα, συλλέγονται ολοκληρωτικά και όχι σε χρονικά παράθυρα, όπου έπειτα αναλύονται σε πραγματικό χρόνο.

- **Serving layer:** Τα δεδομένα φτάνουν στο serving layer από το speed layer σε μορφή real time view. Από τα views αυτά το serving layer εξυπηρετεί τα εισερχόμενα ερωτήματα των πελατών.

Λειτουργία
Η συνεχόμενη ροή δεδομένων (data stream) ένα μόνο μονοπάτι στο σύστημα και εισάγεται σε ένα κατανεμημένο ενιαίο log σε μορφή ροής γεγονότων (event stream). Τα γεγονότα αυτά ταξινομούνται και η κατάσταση κάθε γεγονότος αλλάζει μόνο όταν ένα νέο γεγονός προστεθεί στο log. Ακολούθως εισάγονται στο speed layer όπου εκεί επεξεργάζονται σε πραγματικό χρόνο και προωθούνται σε μορφή real time view στο serving layer όπως γίνεται με παρόμοιο τρόπο στην αρχιτεκτονική λ. Σε περίπτωση που χρειαστεί η ανασύνθεση ολόκληρου του σετ δεδομένων απλά γίνεται επαναφόρτωση της ροής δεδομένων με χρήση παραλληλισμού. [6]

Η γενική λειτουργία της αρχιτεκτονικής κ χαρακτηρίζεται από την σχέση:

Query = κ (Complete data) = κ (Live Streaming Data)

η οποία εκφράζει τις απαντήσεις στα εισερχόμενα ερωτήματα (queries) στο σύστημα ως το αποτέλεσμα της επεξεργασίας (filters, aggregators, αλγόριθμοι, στατιστική), της συνεχούς ροής δεδομένων στο speed layer, σε μορφή real time view.

Εφαρμογές

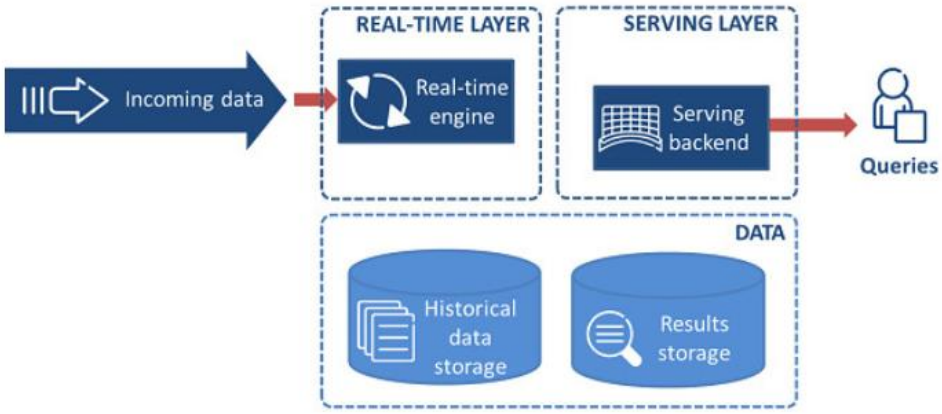
Η αρχιτεκτονική κ είναι κατάλληλη για μοντέλα επεξεργασίας όπου η σειρά των γεγονότων και ερωτημάτων δεν είναι προκαθορισμένη. Είναι κατάλληλη για απλά και ελαφριά συστήματα τα οποία εστιάζουν στην ταχύτητα επεξεργασίας και εξυπηρέτησης, χωρίς να δίνουν τεράστιο βάρος στην ακρίβεια των αποτελεσμάτων. Τα συστήματα επεξεργασίας συνεχούς ροής δεδομένων μπορούν να αλληλοεπιδράσουν με την βάση δεδομένων ανά πάσα στιγμή. Επίσης η αρχιτεκτονική αυτή βρίσκει εφαρμογή σε μοντέλα όπου χρειάζεται χειρισμός αποθηκευτικού χώρου της τάξης των terabyte για κάθε κόμβο του συστήματος έτσι ώστε να μπορέσει να υποστηριχτεί αναπαραγωγή (replication). Ακόμη καθώς και σε συστήματα όπου υπάρχουν πολλά γεγονότα και ερωτήματα τα οποία περιμένουν να εξυπηρετηθούν σε ένα κατανεμημένο σύστημα αρχείων.

Πλεονεκτήματα

Τα πλεονεκτήματα της αρχιτεκτονικής κ είναι η απαίτηση επανεπεξεργασίας των δεδομένων μόνο όταν ο κώδικας αλλάξει, η δυνατότητα οριζόντιας κλιμάκωσης, η απαίτηση λιγότερων πόρων για εφαρμογή μηχανικής μάθησης αφού αυτή γίνεται σε πραγματικό χρόνο λόγω της απουσίας του batch layer. Ακόμη είναι πολύ απλούστερη στην υλοποίηση αφού δεν απαιτεί υλοποίηση 2 ετερογενών συστημάτων όπως με την αρχιτεκτονική λ. Αυτό έχει ως αποτέλεσμα η διαδικασία ανάπτυξης, debugging καθώς και συντήρησης του κώδικα να καθίσταται πολύ πιο απλή.

Μειονεκτήματα

Τα μειονεκτήματα που αντιμετωπίζει η αρχιτεκτονική κ είναι η πιθανότητα μη εξασφάλισης μεγάλης ακρίβειας των δεδομένων λόγω της απουσίας του batch layer το οποίο εξασφαλίζει μία περαιτέρω ακρίβεια στην επεξεργασία των δεδομένων. Ακόμη η μη χρησιμοποίηση του batch layer καθιστά την αρχιτεκτονική κ ανίκανη στο να διαχειριστεί εφαρμογές που έχουν υψηλές απαιτήσεις σε υπολογισμούς όπως επίσης δυσκολεύει και την διαδικασία επεξεργασίας των δεδομένων σε παρτίδες (batch processing).



Σχήμα 1.2 Αρχιτεκτονική κ

Πηγή: *https://www.ericsson.com/en/blog/2015/11/data-processing-architectures--lambda-and-kappa*

2.1.3 Σύγκριση Αρχιτεκτονικών Cloud Computing

	ΑΡΧΙΤΕΚΤΟΝΙΚΗ Λ	**ΑΡΧΙΤΕΚΤΟΝΙΚΗ Κ**
Αρχιτεκτονική επεξεργασίας	Batch – Streaming	Streaming
Recomputing Throughput	Κάθε batch cycle	Όταν απαιτείται λόγω αλλαγής κώδικα
Αξιοπιστία δεδομένων	Αυξημένη αξιοπιστία λόγω batch layer	Προσεγγιστική αλλά συνεπής
Κόστος πόρων	Δύο επίπεδα επεξεργασίας με αυξημένες απαιτήσεων σε πόρων	Ένα επίπεδο επεξεργασίας με μειωμένες απαιτήσεις πόρων
Κόστος συντήρησης	Υψηλό κόστος συντήρησης και διαχείρισης 2 επιπέδων	Χαμηλό κόστος συντήρησης ενός μόνο επιπέδου
Ανάπτυξη και testing	Απαιτητική ανάπτυξη και testing λόγω των 2 διαφορετικών συνόλων κώδικα	Απλούστερη ανάπτυξη και testing ενός μόνο επιπέδου
Εφαρμογές	Σύνθετα συστήματα επεξεργασίας μεγάλης ακρίβειας και υπολογιστικής ισχύς	Απλούστερα streaming συστήματα ελαφριάς υπολογιστικής ισχύς

Πίνακας 1 - Σύγκριση Αρχιτεκτονικών Cloud Computing

2.2 Edge Computing

Με την συνεχή ανάπτυξη των πεδίων εφαρμογής του Internet of Things, τα συστήματα cloud computing αντιμετωπίζουν διάφορα προβλήματα. Η επεξεργασία του τεραστίου όγκου δεδομένων που παράγονται από δισεκατομμύρια συσκευές IoT καθημερινά αποτελεί μεγάλη πρόκληση για τα συστήματα αυτά. Η συνεχής και αυξανόμενη χρήση του cloud ως μέσω επεξεργασίας και αποθήκευσης των δεδομένων που παράγουν τα IoT συστήματα οδηγεί στην αύξηση των χρόνων απόκρισης καθώς και τον φόρτο εργασίας (work load) των server και επιβαρύνουν τα δίκτυα, τα οποία ήδη περιορίζονται από την έλλειψη φάσματος (low Spectral Efficiency) [7]. Για την επεξεργασία των δεδομένων που παράγονται, απαιτείται η μεταφορά τους στο cloud μέσω του διαδικτύου το οποίο δεν είναι ικανό να ανταπεξέλθει στην διαχείριση των μεγάλων αυτών δεδομένων. Η διαδικασία μεταφοράς των δεδομένων στο cloud είναι εξαιρετικά ακριβή σε ενέργεια, χρόνο, εύρος ζώνης και δεν εξασφαλίζει πλήρη ασφάλεια των δεδομένων καθώς και απόλυτη ιδιωτικότητα στους χρήστες. Η ανάγκη αντιμετώπισης των προβλημάτων αυτών που αντιμετωπίζουν οι αρχιτεκτονικές cloud οδήγησε στην ανάπτυξη μιας νέας αρχιτεκτονικής επεξεργασίας μεγάλου όγκου δεδομένων, την αρχιτεκτονική edge computing.

Με τον όρο edge computing αναφερόμαστε σε μία κατανεμημένη αρχιτεκτονική επεξεργασίας δεδομένων όπου η υπολογιστική επεξεργασία των δεδομένων γίνεται τοπικά, κοντά στις πηγές παραγωγής τους, παρά σε κάποιο κεντρικό απομακρυσμένο σύστημα στο cloud [8]. Συγκεκριμένα η επεξεργασία των δεδομένων εκτελείτε είτε σε edge nodes (routers, switches, local datacenters) είτε σε edge devices δηλαδή κάθε έξυπνη συσκευή ή μέρος του IoT συστήματος. Η αρχιτεκτονική edge computing δεν έχει στόχο να αντικαταστήσει τις cloud computing αρχιτεκτονικές, αλλά να τις πλαισιώσει με στόχο την δημιουργία αποδοτικών και κλιμακούμενων IoT συστημάτων. Η αρχιτεκτονική αυτή έρχεται να προσθέσει ένα ενδιάμεσο επίπεδο μεταξύ των πηγών παραγωγής των δεδομένων και του cloud δημιουργώντας την αρχιτεκτονική 3 επιπέδων (3-tier) που υποστηρίζουν τα σημερινά IoT συστήματα. Η δυνατότητα μετακίνησης της λειτουργικότητας επεξεργασίας των δεδομένων στις πηγές παραγωγής τους αναβαθμίζει σημαντικά τα IoT συστήματα δίνοντας λύση στο πρόβλημα συμφόρησης των δικτύων, που παρατηρείται κατά την μετακίνηση των δεδομένων στο cloud καθώς και μείωση των χρόνο απόκρισης των δικτύων κάνοντας παράλληλα εξοικονόμηση πόρων, ενέργειας και χρόνου.

Στην παρακάτω ενότητα θα μελετηθούν 3 είδη αρχιτεκτονικών edge computing:

- Cloudlets
- Fog computing
- Mobile edge computing

2.2.1 Cloudlet

Με τον όρο Cloudlet αναφερόμαστε σε ένα ενισχυμένο μικρής κλίμακας data center το οποίο είναι τοποθετημένο στην άκρη του δικτύου (edge) το οποίο χρησιμοποιεί εικονικές μηχανές (VM) και εξυπηρετεί τις διάφορες πηγές παραγωγής δεδομένων όπως κινητές και έξυπνες συσκευές οι οποίες αναθέτουν το φόρτο επεξεργασίας στο cloudlet [9]. Συγκεκριμένα αποτελεί ένα αποκεντρωμένο ευρέως διασκορπισμένο σύνολο υπολογιστών συνδεδεμένο στο διαδίκτυο και μπορεί να χαρακτηριστεί ως ένα data center σε κουτί όπου ο σκοπός του, όντας το μεσαίο επίπεδο της αρχιτεκτονικής 3 επιπέδων (mobile devices – cloudlet – cloud), είναι να φέρει πιο κοντά τα 2 άλλα επίπεδα. Η τεχνολογία του cloudlet αναπτύχθηκε στο Carnegie Mellon University [10] και είναι σχεδιασμένη να διαχειρίζεται απαιτητικές σε πόρους εφαρμογές καθώς και αφαίρεση του υπολογιστικού φόρτου από τα δίκτυα και τα κεντρικά data center, διατηρώντας μέρος της επεξεργασίας των δεδομένων κοντά στις πηγές παραγωγής τους. [11] Οι συσκευές των χρηστών (user equipment) αποκτούν πρόσβαση στους πόρους των κοντινών Cloudlets μέσω μιας γρήγορης ασύρματης wireless local area σύνδεσης με ένα μόνο hop.

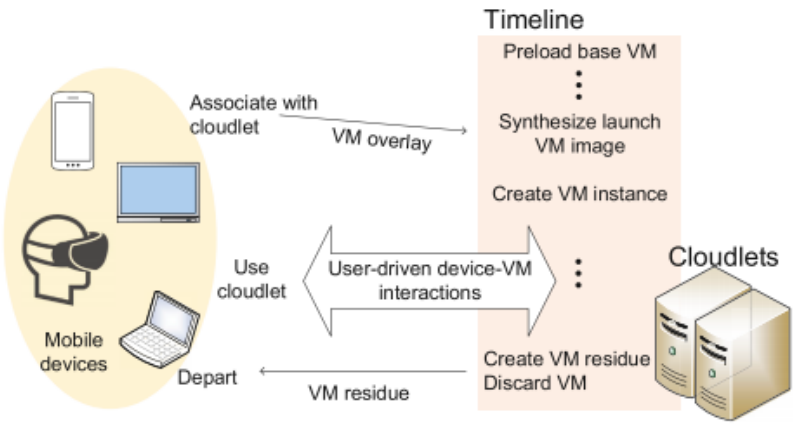
Τα κύρια χαρακτηριστικά της αρχιτεκτονικής με χρήση Cloudlet είναι:

- **Χρήση Cloudlet ως στοιχεία εκφόρτωσης (VM synthesis):** Κύριο χαρακτηριστικό της αρχιτεκτονικής είναι ότι τα cloudlets είναι stateless. Επικοινωνία των συσκευών με το cloud (central core) χρειάζεται μόνο για το αρχικό configuration και όχι κατά την διάρκεια εκφόρτωσης. Μια κινητή συσκευή κατά την εκφόρτωση υπολογιστικού έργου επικοινωνεί με το κοντινότερο cloudlet το οποίο περιέχει μια βασική εικονική μηχανή (base VM) την οποία μπορεί να προμηθευτεί από το cloud. Για να μπορέσει μια ειδική εφαρμογή να τρέξει στο cloudlet χρειάζεται η βασική του εικονική μηχανή να τροποποιηθεί σε μία προσαρμοσμένη με βάση τις ανάγκες της εφαρμογής, εικονική μηχανή. Οι διαφορές της βασικής εικονική μηχανής που διαθέτει το cloudlet και της επιθυμητής από την εφαρμογή εικονικής μηχανής, ονομάζεται VM overlay και υπάρχει σε μορφή zip. Έτσι μόνο ένα VM overlay χρειάζεται να μεταφερθεί στο cloudlet από την κινητή συσκευή, το οποίο θα τροποποιήσει κατάλληλα την βασική εικονική μηχανή του cloudlet. Η διαδικασία αυτή ονομάζεται VM synthesis. Το cloudlet, αφού λάβει το VM overlay το αποσυμπιέζει και το εφαρμόζει πάνω στη βασική του εικονική μηχανή δημιουργώντας το επιθυμητό VM. Τότε η συσκευή αρχίζει την εκφόρτωση στο τροποποιημένο VM και στο τέλος της διαδικασίας το νέο αυτό VM καταστρέφεται αλλά το VM image διατηρείται σε μία μνήμη cache για πιθανή μελλοντική χρήση. Τέλος το cloudlet δημιουργεί ένα κατάλοιπο (VM residue) το οποίο στέλνει πίσω στην συσκευή για να ενσωματωθεί με το VM overlay της συσκευής και να το ενισχύσει για μελλοντικές εκφορτώσεις [8]. Η διαδικασία του VM synthesis είναι εξαιρετικά χρήσιμη σε περιπτώσεις όπου έχουμε αναξιόπιστα δίκτυα ή περιορισμένες πηγές ενέργειας στα cloudlet. Έτσι εάν έχουμε απώλεια σύνδεσης της συσκευής με ένα cloudlet, μπορεί να εντοπιστεί γρήγορα ένα άλλο κοντινό cloudlet και με την μέθοδο του VM synthesis να αρχίσει ξανά η διαδικασία εκφόρτωσης πολύ γρήγορα. Έτσι υπάρχει δυνατότητα για γρήγορη αντικατάσταση των πόρων καθώς και δυναμική τροποποίηση νέων πόρων με

βάσει τις ανάγκες μιας κινητής συσκευής. Επίσης ένα cloudlet μπορεί να υποστηρίξει πολλά ξεχωριστά VM για την εξυπηρέτηση διάφορων συσκευών, ενισχύοντας έτσι την υπολογιστική χωρητικότητα του καθώς και επίτευξη οικονομίας σε θέματα ενέργειας, παρέχοντας παράλληλα ευκολία στην διαδικασία εκφόρτωσης.

- **Χρήση Cloudlet με Thin/Fat clients:** Μέχρι τώρα έχουμε αναφέρει τα cloudlet ως μέσω επεξεργασίας και διαχείρισης απαιτητικών σε πόρους εφαρμογών. Τα τελευταία όμως χρόνια παρατηρείται μία μεγάλη εξέλιξη όσον αφορά την απόδοση συσκευών που περιέχουν ενσωματωμένες CPU, μνήμες, μπαταρίες κτλ. Για τον λόγο αυτό υπάρχει η δυνατότητα επεξεργασίας μέσω των cloudlet με την χρήση των thin clients αλλά και η δυνατότητα χρησιμοποίησης των ιδίων των συσκευών για το μεγαλύτερο μέρος της επεξεργασίας με την χρήση των fat clients.
  Thin clients: Χρησιμοποιείται σε εφαρμογές οι οποίες δεν μπορούν να χρησιμοποιηθούν από κινητές συσκευές αφού προϋποθέτουν εκτέλεση πολύ απαιτητικών σε πόρους αλγορίθμων και μεγάλες βάσεις δεδομένων. Παραδείγματα τέτοιων εφαρμογών είναι εφαρμογές αναγνώρισης προσώπων (face recognition), φωνής και εκτέλεσης εντολών (intelligent virtual assistance), μετατροπής φωνής σε κείμενο (voice to text), επεξεργασία φυσικής γλώσσας, augmented reality.
  Fat clients: Αποτελούν συσκευές οι οποίες είναι ικανές για την διαχείριση απαιτητικών εφαρμογών και επιτρέπουν στην επεξεργασία να γίνεται στην συσκευή και όχι στον server. Μια εφαρμογή που δουλεύει με fat clients είναι το Instagram. Όταν ένας χρήστης του Instagram τραβήξει μία φωτογραφία ή ένα βίντεο με σκοπό την κοινοποίηση του στο κοινωνικό δίκτυο, η διαδικασία τροποποίησης του με φίλτρα γίνεται απευθείας στην συσκευή και μόνο ένα πολύ μικρό μέρος στο cloud. Έτσι μία εφαρμογή μπορεί να σπάσει σε τμήματα όπου αρχικά να εκτελείτε μερική επεξεργασία απευθείας στην συσκευή και έπειτα στο cloud.

- **Χρήση VM στα Cloudlet:** Η χρήση των εικονικών μηχανών στα cloudlet επιτρέπει τον διαχωρισμό των εφαρμογών καθώς και την αποφυγή του σύνθετου προβλήματος της προ εγκατάστασης λογισμικού στο cloudlet για την εξυπηρέτηση κινητών συσκευών. Αντιθέτως η μεταφορά ενός προσαρμοσμένου VM απλοποιεί το πρόβλημα αυτό καθώς εγκλείει το περιβάλλον λογισμικού από τις συσκευές (guest) στην υποδομή του cloudlet (host) δημιουργώντας μια σταθερή διεπαφή (interface) μεταξύ guest και host. Αυτό εξασφαλίζει μεγαλύτερη πιθανότητα συμβατότητας μεταξύ cloudlet και κινητής συσκευής. Η προσέγγιση αυτή είναι πιο γενική από άλλες τεχνικές virtualization αφού δεν απαιτεί οι εφαρμογές να είναι γραμμένες σε συγκεκριμένες γλώσσες προγραμματισμού όπως Java ή C#. Το μεγάλο μέγεθος των VM προϋποθέτει αποθήκευση τους σε μνήμες cache στο cloudlet όπως προαναφέρθηκε για την επανασύνδεση των συσκευών μελλοντικά. Επίσης ένα VM μπορεί να χρησιμοποιηθεί για να εξυπηρετήσει πολλές κινητές συσκευές ταυτόχρονα.



Σχήμα 1.3 VM synthesis

Πηγή: *Edge computing technologies for Internet of Things: a primer Yuan Ai, Mugen Peng *, Kecheng Zhang*

Εφαρμογές

Τα Cloudlet βρίσκουν εφαρμογή σε περιπτώσεις όπου υπάρχει ανάγκη για μείωση του χρόνου απόκρισης (response time) καθώς και παροχή υπολογιστικών πόρων τους οποίους δεν μπορεί να παρέχει μία IoT κινητή συσκευή. Συγκεκριμένα εφαρμογές επαυξημένης πραγματικότητα (Augmented Reality) οι οποίες απαιτούν μεγάλο υπολογιστικό φόρτο όπως επίσης και εφαρμογές cloud gaming όπου το rendering πρέπει να γίνεται σε τοπικό επίπεδο και όχι στο cloud ώστε να εξασφαλιστεί μείωση του χρόνου απόκρισης. Ακόμη η χρήση τους είναι κατάλληλη για φορετά συστήματα γνωστικής βοήθειας πχ Google Glass, όπου απαιτείται επεξεργασία βασισμένη στο cloud, την οποία μπορούν να προσφέρουν τα Cloudlet σε τοπικό επίπεδο μέσω PaaS.

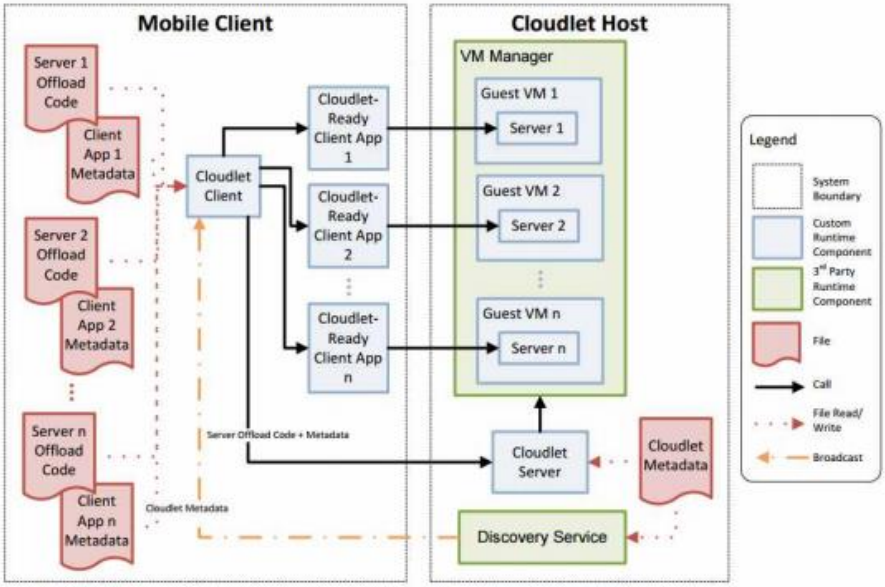
Πλεονεκτήματα

Τα cloudlet χαρακτηρίζονται από αυτοδιαχείριση και έχουν λιγοστές απαιτήσεις ενέργειας και εύρους ζώνης. Αποτελούνται από ομάδες πολυπύρηνων υπολογιστών με εσωτερική συνδεσιμότητα και μεγάλου εύρους wireless LAN. Η χρήση των cloudlet μπορεί να μειώσει τους χρόνους απόκρισης μέχρι και 51% καθώς και τα ποσοστά κατανάλωσης ενέργειας μέχρι 42% σε μία κινητή συσκευή σε σχέση με το cloud. Αποτελούν μια φυσική αναπαράσταση του cloud η οποία είναι λιγότερο επιρρεπής σε σφάλματα και κυβερνοεπιθέσεις προσφέροντας μεγαλύτερη διαθεσιμότητα σε μη ιδανικά περιβάλλοντα. Ακόμη τα cloudlet διαθέτουν υποστήριξη για αλληλεπίδραση σε πραγματικό χρόνο, επίγνωση τοποθεσίας καθώς και παροχή μεγάλου αριθμού από nodes το οποίο δίνει την δυνατότητα για κατανεμημένη γεωγραφικά διανομή δεδομένων στις κινητές συσκευές μέσω ελαχίστων hops.

Επίσης υποστηρίζουν διάφορα πρωτόκολλα ασύρματης επικοινωνίας όπως WLAN, WiFi, 3G, 4G, ZigBee, καθώς και ενσύρματες επικοινωνίες χωρίς να εξαρτώνται από την ποιότητα του κεντρικού δικτύου.

Μειονεκτήματα

Σε ορισμένες περιπτώσεις όμως, τα cloudlet υστερούν σε υπολογιστική δύναμη καθώς και σε δυνατότητες αποθήκευσης για αυτό και πρέπει να υπάρχει συνεργασία με το cloud για να ανταπεξέλθουν στις αυξημένες απαιτήσεις.



Echeverria, et al., "On-Demand VM Provisioning for Cloudlet-Based Cyber-Foraging in Resource-Constrained Environments," MobiCASE 2014.

Σχήμα 1.4 Cloudlet

2.2.2 Fog Computing

Ο όρος Fog computing προτάθηκε από τον καθηγητή Jonathan Bar-Magen Numhauser το 2011 καθώς και από την Cisco αργότερα για την υποστήριξη του IoT. Η κοινοπραξία της OpenFog ορίζει το fog computing ως μία οριζόντια αρχιτεκτονική η οποία κατανέμει πόρους, υπηρεσίες επεξεργασίας και αποθηκευτικό χώρο από το cloud στις τελικές συσκευές των χρηστών. Ένα fog περιβάλλον είναι ένα ενδιάμεσο περιβάλλον μεσαίας επεξεργαστικής δυνατότητας μεταξύ cloud και τελικών χρηστών το οποίο επιτρέπει την επεξεργασία μεγάλων δεδομένων φέρνοντας τις υπηρεσίες πιο κοντά στους χρήστες, μειώνοντας τον χρόνο απόκρισης καθώς και το κόστος μεταφοράς των δεδομένων στο cloud με την χρήση gateways, access points, και routers [12]. Η αρχιτεκτονική του fog computing είναι μια συμπληρωματική αρχιτεκτονική του cloud computing και είναι σχεδιασμένη ώστε να παρέχει μια κατανεμημένη εφαρμογή όπου οι τελικές συσκευές (edge devices) εκτελούν την διαδικασία επεξεργασίας οδηγώντας στην αποκέντρωση των data center. Η Fog Computing αρχιτεκτονική συνεπάγεται κατανομή της επικοινωνίας, επεξεργασίας, πόρων αποθήκευσης και υπηρεσιών σε ή κοντά σε συσκευές και συστήματα των τελικών χρηστών. Κάθε συσκευή με συγκεκριμένη υποδομή μπορεί να λειτουργήσει σαν fog κόμβος παρέχοντας υπηρεσίες επεξεργασίας, αποθήκευσης και δικτύωσης εστιάζοντας σε IoT εφαρμογές οι οποίες είναι ευαίσθητες ως προς τα δεδομένα και χρόνο. Οι κόμβοι του fog είναι επίσης ενωμένοι με το cloud για την διαχείριση πολύπλοκων υπολογισμών και παροχή μεγαλύτερων δυνατοτήτων αποθήκευσης, όπου αυτή απαιτείται, συνθέτοντας το IoT-Fog-cloud framework ή Fog to Cloud (F2C).

Καθώς η τεχνολογία του Fog computing αρχίζει να αναπτύσσεται, έχουν προταθεί διάφορες layer-based, hierarchical και network-based αρχιτεκτονικές από πολλούς ερευνητές. Η πρώτη αρχιτεκτονική που προτάθηκε ήταν από τον Bonomi το 2012 ο οποίος όρισε το fog layer ως ένα κατανεμημένο επίπεδο μεταξύ του κεντρικού δικτύου και των αισθητήρων-συσκευών. Ακολούθησαν πολλές άλλες Fog computing αρχιτεκτονικές οι οποίες θα μελετηθούν πιο κάτω [13]:

- **Fog layered αρχιτεκτονική:** Προτάθηκε από τον Aazam και αποτελεί μια αρχιτεκτονική 6 επιπέδων. Το κατώτερο επίπεδο είναι το physical and virtualization layer το οποίο είναι υπεύθυνο για την διαχείριση των φυσικών και εικονικών κόμβων καθώς και αισθητήρων. Το the Monitoring layer επιβλέπει το δίκτυο και τις λειτουργίες των κόμβων και είναι υπεύθυνο για την ανάθεση εργασιών στους κόμβους καθώς και το πότε θα εκτελεστούν. Επίσης επιβλέπει την κατανάλωση ενέργειας λόγω των ενεργειακών περιορισμών των συσκευών. Το επόμενο επίπεδο είναι το pre-processing layer το οποίο προ επεξεργάζεται τα δεδομένα για να εξάγει τα απαραίτητα και χρήσιμα μέρη τους. Το επίπεδο που ακολουθεί είναι το temporary storage layer στο οποίο αποθηκεύονται προσωρινά τα δεδομένα μέχρι την μεταφορά τους στο cloud. Η προστασία και ασφάλεια των δεδομένων εξασφαλίζεται από το security layer το οποίο παρέχει υπηρεσίες κρυπτογράφησης και εξασφάλισης της ιδιωτικότητας. Το τελευταίο επίπεδο είναι το topmost layer το οποίο φορτώνει τα ήδη επεξεργασμένα δεδομένα στο

cloud, με το μεγαλύτερο μέρος της επεξεργασίας να έχει εκτελεστεί στο περιβάλλον του Fog.

Παρόμοια αρχιτεκτονική επιπέδων πρότεινε και ο Arkian ο οποίος όρισε 4 επίπεδα τα Data generator layer, Cloud computing layer, Fog computing layer και Data consumer layer. Στο Data generator layer βρίσκονται οι IoT συσκευές οι οποίες επικοινωνούν με το Cloud computing layer μέσω του Fog computing layer στο οποίο γίνεται η προ επεξεργασία των δεδομένων παρέχοντας μικρούς χρόνους απόκρισης. Το κύριο συστατικό του Fog computing layer είναι ο Fog server ο οποίος τοποθετείται σε τοποθεσίες κοντά στους χρήστες επικοινωνώντας με τις συσκευές μέσω ασύρματων συνδέσεων ενός hop παρέχοντας υπηρεσίες χωρίς την βοήθεια του cloud ή άλλων fog server. Η διαφορά της αρχιτεκτονικής αυτής είναι ότι το Data consumer layer μπορεί να επικοινωνήσει άμεσα και με τα 3 άλλα επίπεδα. Άλλη μια αρχιτεκτονική επιπέδων η οποία αποτελείται από 5 επίπεδα προτάθηκε από τον Dastjerdi. Το υψηλότερο επίπεδο είναι το IoT application layer το οποίο παρέχει τις υπηρεσίες στους τελικούς χρήστες. Ακολουθεί το software-defined resource management layer το οποίο ασχολείται με θέματα επιτήρησης και ασφάλειας. Έπειτα έχουμε το επόμενο επίπεδο το οποίο είναι υπεύθυνο για την διαχείριση των cloud πόρων και υπηρεσιών και ακολουθεί το network layer το οποίο εξασφαλίζει την συνδεσιμότητα μεταξύ των συσκευών. Το τελευταίο επίπεδο είναι περιλαμβάνει τις τελικές συσκευές δηλαδή αισθητήρες, gateways κτλ.

- **Hierarchical Fog αρχιτεκτονική:** Η αρχιτεκτονική αυτή προτάθηκε από τον Giang και ομαδοποιεί τις συσκευές του Fog layer σε 3 διαφορετικούς τύπους ανάλογα με τις δυνατότητες I/O και επεξεργασίας. Έτσι έχουμε τους Edge nodes οι οποίοι παράγουν δεδομένα, τους Input-Output (IO) nodes οι οποίο χαρακτηρίζονται από μικρή επεξεργαστική ισχύ και διατηρούν επικοινωνία με τους Edge nodes και τέλος τους Compute nodes οι οποίοι κατέχουν υπολογιστικού πόρους.

  Άλλη ιεραρχική αρχιτεκτονική προτάθηκε από τον Hosseinpour οποίος διαχωρίζει το Fog computing layer σε 3 βασικά επίπεδα, το κατώτερο επίπεδο, το επίπεδο των gateway Fog node και το επίπεδο των core Fog nodes, τα οποία μπορούν να επεκταθούν σε n αριθμό επιπέδων. Η επεξεργασία και η αποθήκευση μπορεί να εκτελεστεί σε όλα τα επίπεδα εκτός από το κατώτερο όπου περιλαμβάνει τις συσκευές και αισθητήρες.

- **OpenFog αρχιτεκτονική:** Η αρχιτεκτονική αυτή έχει ως στόχο την εκτέλεση των υπολογισμών κοντά στους τελικούς χρήστες για την επίτευξη της μείωσης του χρόνου απόκρισης, εύρους ζώνης κτλ. Τα αιτήματα των χρηστών δρομολογούνται στο κοντινότερο από τους χρήστες σημείο όπου υπάρχουν διαθέσιμοι υπολογιστικοί πόροι χωρίς να απαιτείται συγχρονισμός από το κεντρικό δίκτυο για αυτό και εξασφαλίζονται επικοινωνίες με πολύ μικρούς χρόνους απόκρισης. Αποτελείται από 3 views, Software (Top layer - Application Services, Application Support, Node Management (IB) και Software Backplane), System (Middle layer - Hardware Virtualization μέσω Hardware Platform Infrastructure) και Node (Bottom layer - Protocol Abstraction Layer and Sensors, Actuators, και Control) τα οποία χρησιμοποιούνται ανάλογα για την διαχείριση διαφόρων σεναρίων χρήσης.

- **Fog network αρχιτεκτονική:** Η αρχιτεκτονική αυτή προτάθηκε από τον Intharawijitr ο οποίος επισήμανε τα προβλήματα των fog συσκευών όσον αφορά το latency σε 5G δίκτυα. Στην αρχιτεκτονική αυτή ο edge router λειτουργεί ως fog server ο οποίος εκτελεί την επεξεργασία των δεδομένων. Ο fog server δεν προωθεί το αίτημα στο κεντρικό δίκτυο εκτός εάν υπάρχει αίτημα για μια cloud υπηρεσία.

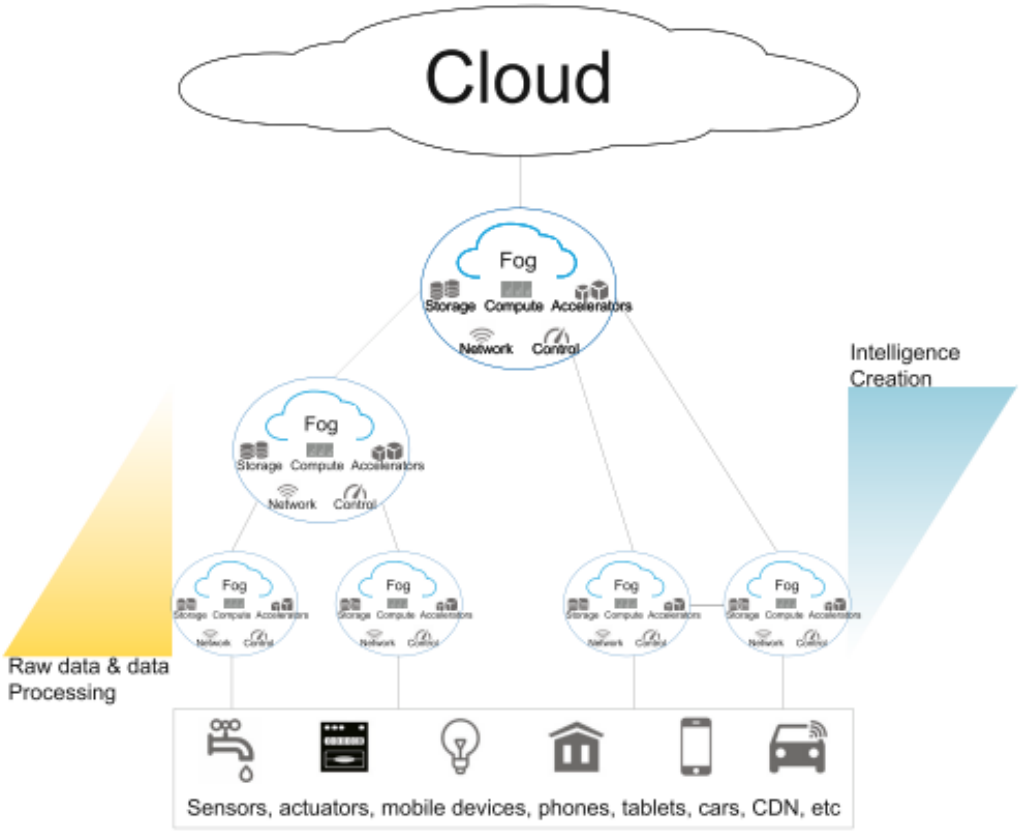
- **Fog architecture for Internet of energy:** Προτάθηκε από τον Baccarelli και προϋποθέτει σύνδεση όλων των fog συσκευών (IoT αισθητήρες, smart car, smartphone κτλ.) στον ασύρματο βασικό σταθμό με την χρήση συνδεσιμότητας 2 πλευρών (Fog to Things (F2T) και Things to Fog (T2F)) μέσω TCP/IP συνδέσεων ενός hop. Οι συσκευές οι οποίες είναι συνδεδεμένες με τον ίδιο βασικό σταθμό ομαδοποιούνται στο ίδιο σύνολο και όλοι οι βασικοί σταθμοί θεωρούνται ως fog nodes οι οποίοι ενώνονται με συνδέσεις Fog to Fog (F2F) χρησιμοποιώντας inter-Fog physical wireless. Χρησιμοποιείται virtualization layer το οποίο χρησιμοποιεί αποδοτικά τους περιορισμένους πόρους και δημιουργεί εικονικούς κλώνους των φυσικών μερών. Ο Fog node εξυπηρετεί τα κλωνοποιημένα φυσικά μέρη μέσω ενός εικονικού δικτύου το οποίο επιτρέπει P2P επικοινωνίες μεταξύ κλώνων χρησιμοποιώντας TCP/IP end to end συνδέσεις.

- **Fog computing βασισμένη στο ανθρώπινο νευρικό σύστημα:** Η αρχιτεκτονική αυτή προτάθηκε από τους Sun and Zhang οι οποίοι παρουσίασαν μια fog αρχιτεκτονική βασισμένη στο ανθρώπινο νευρικό σύστημα. Στην αρχιτεκτονική αυτή, το εγκεφαλικό κεντρικό νεύρο θεωρείται το cloud data centre ενώ το Fog computing data centre θεωρείται ως το κεντρικό νεύρο της σπονδυλικής στήλης. Τα περιφερικά νεύρα αναπαριστώντα από τις IoT συσκευές. Αυτά τα 3 κεντρικά νεύρα λειτουργούν με τέτοιο τρόπο έτσι ώστε να συνδέονται με ολόκληρο το σώμα του όλου συστήματος. Η δομή αυτή είναι σχεδιασμένη για να αντικατοπτρίζει πλήρως το ανθρώπινο σώμα όπου ο εγκέφαλος είναι υπεύθυνος για την ολοκλήρωση των εργασιών σε συνεργασία με τα διασκορπισμένα περιφερειακά νεύρα και την σπονδυλική στήλη. Οι έξυπνες IoT συσκευές (τηλέφωνα, αισθητήρες, smartwatch κτλ.) δηλαδή τα περιφερειακά νεύρα είναι κατανεμημένα σε όλο το σύστημα. Έτσι όπως το νεύρο της σπονδυλικής στήλης συνδέει τον εγκέφαλο με τα περιφερειακά νεύρα, έτσι και το Fog computing data centre ενώνει τις IoT συσκευές με τα cloud data center.

- **Αρχιτεκτονική Integrated Fog cloud IoT (IFCIoT):** Η αρχιτεκτονική αυτή η οποία προτάθηκε από τον Munir, περιλαμβάνει μία ομοσπονδία cloud υπηρεσιών για IoT συσκευές μέσω μιας fog υποδομής η οποία αποτελείται από ένα σύνολο εξωτερικών και εσωτερικών cloud server. Οι κόμβοι (fog nodes) αποτελούνται κυρίως από Gateway devices, smart routers, edge servers και base stations στους οποίους γίνεται η επεξεργασία των δεδομένων. Οι κόμβοι αυτοί είναι αυτόνομοι και εξασφαλίζουν υπηρεσίες, ανθεκτικές σε διακοπές. Το fog περιβάλλον μπορεί να βρίσκεται είτε τοπικά σε κάποιο κτήριο, είτε κατανεμημένο γεωγραφικά σε πολλά κτήρια σε διάφορες

περιοχές του IFCIoT συστήματος. Οι IoT συσκευές συνδέονται μέσω ασύρματης σύνδεσης WLAN, WiMAX κτλ. με τους κόμβους (fog nodes) και ολόκληρο το fog συνδέεται με την ομοσπονδία cloud μέσω του κεντρικού δικτύου καθώς και με άλλα fog ασύρματα.

Όλες οι παραπάνω αρχιτεκτονικές επεξεργάζονται τα δεδομένα κοντά στις πηγές παραγωγής (edge) και χρησιμοποιούν τις fog συσκευές σαν προσωρινούς χώρους αποθήκευσης και προ επεξεργασίας των δεδομένων, ενώ βασίζονται στο cloud για μακροχρόνια αποθήκευση και διαχείριση πολύ απαιτητικού υπολογιστικού φόρτου. Οι περισσότερες από τις αρχιτεκτονικές αυτές αναπαριστούν τις fog συσκευές με τις πραγματικές φυσικές συσκευές και όχι με εικονικές συσκευές όπως για παράδειγμα ο Cisco UCS server.



Σχήμα 1.5 Fog Computing

Πηγή: *Edge computing technologies for Internet of Things: a primer Yuan Ai, Mugen Peng *, Kecheng Zhang*

Εφαρμογές

Σε μία fog computing αρχιτεκτονική οτιδήποτε διαθέτει αισθητήρες, υπολογιστική ισχύ και αποθηκευτικό χώρο (smart συσκευές, αυτοκίνητα, κτήρια κτλ.) μπορεί να λειτουργήσει σαν κόμβος (fog node) στο fog δίκτυο. Κάθε κόμβος μπορεί να εξορύξει, αναλύσει, μεταδώσει ή και να επεξεργαστεί μεγάλο όγκο ετερογενών δεδομένων σε πραγματικό χρόνο. Αυτό δημιουργεί ένα μεγάλο σύνολο εφαρμογών για την αρχιτεκτονική του fog computing στον τομέα του IoT. Κάποιες από τις σημαντικότερες εφαρμογές την αρχιτεκτονικής αυτής είναι:

- **Συστήματα οδικού δικτύου:** Κάθε είδους όχημα μπορεί να λειτουργήσει σαν κόμβος στο fog δίκτυο αλληλοεπιδρώντας με άλλα οχήματα, χρησιμοποιώντας αισθητήρες και ασύρματα δίκτυα, για την παροχή υπηρεσιών όπως προηγμένα συστήματα οδικής βοήθειας, αυτόνομη οδήγησης, πλοήγησης καθώς και συστήματα παροχής άμεσης βοήθειας σε έκτακτα περιστατικά ατυχημάτων.

- **Συστήματα ελέγχου τροχαίας:** Έξυπνα συστήματα ελέγχου τροχαίας τα οποία λαμβάνουν διάφορα δεδομένα από κάμερες, φανάρια με αισθητήρες ταχύτητας κτλ. για την αυτοματοποίηση και ρύθμιση της οδικής προτεραιότητας και αποσυμφόρησης όπως επίσης και αντιμετώπιση οδικών παραβάσεων για την αποφυγή πιθανόν ατυχημάτων.

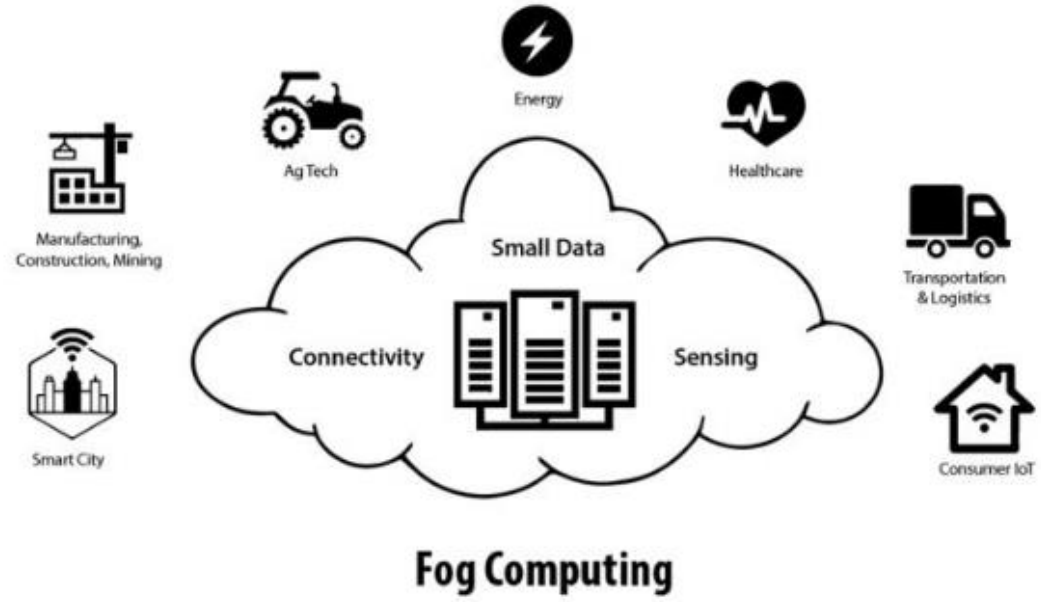
- **Συστήματα παρακολούθησης και ασφάλειας:** Χρήση έξυπνων καμερών και μικροφώνων για την εξασφάλιση της προστασίας ατόμων, αγαθών και χώρων καθώς και έλεγχο εξουσιοδότησης εισόδου εξειδικευμένου προσωπικού σε ιδιωτικούς χώρους. Οι συσκευές των συστημάτων αυτών έχουν δυνατότητες αναγνώρισης προσώπου και φωνής χρησιμοποιώντας σύνθετους αλγορίθμους και παράγοντας τεράστιο όγκο δεδομένων καθημερινά, τον οποίο μία fog computing αρχιτεκτονική είναι ικανή να διαχειριστεί αποδοτικά.

- **Συστήματα ελέγχου κτηρίων και εγκαταστάσεων:** Χρήση πολλών αισθητήρων για την μέτρηση διαφόρων παραμέτρων σε κτήρια και εγκαταστάσεις όπως θερμοκρασία, υγρασία, ποιότητα αέρα, κατανάλωση ενέργειας, διαθεσιμότητα στο χώρο του πάρκινγκ καθώς και παροχή διάφορων αυτοματοποιημένων λειτουργιών όπως αυτόματοι διακόπτες, πόρτες, σαρωτές καρτών, ανελκυστήρες κτλ. Όλα αυτά βοηθούν στην συντήρηση και αναβάθμιση κτηρίων αλλά και στην αποφυγή ή αντιμετώπιση καταστροφών από πυρκαγιές ή σεισμούς.

- **Συστήματα ιατρικής παρακολούθησης:** Τα συστήματα αυτά είναι ικανά μέσω των έξυπνων συσκευών να εντοπίζουν και προβλέπουν καταστάσεις και προβλήματα που μπορεί να προκύψουν σε ασθενείς. Παρακολουθούν μέσω αισθητήρων μέτρησης σε πραγματικό χρόνο, διάφορες παραμέτρους όπως παλμοί καρδιάς, πίεση, κυκλοφορία αίματος, ποσοστά οξυγόνου κτλ. στέλνοντας τα δεδομένα μέσω των fog server σε ιατρικά κέντρα για άμεση διάγνωση και περίθαλψη. Ακόμη τα δεδομένα αυτά μπορούν να χρησιμοποιηθούν ανώνυμα και με ασφάλεια για την ενίσχυση του ερευνητικού τομέα της ιατρικής.

- **Συστήματα γεωργίας και καλλιέργειας:** Χρήση αισθητήρων θερμοκρασίας, υγρασίας, φωτός, αέρα κτλ. για την συλλογή δεδομένων με σκοπό την αντιμετώπιση καταστροφών σοδειών από ακραίες καιρικές συνθήκες. Επίσης δίνεται δυνατότητα λήψης κατάλληλων αποφάσεων και εκτέλεση ενεργειών από μακριά (πχ αυτόματο πότισμα) με βάση τις μετρήσεις διαφόρων παραμέτρων ανά πάσα στιγμή, για την αύξηση των σοδειών. Εξαιρετικά χρήσιμο για χώρες όπου η οικονομία βασίζεται σε μεγάλο βαθμό από την γεωργία [14].

Η σύνθεση των παραπάνω εφαρμογών οδηγεί στην δημιουργία των λεγόμενων έξυπνων πόλεων (smart cities) και κοινοτήτων όπου διάφορα υπο συστήματα επικοινωνούν και αλληλοεπιδρούν μεταξύ τους με σκοπό την εξασφάλιση ασφάλειας, εξοικονόμηση πόρων και ενέργειας καθώς και την αυτοματοποίηση πολλών διαδικασιών για την ενίσχυση του βιοτικού επιπέδου των ανθρώπων.



Σχήμα 1.6 Εφαρμογές του Fog Computing

Πηγή: *https://www.eurekalert.org/pub_releases/2018-12/uoac-fcl120618.php*

Πλεονεκτήματα

Η αρχιτεκτονική του fog computing προσφέρει μεγάλο πλεονέκτημα όσον αφορά τον τομέας της ιδιωτικότητας και της ασφάλειας. Η δυνατότητα ανάλυσης και επεξεργασίας των δεδομένων τοπικά χωρίς την αποστολή τους στο cloud, εξασφαλίζει μεγαλύτερο βαθμό προστασίας από κακόβουλους χρήστες και υποκλοπές δίνοντας την δυνατότητα στις τοπικές ομάδες IT να ελέγχουν ευκολότερα, την ασφάλεια των συσκευών IoT και των δεδομένων που διακινούνται στο τοπικό δίκτυο. Ακόμη εξασφαλίζουν μεγάλη εξοικονόμηση και πλήρη αξιοποίηση τους εύρους ζώνης με αποτέλεσμα την μείωση του κόστους λειτουργίας του συστήματος. Το γεγονός ότι το μεγαλύτερο μέρος της διαχείρισης των δεδομένων γίνεται κοντά στις πηγές παραγωγής τους, δίνει την δυνατότητα επεξεργασίας και ανάλυσης σε πραγματικό χρόνο ικανοποιώντας εφαρμογές οι οποίες είναι απαιτητικές όσον αφορά τον χρόνο απόκρισης. Η ιδιότητα που έχουν οι fog κόμβοι για σύνδεση και αποσύνδεση ανά πάσα στιγμή, καθώς και η δυνατότητα τοποθέτησης τους σε αντίξοα περιβάλλοντα όπως κάτω από την θάλασσα, σε οχήματα, εργοστάσια, κτλ. δίνουν μεγάλη ευελιξία σε ολόκληρη την λειτουργία της fog αρχιτεκτονικής αυξάνοντας παράλληλα και την ποιότητα εξυπηρέτησης.

Μειονεκτήματα

Αν και η αρχιτεκτονική του fog computing παρέχει πολλά πλεονεκτήματα, αντιμετωπίζει κάποια προβλήματα ως προς την λειτουργία της. Παρά το μεγάλο βαθμό προστασίας των δεδομένων που προσφέρει η αρχιτεκτονική αυτή, το γεγονός ότι η επεξεργασία γίνεται παράλληλα σε πολλά μέρη του δικτύου και όχι σε κάποιο κεντρικό σημείο αυξάνει το βαθμό δυσκολίας ελέγχου όλων των σημείων αυτών από τις ομάδες IT, όσον αφορά την ταυτοποίηση και ασφάλεια των συνδεδεμένων συσκευών, άρα και το βαθμό δυσκολίας διατήρησης της προστασίας των δεδομένων. Ακόμη κενά ασφαλείας μπορεί να παρατηρηθούν και σε φυσικό επίπεδο αφού οι κόμβοι (fog nodes), βρίσκονται διασκορπισμένοι γεωγραφικά στο δίκτυο και πιθανόν σε λιγότερο ασφαλή τοποθεσίες από κάποιο κεντρικό data center. Ένα άλλο μειονέκτημα που αντιμετωπίζει η αρχιτεκτονική αυτή είναι αυξημένη πολυπλοκότητα που παρουσιάζεται κατά την σχεδίαση των συστημάτων αυτών. Ένα δίκτυο το οποίο αποτελείται από ένα τεράστιο αριθμό κόμβων οι οποίοι είναι διασκορπισμένοι σε διάφορες τοποθεσίες, συνδέονται και αποσυνδέονται ανά πάσα στιγμή καθώς και αποθηκεύουν, επεξεργάζονται και μεταδίδουν δεδομένα συνεχώς καθιστά την σχεδίαση του πολύπλοκη και δύσκολη [15]. Επίσης η τοποθέτηση των fog server στα κατάλληλα σημεία του δικτύου, για την πλήρη αξιοποίηση των δυνατοτήτων τους αλλά και την ευκολία στην συντήρηση τους, απαιτεί εκτενή μελέτη και σχεδιασμό.

2.2.3 Mobile Edge Computing

Ως Mobile edge computing (MEC) ή Multi-access Edge Computing ορίζεται μια αρχιτεκτονική η οποία αποτελεί κλειδί για την τεχνολογία IoT. Η αρχιτεκτονική αυτή παρουσιάστηκε από τον ETSI (European Telecommunications Standards Institute) ως μια νέα τεχνολογία η οποία παρέχει ένα περιβάλλον IT υπηρεσιών και υπολογιστικών υπηρεσιών παρόμοιων με αυτές του cloud computing κοντά στους χρήστες στις άκρες του δικτύου (mobile network edge) μέσω του RAN (Radio Access Network) [16]. Είναι μια αποκεντρωμένη αρχιτεκτονική η οποία συνδυάζει τεχνολογίες πληροφορικής και τηλεπικοινωνιών προσφέροντας τις υπηρεσίες της μέσω της χρήσης τοπικών κυψελοειδών σταθμών (cellular base stations). Έτσι δημιουργείται ένα περιβάλλον το οποίο χαρακτηρίζεται από χαμηλό latency και πολύ μεγάλο bandwidth πράγμα που επιτρέπει στους χρήστες του δικτύου (wireless subscribers) να έχουν πρόσβαση σε τοπικούς server για τις ανάγκες επεξεργασίας, μέσα στο εύρος του ασύρματου Radio Access Network (RAN).

Radio Access Network (RAN)

Το Radio Access Network (RAN) είναι μέρος της τεχνολογίας των κινητών επικοινωνιών. Τα τηλέφωνα χρησιμοποιούν ραδιοκύματα για την επικοινωνία, μετατρέποντας δεδομένα όπως η φωνή σε ψηφιακά σήματα τα οποία μεταδίδονται μέσων ραδιοκυμάτων. Έτσι για να ενωθεί ένα τηλέφωνο σε κάποιο δίκτυο ή στο διαδίκτυο πρέπει να υπάρξει πρώτα μία σύνδεση σε ένα RAN. Ένα RAN αποτελεί ενδιάμεσο επίπεδο μεταξύ κινητών συσκευών και κεντρικού δικτύου, το οποίο αξιοποιώντας του δέκτες ραδιοκυμάτων παρέχει σύνδεση με το κεντρικό δίκτυο. Το δίκτυο RAN καλύπτει μια μεγάλη γεωγραφική περιοχή η οποία χωρίζεται σε πολλές κυψελίδες (cells), όπου για κάθε κυψελίδα αντιστοιχεί κάποιος base station. Οι σταθμοί αυτοί είναι ενωμένοι μεταξύ τους καθώς και με τον Radio Network Controller (RNC) ή αλλιώς Base Station Controller (BSC) μέσω μικροκυμάτων ή γραμμές σταθερής τηλεφωνίας. Ο Radio Network Controller είναι υπεύθυνος για τον έλεγχο των base station και διαθέτει επίσης λειτουργίες διαχείρισης. Οι RNC είναι ενωμένοι με ένα ή δυο δίκτυα δίνοντας την δυνατότητα στο Radio Access Network να παρέχει σύνδεση μεταξύ κινητών συσκευών και κεντρικού δικτύου.

Μια άλλη παρόμοια τεχνολογία που έχει εμφανιστεί τον τελευταίο καιρό είναι αυτή του Cloud Radio Access Network (CRAN) η οποία αποτελεί μια κεντρική και συνεργατική επεξεργαστική λύση, αποδοτική ως προς την κατανάλωση ενέργειας. Πρακτικά το CRAN συγκεντρώνει όλη την υπολογιστική ισχύ των base station σε ένα κεντρικό σύνολο. Τα σήματα ραδιοσυχνοτήτων τα οποία είναι γεωγραφικά κατανεμημένα, συλλέγονται από απομακρυσμένες κεφαλές και μεταδίδονται στο cloud μέσω ενός οπτικού δικτύου μεταφοράς. Έτσι η χρήση του Cloud Radio Access Network έχει στόχο να μειώσει τις κυψέλες προσφέροντας παράλληλα καλύτερες υπηρεσίες, μείωση του κόστους, χωρίς όμως μείωση της γεωγραφικής κάλυψης.

Διάφορα είδη αρχιτεκτονικών Mobile Edge Computing έχουν προταθεί τα τελευταία χρόνια με σκοπό την ενσωμάτωση λειτουργιών και δυνατοτήτων του cloud στα δίκτυα κινητής τηλεφωνίας. Μερικές MEC αρχιτεκτονικές από αυτές είναι:

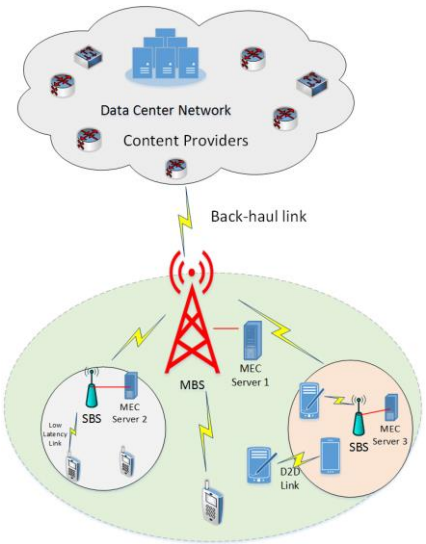
- **Small cell cloud (SCC):** Η αρχιτεκτονική αυτή έχει προταθεί από το Ευρωπαϊκό project TROPIC το 2012. Έχει ως στόχο να ενισχύσει τις μικρές κυψέλες (SCeNBs) με επιπλέον υπολογιστικές και αποθηκευτικές ικανότητες μέσω του network function virtualization (NFV). Τα δίκτυα θα αποτελούνται από μεγάλο αριθμό SCeNBs έτσι η αρχιτεκτονική SCC θα προσφέρει μεγάλη επεξεργαστική ισχύ στη διάθεση των κινητών συσκευών για την κάλυψη εφαρμογών οι οποίες είναι απαιτητικές σε latency. Η ενσωμάτωση του SCC στο κινητό δίκτυο θα γίνει με την βοήθεια του small cell manager (SCM) ο οποίος είναι υπεύθυνος για τον έλεγχο του SCC. Ο SCM θα διαχειρίζεται τους πόρους επεξεργασίας και αποθήκευσης που παρέχουν οι κυψέλες SCeNB οι οποίες μπορούν να ξεκινήσουν και να σταματήσουν την λειτουργία τους ανά πάσα στιγμή. Επίσης θα αποφασίζει για την ανάθεση νέων υπολογιστικών εργασιών ή και συγχώνευση τρεχόντων. Οι υπολογιστικοί πόροι αναπαρίστανται μέσω εικονικών μηχανών οι οποίες βρίσκονται μέσα στις κυψέλες SCeNBs.

- **Mobile micro clouds (MMC):** Η αρχιτεκτονική αυτή προτάθηκε από τον S. Wang. Σε αντίθεση με την SCC η MMC αρχιτεκτονική επιτρέπει στις συσκευές των χρηστών να χρησιμοποιούν τους υπολογιστικούς πόρους ενός μόνο MMC το οποίο είναι συνδεδεμένο απευθείας με ασύρματη σύνδεση σε κάποιο βασικό σταθμό. Έτσι δεν υπάρχει κάποιος ενδιάμεσος μεσολαβητής όπως ο SCM στην SCC, ο οποίος να είναι υπεύθυνος για τον έλεγχο, έτσι εδώ έχουμε έναν πλήρως κατανεμημένο τρόπο ελέγχου. Τα MMC είναι συνδεδεμένα απευθείας με το τελικό δίκτυο εξασφαλίζοντας συνεχή εξυπηρέτηση ακόμα και κατά την συνεχή μετακίνηση των κινητών συσκευών στον δίκτυο.

- **Fast moving personal cloud (MobiScud):** Η αρχιτεκτονική fast moving personal cloud (MobiScud) χρησιμοποιεί τις τεχνολογίες software defined network (SDN) και NFV για να ενσωματώσει cloud λειτουργικότητες στα κινητά δίκτυα. Σε αντίθεση με τις αρχιτεκτονικές SCC και MMC οι υπολογιστικοί πόροι δεν βρίσκονται στις κυψέλες SCeNB και eNB αλλά παρέχονται με κατανεμημένο τρόπο από το cloud κοντά στο εύρος του RAN. Διαθέτει παρόμοια οντότητα διαχείρισης όπως ο SCM της αρχιτεκτονικής SCC, τον MobiScud control (MC) ο οποίος επικοινωνεί με το κινητό δίκτυο, το cloud και τους SDN switches. Ο MC έχει 2 λειτουργικότητες, τον έλεγχο ανταλλαγής μηνυμάτων μεταξύ των στοιχείων στο δίκτυο καθώς και την οργάνωση της κίνησης των δεδομένων μέσω των SDN και συγχώνευση των VM κατά την μετακίνηση των συσκευών μέσα στο δίκτυο.

- **Follow me cloud (FMC):** Παρόμοια ιδέα με αυτή της αρχιτεκτονικής fast moving personal cloud (MobiScud) όπου οι συσκευές μετακινούνται μέσα στο δίκτυο (roaming) και οι υπηρεσίες cloud τις ακολουθούν. Στην αρχιτεκτονική αυτή η υπολογιστική και αποθηκευτική ισχύς βρίσκεται ακόμη πιο μακριά από τις συσκευές των χρηστών σε σύγκριση με τις προηγούμενες αρχιτεκτονικές. Ακόμη το κεντρικό δίκτυο (CN) είναι κατανεμημένο στην περίπτωση αυτή για την αντιμετώπιση του μεγάλου αριθμού κινητών συσκευών. Εδώ έχουμε 2 νέες οντότητες τους DC/GW mapping entity και FMC controller (FMCC) οι οποίες μπορεί να είναι είτε λειτουργικότητες που συνυπάρχουν με τους κόμβους του δικτύου ή λογισμικό που τρέχει στα data center. Το DC/GW mapping entity αντιστοιχεί τα data center με τα κατανεμημένα μέρη του CN λαμβάνοντας υπόψη παραμέτρους όπως τοποθεσία, αριθμός hops κτλ. Ο FMC controller (FMCC) διαχειρίζεται τους πόρους των data center και τις υπηρεσίες αυτών και καθορίζει πιο data center εξυπηρετεί κάθε συσκευή ενός χρήστη.

- **CONCERT:** Η αρχιτεκτονική CONSERT προτάθηκε από τους J. Liu, T. Zhao, S. Zhou, Y. Cheng, και Z. Niu το 2014. Χρησιμοποιεί και αυτή τις τεχνολογίες software defined network (SDN) και NFV για αυτό και οι υπολογιστικοί καθώς και οι αποθηκευτικοί πόροι αναπαρίστανται ως εικονικοί πόροι. Ο έλεγχος των επικοινωνιών των στοιχείων της αρχιτεκτονικής καθώς και η διαχείριση των πόρων γίνεται από τον conductor. Γίνεται χρήση των radio interface equipment (RIEs) οι οποίοι αναπαριστούν σε φυσικό επίπεδο τους eNB, SDN switches, και υπολογιστικούς πόρους. Σε αντίθεση με προηγούμενες αρχιτεκτονικές όπου είχαμε πλήρη κατανομή των πόρων, εδώ έχουμε μια ιεραρχημένη κατανομή η οποία προσφέρει ελαστικότητα στην διαχείριση του δικτύου και των cloud υπηρεσιών. Έτσι εάν οι τοπικοί server οι οποίοι βρίσκονται στους base station και διαθέτουν μικρότερη υπολογιστική δύναμη αδυνατούν να εξυπηρετήσουν τις ανάγκες, γίνεται ανάθεση στο επόμενο επίπεδο ιεραρχίας όπου έχουμε τους περιφερειακούς και κεντρικούς server [17].



Σχήμα 1.7 Mobile Edge Computing

Πηγή: *https://www.sciencedirect.com/science/article/pii/S2352864819300227#bib15*

Εφαρμογές

Η αρχιτεκτονική του Mobile edge computing έρχεται να προσφέρει πολλές σύγχρονες εφαρμογές όπως [18]:

- **Augmented Reality (AR):** Η αρχιτεκτονική MEC θα επηρεάσει θετικά τις υπηρεσίες AR προσφέροντας γρήγορα και τοπικά δεδομένα για την διαμόρφωση της αναπαράστασης του γύρω περιβάλλοντος σε πραγματικό χρόνο. Για παράδειγμα θα υπάρχει η δυνατότητα σε μία ξενάγηση, να χρησιμοποιείται η συσκευή τηλεφώνου όπου με την βοήθεια της τεχνολογίας AR να προσφέρει μία διαδραστική ιστορική πληροφόρηση μέσω επαυξημένης πραγματικότητας.

- **Υπηρεσίες video streaming:** Οι υπηρεσίες video streaming θα έχουν πιο αποδοτική λειτουργία με την χρήση της αρχιτεκτονικής MEC, αφού θα υπάρχει ανάλυση του τοπικού δικτύου και ανάθεση του στους χρήστες, διαμορφώνοντας έτσι την ποιότητα κατάλληλα παρέχοντας μία συνεχή ροή μετάδοσης.

- **Αυτόνομα οχήματα:** Μέσω της αρχιτεκτονικής MEC, υπάρχει η δυνατότητα ανταλλαγής πληροφοριών όσον αφορά την οδική υποδομή (θέση πεζών, αυτοκίνητα, ζώα, καιρικές συνθήκες κτλ.) μεταξύ κινούμενων οχημάτων χωρίς την ανάγκη επικοινωνίας με κάποιο κεντρικό cloud server. Αυτό επιτρέπει στα οχήματα, με την χρήση των τεχνολογιών MEC, Artificial Intelligence (AI) και Machine Learning (ML) να αποκτήσουν δυνατότητες αυτονομίας, έχοντας επίγνωση του γύρω περιβάλλοντος σε πραγματικό χρόνο καθώς θα κινούνται, παίρνοντας τις κατάλληλες αποφάσεις. Καθοριστικό ρόλο στην σωστή λειτουργία αυτής της εφαρμογής παίζουν οι πολύ χαμηλοί χρόνοι απόκρισης που μπορεί να προσφέρει η αρχιτεκτονική αυτή μέσω της τοπικής επεξεργασίας των δεδομένων [19].

- **Cloud gaming - Multiplayer gaming - Mobile gaming:** Η αρχιτεκτονική MEC επιτρέπει την μετακίνηση της απαιτητικής επεξεργασίας γραφικών και απαιτητικών υπολογισμών οι οποίοι εκτελούνται σε πραγματικό χρόνο για την εξυπηρέτηση των online παιχνιδιών, από το cloud σε κοντινότερα προς τους χρήστες δίκτυα. Έτσι οι χρήστες έχουν πρόσβαση σε gaming υπηρεσίες με καλύτερη ποιότητα εμπειρίας οπουδήποτε μέσα στο δίκτυο τους λόγω του χαμηλού latency. Ακόμη προσφέρεται η δυνατότητα ανάπτυξης και στον χώρο του mobile gaming όπου παιχνίδια και εφαρμογές με υψηλές απαιτήσεις σε γραφικά μπορούν να τρέξουν σε συσκευές (smartphones) με χαμηλή επεξεργαστική ισχύ, αναθέτοντας τον φόρτο επεξεργασίας σε κάποιο server στο τοπικό δίκτυο εξοικονομώντας παράλληλα ενέργεια κατά την χρήση της κινητής συσκευής.

- **Video analytics:** Η μεγάλη χρήση των συστημάτων παρακολούθησης σε πόλεις και οργανισμούς καθώς και η αύξηση της ποιότητα του παραγόμενου υλικού οδηγεί στην παραγωγή τεραστίων σε μέγεθος δεδομένων. Με την χρήση της αρχιτεκτονικής MEC υπάρχει δυνατότητα τοπικής επεξεργασίας και ανάλυσης του οπτικοακουστικού υλικού αυτού χωρίς να υπάρχει η ανάγκη να σταλεί σε κάποιο κεντρικό σύστημα μειώνοντας

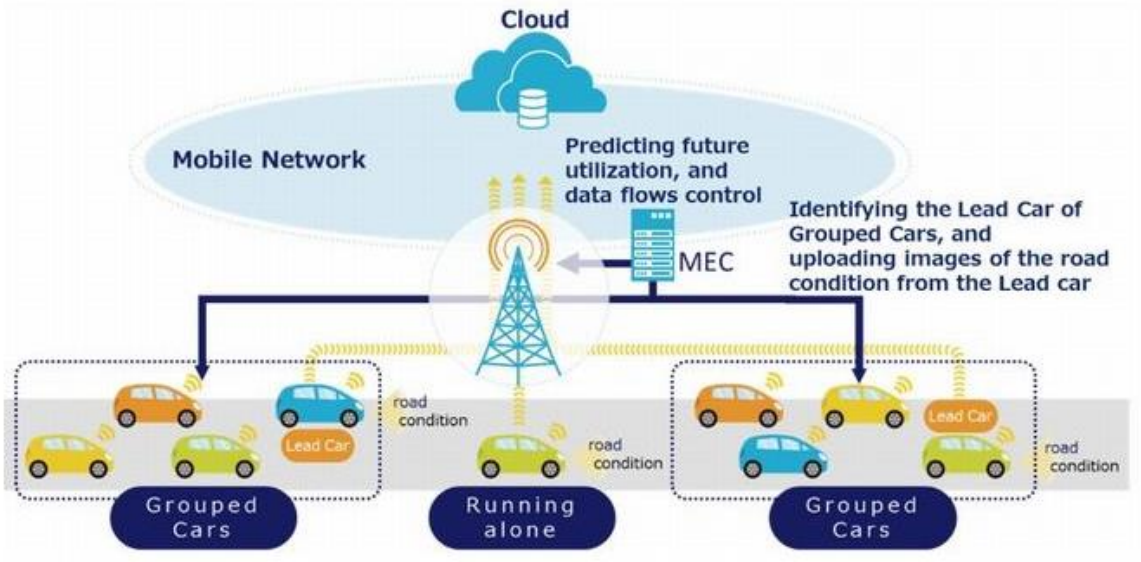
έτσι την κίνηση στο δίκτυο. Έτσι επιτρέπεται η συλλογή, επεξεργασία και ανάλυση υλικού από διάφορες συσκευές σε πραγματικό χρόνο καθώς και η χρήση διάφορων εφαρμογών όπως αναγνώρισης προσώπου, φωνής κτλ. Οι λειτουργίες αυτές πραγματοποιούνται σε τοπικό επίπεδο μειώνοντας σημαντικά το κόστος, χρόνο και χώρο που θα χρειαζόταν η μετακίνηση του μη επεξεργασμένου υλικού στο cloud.

Πλεονεκτήματα

Το κύριο πλεονέκτημα της αρχιτεκτονικής Mobile Edge Computing είναι η μείωση της συμφόρησης στα κινητά δίκτυα. Τα υπάρχοντα 4G δίκτυα έχουν φτάσει στα όρια τους με τον αριθμό των συνδεδεμένων συσκευών να υπολογίζονται στα 100 δισεκατομμύρια μέχρι το 2025. Έτσι η έξυπνη διαχείριση της κίνησης των δεδομένων στο δίκτυο από την επερχόμενη τεχνολογία 5G κρίνεται απαραίτητη και μπορεί να επιτευχθεί με την χρήση του MEC. Ακόμη ένα άλλο μεγάλο πλεονέκτημα που παρουσιάζει η αρχιτεκτονική αυτή είναι η μεγάλη μείωση του latency που μπορεί να επιφέρει στις διάφορες υπηρεσίες που παρέχονται στα δίκτυα, φέρνοντας τα δεδομένα πιο κοντά στους χρήστες τροφοδοτώντας απευθείας με αυτά τις κινητές τους συσκευές. Αυτό οδηγεί στην διασφάλιση μεθόδων απεξάρτησης από τις cloud τεχνολογίες, οι οποίες χρησιμοποιούνταν καθολικά λόγω της έλλειψης υπολογιστικής ισχύς και πόρων στις συσκευές μας. Έτσι τα δίκτυα απαλλάσσονται από τον μεγάλο φόρτο μετακίνησης μεγάλου όγκου δεδομένων στο cloud για επεξεργασίας μεγάλο μέρος του οποίου γίνεται πλέον σε τοπικό επίπεδο.

Μειονεκτήματα

Παράλληλα η αρχιτεκτονική του Mobile edge computing έχει να αντιμετωπίσει κάποιες προκλήσεις όσον αφορά την συμβατότητα των κινητών συσκευών και δικτύων. Οι διάφοροι κατασκευαστές στην IoT βιομηχανία αφήνουν μικρά περιθώρια διαλειτουργικότητας και συμβατότητας με συσκευές άλλων κατασκευαστών. Για παράδειγμα διάφορες συσκευές συνδέονται με διάφορους τρόπους όπως 3G, LTE, Wi-Fi κτλ. καθιστώντας δύσκολη την χρησιμοποίηση ενός καθολικού μοντέλου επικοινωνίας μεταξύ των συσκευών οι οποίες κάνουν χρήση διαφορετικών πρωτοκόλλων επικοινωνίας και ανταλλαγής μηνυμάτων. Ακόμη παρατηρούνται παρόμοια προβλήματα σε θέματα ασφάλειας με την αρχιτεκτονική fog computing, αφού και στην αρχιτεκτονική MEC έχουμε μία κατανεμημένη επεξεργασία δεδομένων στο δίκτυο και όχι κάποιο κεντρικό σύστημα το οποίο μπορεί να προστατευθεί πιο εύκολα. Έτσι απαιτείται μεγαλύτερη προσπάθεια για την εξασφάλιση της προστασίας των δεδομένων κατά την διακίνηση και επεξεργασία σε κάθε κυψέλη και σταθμό του δικτύου, αφού ένα γεωγραφικά κατανεμημένο σύστημα είναι πολύ πιο ευάλωτο σε επιθέσεις από ένα κεντρικό όπως είναι τα cloud computing συστήματα.



Σχήμα 1.8 Εφαρμογή MEC σε έξυπνα οχήματα

Πηγή: *https://www.technative.io/successful-poc-demonstration-of-data-flows-control-function-by-edge-computing/*

2.2.4 Σύγκριση Αρχιτεκτονικών Edge Computing

	CLOUDLET COMPUTING	FOG COMPUTING	MOBILE EDGE COMPUTING
Τοποθεσία Κόμβων	Μεταξύ end devices και cloud	Μεταξύ end devices και cloud	Radio Access Network
Hardware Αρχιτεκτονικής	Μικρά datacentres	Routers, Switches, Access Points, Gateways	Servers σε base stations
Software Αρχιτεκτονικής	Cloudlet Agent	Fog Abstraction Layer	Mobile Orchestrator
Υπολογιστική ισχύς	Υψηλή	Μεσαία	Υψηλή
Context awareness	Χαμηλό	Μεσαίο	Υψηλό
Μηχανισμός Επικοινωνίας	Wi-Fi	Bluetooth, Wi-Fi, Mobile Network	Mobile Network
Επικοινωνία μεταξύ κόμβων	Μερική	Πλήρης	Μερική
Proximity	Ένα Hop	Ένα ή περισσότερα Hop	Ένα Hop

Πίνακας 2 - Σύγκριση Αρχιτεκτονικών Edge Computing

3. Συστατικά Big Data αρχιτεκτονικών και IoT συστημάτων

3.1 Πρωτόκολλα επικοινωνίας και ανταλλαγής δεδομένων

Η τεχνολογία του Internet of Things βασίζεται στην επικοινωνία μεταξύ έξυπνων συσκευών μέσω της ανταλλαγής μηνυμάτων και δεδομένων για την εκτέλεση διαφόρων λειτουργειών και διεργασιών. Το γεγονός ότι μια έξυπνη συσκευή μπορεί να είναι μια κάμερα, ένας μικρός αισθητήρας ή ένα ρολόι δεν αφήνει πολλά περιθώρια όσον αφορά θέματα ενεργειακής κατανάλωσης, πρωτοκόλλων επικοινωνίας και υπολογιστικής ισχύς. Για τον λόγο αυτό δεν μπορούμε να χρησιμοποιήσουμε βαριά πρωτόκολλα επικοινωνίας, τα οποία χρησιμοποιούνται στο διαδίκτυο όπως για παράδειγμα το HTTP. Ακόμη η ετερογένεια στον σχεδιασμό και λειτουργικότητα των συσκευών αυτών, οι οποίες προέρχονται από διαφορετικούς κατασκευαστές, προϋποθέτει χρήση διαφόρων πρωτοκόλλων επικοινωνίας. Τα σημερινά IoT συστήματα ποικίλουν ως προς την φύση και λειτουργία τους, έχοντας διαφορετικές απαιτήσεις στον τρόπο επικοινωνίας και ανταλλαγής δεδομένων. Έτσι η επιλογή και χρήση πρωτοκόλλων τα οποία είναι συντηρητικά ως προς την κατανάλωση ενέργειας και υπολογιστικών πόρων αποτελεί κύρια ανάγκη αλλά και πρόκληση για τα σημερινά IoT συστήματα. Στην παρακάτω ενότητα θα μελετηθούν διάφορα πρωτόκολλα επικοινωνίας και ανταλλαγής δεδομένων.

3.1.1 MQTT (Message Queue Telemetry Transport)

Το MQTT είναι ένα ελαφρύ και απλό πρωτόκολλο ανταλλαγής μηνυμάτων το οποίο χρησιμοποιεί την μέθοδο public/subscribe. Δημιουργήθηκε από τον Dr Andy Stanford-Clark της IBM και τον Arlen Nipper της εταιρείας Arcom το 1999 [20]. Είναι σχεδιασμένο για συσκευές οι οποίες είναι περιορισμένες σε θέματα κατανάλωσης ενέργειας και υπολογιστικών πόρων καθώς και για αναξιόπιστα δίκτυα με υψηλό latency. Εξασφαλίζει εύκολα την επίτευξη επικοινωνίας μεταξύ συσκευών χρησιμοποιώντας πρωτόκολλο TCP, καθιστώντας το πρωτόκολλο αυτό μια πολύ καλή λύση για διάφορες IoT εφαρμογές αφού είναι ιδανικό για χρήση σε περιβάλλοντα με πολλούς περιορισμούς όπως αυτό της επικοινωνίας μεταξύ μηχανών (Machine to Machine M2M).

Η αρχιτεκτονική του πρωτοκόλλου MQTT αποτελείται από 3 στοιχεία:

- **Publishers:** Είναι κάθε αισθητήρας/συσκευή ο οποίος συλλέγει δεδομένα τα οποία διανεμηθούν

.

- **Broker:** Λειτουργεί σαν διαμεσολαβητής και είναι υπεύθυνος για την διανομή των δεδομένων στους Subscribers.

- **Subscribers:** Είναι κάθε εφαρμογή η οποία θέλει να λάβει δεδομένα για την εκτέλεση της λειτουργίας της.

Λειτουργία

Η διανομή των δεδομένων και της πληροφορίας γίνεται μέσω συγκεκριμένης ιεράρχησης, των topics. Τα topics λειτουργούν σαν αναγνωριστικά για την διαχώριση των δεδομένων σε διάφορα θέματα. Όταν ένας publisher έχει δεδομένα προς διανομή, τότε στέλνει τα δεδομένα αυτά σε κάποιο συγκεκριμένο topic στον broker. Ένας subscriber για να λάβει δεδομένα από ένα συγκεκριμένο topic πρέπει να εγγραφεί σε αυτό. Αν ο broker λάβει κάποιο μήνυμα σε κάποιο topic στο οποίο δεν έχει εγγραφεί κάποιος subscriber, τότε απορρίπτει το μήνυμα αυτό, εκτός εάν έχει επισημανθεί ως μήνυμα το οποίο πρέπει να αποθηκευτεί (retained message) και να σταλεί μόλις κάποιος subscriber εγγραφεί στο συγκεκριμένο topic. Δεν απαιτείται κάποια αλληλεπίδραση μεταξύ publishers και subscribers. Επικοινωνούν με τον broker ο οποίος λειτουργεί σαν διαμεσολαβητής, χωρίς να γνωρίζουν πληροφορίες ο ένας για τον άλλο. Κάθε client επικοινωνεί με μόνο ένα broker αλλά σε ένα σύστημα ενδέχεται να υπάρχουν πολλοί broker για την ανταλλαγή δεδομένων.

Πλεονεκτήματα

Ο σχεδιασμός και ο τρόπος λειτουργίας του πρωτοκόλλου MQTT προσφέρει πολλά πλεονεκτήματα στην επικοινωνία μεταξύ των συσκευών σε ένα IoT σύστημα. Αρχικά είναι ένα αποδοτικό πρωτόκολλο επικοινωνίας και ανταλλαγής μηνυμάτων τον οποίο είναι πολύ γρήγορο και εύκολο ως προς την υλοποίηση του λόγω της αυξημένης απλότητας που χαρακτηρίζει τον σχεδιασμό και την λειτουργία του. Η χρήση μικρών πακέτων για την μετάδοση των δεδομένων (2bytes – 256Mbytes), βοηθά στην μείωση της χρησιμοποίησης του δικτύου καθώς και στην γρήγορη και αποδοτική μετάδοση των μηνυμάτων. Ακόμη το πρωτόκολλο αυτό βοηθά στην εξοικονόμηση ενέργειας αφού δεν απαιτεί υψηλή κατανάλωση ενεργειακών πόρων από τις συσκευές για την λειτουργία του καθώς και στην μείωση των απαιτήσεων σε εύρος ζώνης στο δίκτυο.

Μειονεκτήματα

Παρά το γεγονός ότι το πρωτόκολλο MQTT αποτελεί ένα από τα κυριότερα πρωτόκολλά επικοινωνίας και ανταλλαγής δεδομένων σε IoT συστήματα, αντιμετωπίζει κάποια μειονεκτήματα. Το πρωτόκολλο MQTT μεταδίδει με πιο αργό ρυθμό σε σχέση με το πρωτόκολλο CoAP. Παρά το γεγονός ότι αποτελεί ένα πρωτόκολλο μικρής ενεργειακής απαίτησης, η χρήση του TCP μπορεί να οδηγήσει σε αύξηση της κατανάλωσης ενέργειας λόγω της συνεχούς διατήρησης των sockets καθώς και της διαδικασίας handshake. Ένα άλλο μειονέκτημα που αντιμετωπίζει το πρωτόκολλο MQTT, είναι το γεγονός ότι χρησιμοποιεί ένα μη σταθερό σύστημα (topics-subscriptions) σε σχέση με πρωτόκολλα όπως το CoAP (URI), όσον αφορά τον διαχωρισμό των δεδομένων. Σε γενικές γραμμές το πρωτόκολλο MQTT δεν μπορεί εύκολα να κλιμακωθεί λόγω των brokers, και δεν μπορεί να υποστηρίξει βαριές διανομές δεδομένων αφού αποτελεί ένα πολύ ελαφρύ πρωτόκολλο ανταλλαγής μηνυμάτων.

3.1.2 CoAP (Constrained Application Protocol)

Το πρωτόκολλο CoAP (Constrained Application Protocol), είναι ένα πρωτόκολλο επικοινωνίας Machine to Machine, το οποίο υποστηρίζει αρχιτεκτονικές Publish/Subscribe και Resource/Observe και είναι παρόμοιο με το HTTP πρωτόκολλο. Σχεδιάστηκε από το IETF CoRE Working Group. Είναι κατάλληλο για την επικοινωνία μεταξύ συσκευών περιορισμένων δυνατοτήτων σε κατανάλωση ενέργειας και υπολογιστικών πόρων με το διαδίκτυο, καθώς και δικτύων τα οποία χαρακτηρίζονται από χαμηλό εύρος ζώνης και διαθεσιμότητα. Επίσης επιτρέπει την επικοινωνία μεταξύ συσκευών που ανήκουν σε διαφορετικά δίκτυα τα οποία είναι συνδεδεμένα στο διαδίκτυο, όπως επίσης και για την αποστολή SMS σε δίκτυα κινητής τηλεπικοινωνίας. Το CoAP έχει παρόμοιο τρόπο λειτουργίας με το HTTP και υποστηρίζει το REST στυλ αρχιτεκτονικής, έχοντας την δυνατότητα να μεταφράζεται εύκολα σε HTTP, διατηρώντας παράλληλα τις απαιτήσεις των IoT συστημάτων για multicast, απλότητα και χαμηλή επιβάρυνση στο δίκτυο. Σε αντίθεση με το πρωτόκολλο MQTT δεν χρησιμοποιεί topics και TCP για την επικοινωνία, αλλά Uniform Resource Identifier (URI) και UDP αντίστοιχα.

Λειτουργία

Η λειτουργία του πρωτοκόλλου CoAP βασίζεται στο πρότυπο του HTTP, δηλαδή ενός client-server πρωτοκόλλου όπου ο client στέλνει αιτήματα στον server και ο server στέλνει πίσω την απάντηση, σχεδιασμένο πάνω στις ανάγκες ενός IoT συστήματος. Οι client μπορούν να χρησιμοποιήσουν αντίστοιχα αιτήματα με το HTTP πρωτόκολλο, δηλαδή τα GET, PUT, POST and DELETE αιτήματα, στα ανάλογα URI (Unique Universal Resource Identifier) που αντιστοιχούν σε κάθε resource [21]. Οι publisher κοινοποιούν τα δεδομένα που θέλουν να στείλουν στα αντίστοιχα URI, από τα οποία οι subscriber τα λαμβάνουν. Έτσι μόλις κοινοποιηθούν δεδομένα σε κάποιο URI, όλοι οι subscriber ενημερώνονται. Το πρωτόκολλο CoAP αποτελείται από 2 επίπεδα, το Message layer και το request/response layer.

Το Message layer υποστηρίζει 4 είδη μηνυμάτων, τα CON (confirmable), NON (non-confirmable), ACK (Acknowledge) και RST (Reset) για την εξυπηρέτηση των εισερχόμενων αιτημάτων:

- **Confirmable message (CON):** Ένα confirmable message (CON) εξασφαλίζει ότι το αίτημα θα ληφθεί από τον server. Αυτό επιτυγχάνεται αφού το μήνυμα θα σταλεί ξανά και ξανά μέχρι ο server να στείλει πίσω την κατάλληλη απάντηση (acknowledge message ACK).

- **Non-confirmable message (NON):** Αυτού του είδους τα μηνύματα δεν χρειάζονται απάντηση (ACK) από το server. Δεν περιέχουν κάποια κρίσιμη πληροφορία η οποία πρέπει να μεταφερθεί οπωσδήποτε στον server. Κυρίως περιέχουν τιμές από αισθητήρες ή άλλα δεδομένα.

- **Acknowledge message (ACK):** Η απάντηση του server για την επιβεβαίωση λήψης κάποιου εισερχόμενου αιτήματος.

- **Reset message (RST):** Αν ο server δεν μπορεί να εξυπηρετήσει το εισερχόμενο αίτημα, ως απάντηση θα σταλεί ένα Reset message (RST) αντί για ένα ACK message.

Το request/response layer είναι υπεύθυνο για την μέθοδο εξυπηρέτησης αιτημάτων και διαχείριση των request/response μηνυμάτων. Οι τρείς μέθοδοι που ακολουθούνται είναι:

- **Piggy-backed:** Ο client στέλνει κάποιο μήνυμα στο server (CON ή NON) και λαμβάνει απάντηση ACK. Εάν η απάντηση είναι επιτυχής τότε το ACK θα περιέχει το μήνυμα της απάντησης αλλιώς θα περιέχει κάποιο κωδικό αποτυχίας.

- **Separate response:** Σε περίπτωση που ο server λάβει κάποιο CON μήνυμα το οποίο δεν μπορεί να εξυπηρετήσει τότε στέλνει πίσω ένα κενό ACK εάν ο client στείλει ξανά το αντίστοιχο CON. Όταν ο server είναι έτοιμος να απαντήσει στέλνει ένα CON πίσω στον client ο οποίος απαντάει με την σειρά του μέσω ενός ACK.

- **Non confirmable request and response:** Σε αντίθεση με την περίπτωση του Piggy-backed, εάν ο client στείλει κάποιο NON μήνυμα, τότε ο server δεν είναι υποχρεωμένος να απαντήσει με κάποιο ACK, αλλά στέλνει και αυτός ένα NON μήνυμα με την απάντηση.

Πλεονεκτήματα

Το πρωτόκολλο CoAP είναι ικανό να συνεχίσει να δουλεύει ακόμη και σε δίκτυα τα οποία είναι πολύ περιορισμένα σε συνδεσιμότητα λόγω αυξημένης συμφόρησης, στα οποία αποτυγχάνουν πρωτόκολλα όπως το MQTT τα οποία είναι βασισμένα στο TCP. Είναι ένα αποδοτικό και συνάμα συντηρητικό πρωτόκολλο το οποίο επιτρέπει στις συσκευές να λειτουργούν ακόμη και με πολύ χαμηλό σήμα. Ακόμη παρέχει ασφάλεια μέσω του Datagram Transport Layer Security (DTLS), και η χρήση του UDP το καθιστά κατάλληλο για IoT συστήματα. Ο σχεδιασμός του προσφέρει αξιοπιστία όσον αφορά θέματα χαμηλού εύρους ζώνης, υψηλής συμφόρησης δικτύου και χαμηλής ενεργειακής κατανάλωσης καθώς και υποστήριξη δικτύων με δισεκατομμύρια κόμβους.

Μειονεκτήματα

Κάποια μειονεκτήματα που αντιμετωπίζει το πρωτόκολλο CoAP είναι η αύξηση στον απαιτούμενο χρόνο επεξεργασίας λόγο των ACK που στέλνονται για κάθε μήνυμα που λαμβάνεται καθώς και η αύξηση του φόρτου υλοποίησης του πρωτοκόλλου λόγω της χρήσης του Datagram Transport Layer Security (DTLS). Ακόμη το πρωτόκολλο CoAP αντιμετωπίζει προβλήματα στην επικοινωνία λόγω του Network Address Translation **(**NAT**).** Η χρήση των ACK εξασφαλίζει αξιοπιστία όσον αφορά την μετάδοση του μηνύματος αλλά δεν παρέχει διασφάλιση όσον αφορά την σωστή αποκρυπτογράφηση του μηνύματος.

3.1.3 DDS (Data Distribution Service)

Το πρωτόκολλο DDS (Data Distribution Service) είναι ένα πρωτόκολλο ή αλλιώς ένα middleware το οποίο είναι σχεδιασμένο για κλιμακώσιμη, M2M επικοινωνία σε πραγματικό χρόνο. Δημιουργήθηκε από την Real-Time Innovations και την Thales Group, μια γαλλική εταιρεία άμυνας. Το 2004 η Object Management Group (OMG) παρουσίασε το DDS version 1.0. Είναι κατάλληλο για κατανεμημένη επεξεργασία συνδέοντας απευθείας συσκευές και εφαρμογές χωρίς να απαιτείται ένας ενδιάμεσος μεσολαβητής (broker), όπως γίνεται με τα πρωτόκολλα MQTT και CoAP. Χρησιμοποιεί την μέθοδο ανταλλαγής δεδομένων publish-subscribe καθώς και την μέθοδο του multicast. Ακόμη κάνει χρήση πρωτοκόλλου UDP με την δυνατότητα να υποστηρίξει και TCP πρωτόκολλο.

Λειτουργία

Όπως προαναφέρθηκε το πρωτόκολλο DDS είναι ένα αποκεντρωμένο πρωτόκολλο, δηλαδή οι πηγές παραγωγής δεδομένων επικοινωνούν κατευθείαν με τις εφαρμογές (peer to peer). Τα δεδομένα περνούν από κάποιο κεντρικό σημείο πχ data center, μόνο όταν ζητηθούν εκεί. Χρησιμοποιεί ένα εικονικό καθολικό χώρο αποθήκευσης (global data space GDS) όπου κάθε κόμβος έχει αποθηκευμένα τοπικά τα δεδομένα που χρειάζεται, τα οποία ενημερώνονται από το πρωτόκολλο DDS με αποστολή μηνυμάτων στους κατάλληλους κόμβους με multicast για γρήγορη ενημέρωση. Έτσι δημιουργείται μία ψευδαίσθηση ύπαρξης ενός κεντρικού χώρου αποθήκευσης ο οποίος στην πραγματικότητα δεν υπάρχει, με τα δεδομένα να βρίσκονται αποθηκευμένα τοπικά σε κάθε κόμβο. Κάθε publisher ή subscriber μπορεί να συμμετάσχει ή να αποχωρήσει από το GDS οποιαδήποτε στιγμή καθώς. Ακόμη υπάρχει δυνατότητα εγγραφής και ανάγνωσης δεδομένων στο GDS ασύγχρονα και αυτόνομα. Μέσω του global data space, τα δεδομένα μοιράζονται σε cloud, mobile και edge εφαρμογών ανεξαρτήτως συστήματος και γλώσσας, τα οποία φιλτράρονται για την αποστολή μόνο της απαραίτητης πληροφορίας που χρειάζεται κάποιο end-point μειώνοντας έτσι το μέγεθος τους. Γενικά το πρωτόκολλο αυτό είναι υπεύθυνο για το ποιος θα λάβει το μήνυμα, πού βρίσκεται ο παραλήπτης και τι θα γίνει σε περίπτωση όπου το μήνυμα δεν παραληφθεί χωρίς να χρειάζεται εξωτερική παρεμβολή από κάποια εφαρμογή. Η μέθοδος ανταλλαγής μηνυμάτων και δεδομένων που χρησιμοποιείται είναι η publish – subscribe όπου κάθε κόμβος μπορεί να λειτουργήσει σαν publisher, subscriber ή και τα 2 ταυτόχρονα. Κάθε κόμβος που παράγει δεδομένα, δηλαδή ένας publisher δημιουργεί ένα topic στο οποίο κοινοποιεί τα δεδομένα που θέλει να στείλει (samples), τα οποία διανέμονται από το πρωτόκολλο DDS στους subscriber, οι οποίοι έχουν δηλώσει ενδιαφέρον για το συγκεκριμένο topic.

Η αρχιτεκτονική του πρωτοκόλλου DDS αποτελείται από 2 επίπεδα:

- **DCPS (Data Centric Publish Subscribe) layer:** Είναι το επίπεδο το οποίο είναι υπεύθυνο για την διανομή των δεδομένων στους subscriber. Αποτελεί ένα real time publish/subscribe API βασισμένο σε topics για την ανταλλαγή μηνυμάτων και δεδομένων.

- **DLRL (Data Local Reconstruction Layer):** Το επίπεδο αυτό παρέχει μια διεπαφή (interface) για τις λειτουργίες του DCPS, επιτρέποντας ουσιαστικά τον κατανεμημένο διαμοιρασμό των δεδομένων ανάμεσα στις IoT συσκευές [22].

Πλεονεκτήματα

Το πρωτόκολλο επικοινωνίας DDS είναι ένα ευέλικτο πρωτόκολλο το οποίο δεν παρουσιάζει εξάρτηση από κάποιο συγκεκριμένο λειτουργικό σύστημα, υλικό ή γλώσσα προγραμματισμού. Η έλλειψη του ενδιάμεσου μεσολαβητή (broker) καθιστά την υλοποίηση του πολύ απλή, παρέχει δυνατότητα δυναμικής κλιμάκωσης λόγω της δυναμικής ανακάλυψης των publisher/subscriber και συνεισφέρει στην ελαχιστοποίηση του latency. Ακόμη υπάρχει αυξημένη αξιοπιστία όσον αφορά την συνεχή λειτουργία του, καθώς και εξασφάλιση υψηλής απόδοσης με χρόνους απόκρισης που φτάνουν μέχρι και 30 μsec. Το γεγονός αυτό καθιστά το πρωτόκολλο DDS κατάλληλο για χρήση σε συστήματα με κρίσιμες λειτουργίες. Παρέχει ανωνυμία όσον αφορά τις πληροφορίες σύνδεσης και τοπολογίας καθώς και ασφάλεια μέσω των δυνατοτήτων διαπίστευσης και κρυπτογράφησης. Υποστηρίζει την μέθοδο του multicast καθώς και φιλτράρισμα των δεδομένων, με στόχο την εξασφάλιση αποδοτικής χρήσης του δικτύου και υπολογιστικών πόρων.

Μειονεκτήματα

Ένα μειονέκτημα που παρουσιάζει το πρωτόκολλο DDS αφορά την επιβάρυνση του δικτύου. Λόγω των διαφορετικών τύπων δεδομένων που μπορούν να σταλούν σε ένα πακέτο, υπάρχει ανάγκη για επιπρόσθετα δεδομένα ταυτοποίησης. Αυτό σε συνδυασμό με την δυνατότητα δυναμικής ανακάλυψης που παρέχει το πρωτόκολλο DDS μέσω των heartbeat πακέτων, ενδέχεται να υπάρξει μερική αύξηση του φόρτου στο δίκτυο σε σχέση με άλλα πρωτόκολλα. Ακόμη η παροχή μεγάλου αριθμού επιλογών που προσφέρει το πρωτόκολλο DDS, δεν το καθιστά την ευκολότερη επιλογή χρήσης για κάποιο σύστημα.

3.1.4 AMQP (Advanced Message Queuing Protocol)

Το πρωτόκολλο AMQP (Advanced Message Queuing Protocol) σχεδιάστηκε από τον John O'Hara στην τράπεζα JPMorgan Chase του Λονδίνου το 2003. Είναι ένα γρήγορο, ελαφρύ και δυαδικό M2M πρωτόκολλο, το οποίο υποστηρίζει request/response και publish/subscribe αρχιτεκτονικές. Σχεδιάστηκε για την επίτευξη διαλειτουργικότητας μεταξύ διαφόρων συστημάτων και εφαρμογών, ανεξάρτητα από τον εσωτερικό σχεδιασμό τους, χρησιμοποιώντας ένα δυαδικό σύστημα ανταλλαγής μηνυμάτων. Χρησιμοποιεί TCP πρωτόκολλο καθώς και μεθόδους διαπίστευσης και κρυπτογράφησης όπως οι SASL και TLS. Είναι κατάλληλο για περιβάλλοντα με πολλαπλούς client, βασίζοντας τη λειτουργία του στην ασύγχρονη ουρά μηνυμάτων όπως επίσης και στην χρήση των producers, brokers και consumers (subscribers). Οι επικεφαλίδες των μηνυμάτων λαμβάνουν μέγεθος από 8byte μέχρι κάποια μέγιστη τιμή η οποία καθορίζεται από τον broker/server.

Λειτουργία

Η λειτουργία του πρωτοκόλλου AMQP (Advanced Message Queuing Protocol) διαφέρει ανάλογα με την έκδοση που χρησιμοποιείται. Υπάρχουν 2 διαφορετικές εκδόσεις:

- **AMQP 0.9.1:** Η έκδοση AMQP 0.9.1 ακολουθεί την αρχιτεκτονική publish/subscribe η οποία βασίζεται σε 2 οντότητες του broker, τις exchange και ουρές μηνυμάτων (message queue). Με τον όρο exchange ορίζεται η οντότητα η οποία είναι υπεύθυνη για την λήψη των μηνυμάτων από τον publisher και την οδήγηση τους στις κατάλληλες ουρές μηνυμάτων ανάλογα με το πόσοι client έκαναν αίτηση για τα δεδομένα αυτά. Οι ουρές μηνυμάτων αποθηκεύουν τα εισερχόμενα μηνύματα μέχρι την κάποιος subscriber να τα ζητήσει. Ένας consumer δημιουργεί μια ουρά η οποία συνδέεται (bind) με μια οντότητα exchange. Η διαδικασία σύνδεσης ενός exchange και μιας ουράς μηνυμάτων ονομάζεται binding και περιέχει τους κανόνες και συνθήκες για την κατανομή των μηνυμάτων. Οι publishers και subscribers ανακαλύπτουν ο ένας τον άλλον μέσω του ονόματος της οντότητας exchange το οποίο γνωστοποιείται δημόσια μετά την δημιουργία της [23].

- **AMQP 1.0:** Η νεότερη έκδοση AMQP 1.0 σε αντίθεση με την έκδοση AMQP 0.9.1 ακολουθεί την αρχιτεκτονική request/response. Δεν ακολουθείται κάποιος συγκεκριμένος μηχανισμός αλλά χρησιμοποιείται μια peer to peer μέθοδος ανταλλαγής μηνυμάτων η οποία μπορεί να χρησιμοποιηθεί χωρίς κάποιον ενδιάμεσο μεσολαβητή (broker) ο οποίος είναι απαραίτητος μόνο στην περίπτωση όπου απαιτείται αποθήκευση των δεδομένων πριν την προώθηση τους σε κάποιον client.

Η αξιοπιστία είναι ένα από τα κύρια χαρακτηριστικά του πρωτοκόλλου AMQP και για τον λόγο αυτό προσφέρει 3 επίπεδα ποιότητας εξυπηρέτησης (Quality of Service) όσον αφορά την παράδοση μηνυμάτων:

- **At-most-once:** Το μήνυμα αποστέλνεται μόνο μια φορά, με την πιθανότητα να χαθεί κατά την αποστολή του. Ο αποστολέας δεν λαμβάνει πληροφορίες σχετικά με την επιτυχή ή ανεπιτυχή παράδοση του μηνύματος. Σε περίπτωση αποτυχίας ο αποστολέας δεν στέλνει ξανά το μήνυμα.

- **At-least-once:** Υπάρχει εγγύηση ότι το μήνυμα θα παραδοθεί αφού ο αποστολέας θα συνεχίσει να στέλνει το μήνυμα μέχρι να πάρει την κατάλληλη επιβεβαίωση επιτυχούς παραλαβής του μηνύματος. Εάν ο παραλήπτης λάβει το μήνυμα αλλά η επιβεβαίωση προς τον αποστολέα χαθεί κατά την αποστολή της, τότε ο αποστολέας θα προωθήσει ξανά το μήνυμα δημιουργώντας έτσι διπλά αντίγραφα.

- **Exactly-once:** Εγγυάται παράδοση του μηνύματος μόνο μια φορά λόγω της αμοιβαίας ανταλλαγής μηνυμάτων επιβεβαίωσης και από τις δύο πλευρές.

Πλεονεκτήματα

Το πρωτόκολλο AMQP προσφέρει μεγάλη ασφάλεια λόγω της χρήσης πρωτόκολλου κρυπτογράφησης TLS. Ακόμη υποστηρίζει διάφορες μεθόδους σειριοποίησης όπως Protocol Buffers, MessagePack, Thrift, and JSON για την αποστολή δομημένων δεδομένων παρέχοντας παράλληλα αξιοπιστία στην παράδοση των μηνυμάτων. Η ιδιότητα του να παρέχει μεγάλος εύρος υπηρεσιών όπως αξιόπιστες ουρές μηνυμάτων, publish/subscribe αρχιτεκτονική βασισμένη σε topics, ευέλικτη δρομολόγηση και συναλλαγές, το καθιστά προτιμώμενη επιλογή στα IoT συστήματα. Ένα σημαντικό χαρακτηριστικό του πρωτοκόλλου AMQP είναι η διαλειτουργικότητα, η οποία προσφέρει δυνατότητα ανταλλαγής μηνυμάτων σε ετερογενών συστημάτων και συσκευών ανεξαρτήτως λειτουργικού συστήματος και γλωσσών προγραμματισμού. Ακόμη η δυνατότητα υποστήριξης διαφόρων τοπολογιών όπως client-to-client, client-to-broker και broker-to-broker καθώς και αρχιτεκτονικών επικοινωνίας όπως request/response και publish/subscribe προσφέρει μεγάλη ευελιξία όσον αφορά το εύρος εφαρμογών του πρωτοκόλλου AMQP.

Μειονεκτήματα

Παρά το γεγονός ότι το AMQP είναι ένα ελαφρύ πρωτόκολλο, τα χαρακτηριστικά και οι λειτουργίες που υποστηρίζει όσον αφορά την ασφάλεια δεδομένων, αξιοπιστία και διαλειτουργικότητα αυξάνει το μέγεθος των μηνυμάτων. Ακόμη η εκτέλεση των διαφόρων διαδικασιών για την εξασφάλιση όλων των παραπάνω λειτουργιών και χαρακτηριστικών οδηγεί σε αυξημένες απαιτήσεις ενεργειακής κατανάλωσης, εύρους ζώνης και υπολογιστικών πόρων. Το πρωτόκολλο αυτό είναι κατάλληλο για μέρη του συστήματος όπου δεν υπάρχουν αυστηροί περιορισμοί σε εύρος ζώνης, latency και υπολογιστική ισχύ.

3.1.5 XMPP (Extensible Messaging and Presence Protocol)

Το πρωτόκολλο XMPP (Extensible Messaging and Presence Protocol) είναι ένα ανοικτό πρωτόκολλο επικοινωνίας το οποίο δημιουργήθηκε από την κοινότητα ανοικτού κώδικα Jabber το 1999, όπου αργότερα (2002) επισημοποιήθηκε από την IETF (Internet Engineering Task Force). Βασίζεται στην τεχνολογία XML (Extensible Markup Language) και επιτρέπει την άμεση ανταλλαγή δεδομένων και μηνυμάτων, σχεδόν σε πραγματικό χρόνο μεταξύ των οντοτήτων ενός δικτύου. Χρησιμοποιεί την τυπική αρχιτεκτονική client – server, πρωτόκολλο TCP, έχοντας επίσης την δυνατότητα να υποστηρίξει και την αρχιτεκτονική publish – subscribe. Είναι σχεδιασμένο να υποστηρίζει τις τεχνολογίες του VoIP, video, μεταφορές αρχείων και διάφορες IoT εφαρμογές, επιτρέποντας την επικοινωνία μεταξύ ετερογενών συσκευών ανεξαρτήτως λειτουργικού συστήματος.

Λειτουργία

Η λειτουργία του πρωτοκόλλου XMPP βασίζεται στην αποκεντρωμένη αρχιτεκτονική client-server η οποία μπορεί να επεκταθεί και σε publish-subscribe μοντέλο καθώς και σε 2 μεθόδους επικοινωνίας, την Standard bi-directional socket connection και την Bidirectional streams over Synchronous HTTP (BOSH). Το BOSH επιτρέπει σύνδεση με XMPP δίκτυα σε clients οι οποίοι μπορούν να συνδεθούν στο διαδίκτυο μόνο μέσω HTTP πρωτοκόλλου. Στην περίπτωση όπου χρησιμοποιείται το μοντέλο client-server όπου ένας client ενώνεται με έναν server για να επικοινωνήσει με τους άλλους client, δίνεται επίσης η δυνατότητα σύνδεσης ενός server με άλλους servers σε άλλα domains, δημιουργώντας έτσι ένα «Federation» δηλαδή μια ομοσπονδία από servers ενισχύοντας έτσι την κλιμακωσιμότητα των IoT συστημάτων τα οποία μπορούν να χειριστούν μεγαλύτερο αριθμό αιτημάτων. Το μοντέλο Publish-Subscribe του πρωτοκόλλου XMPP συνθέτει ένα σύστημα το οποίο αποτελείται από μια δενδροειδή δομή, την οποία μπορούν να προσπελάσουν οι χρήστες αποθηκεύοντας δεδομένα στους κόμβους τα οποία κοινοποιούνται ταυτόχρονα. Οι servers και οι clients επικοινωνούν μεταξύ τους μέσω XML streams ανταλλάζοντας δεδομένα σε μορφή XML stanza. Ο όρος stanza ή αλλιώς semantic structured data unit αναφέρεται σε κάποια δομημένη μορφή δεδομένων η οποία αποτελείται από τα εξής 3 μέρη:

- **Message:** Περιέχει τίτλο, περιεχόμενο, διευθύνσεις αποστολέα και προορισμού, τύπο δεδομένων και τα IDs. Ένα message stanza δεν λαμβάνει μήνυμα ενημέρωσης για επιτυχή αποστολή.

- **Presence:** Ειδοποιεί τους client για τυχόν ενημερώσεις όσον αφορά το status έχοντας τον ρόλο του subscription. Εάν κάποιος client κάνει subscribe σε κάποιο presence, κάθε φορά που κάποιος κόμβος ενημερώνει το presence του ο αντίστοιχος client ειδοποιείται.

- **iq:** Είναι υπεύθυνο για την αντιστοίχιση (map) μεταξύ των publishers και subscribers. Λαμβάνει πληροφορίες από τον server ή τους συνδεδεμένους client και διαχειρίζεται τις

ρυθμίσεις του server, λειτουργώντας με παρόμοιο τρόπο όπως οι μέθοδοι GET και POST του HTTP πρωτοκόλλου.

Πλεονεκτήματα

Το πρωτόκολλο XMPP προσφέρει πολλά πλεονεκτήματα στον χώρο του IoT όσον αφορά θέματα κλιμακωσιμότητας, ασφάλειας, ευελιξίας και διαλειτουργικότητας. Με την χρήση καθολικών μοναδικών διευθύνσεων και του DNS (Domain Name System) για την δρομολόγηση και παράδοση των μηνυμάτων στο δίκτυο, επιτυγχάνεται μεγάλη ευελιξία και κλιμακωσιμότητα. Λόγω των μοναδικών XMPP διευθύνσεων, το πρωτόκολλο παρέχει ικανότητα αναγνώρισης απεριόριστου αριθμού συσκευών. Ακόμη δίνει δυνατότητα επικοινωνίας στις συσκευές μεταξύ domains χωρίς την απαίτηση απαραίτητου λογισμικού από τον server. Διαθέτει μεγάλο αριθμό υλοποιήσεων για server, client και βιβλιοθήκες τα οποία είναι διαθέσιμα σε μορφή ανοικτού κώδικα ενισχύοντας την διαλειτουργικότητα. Το πρωτόκολλο γεφυρώνει την επικοινωνία μεταξύ παλιών και μοντέρνων συστημάτων μέσω των XMPP Gateway, οι οποίες είναι υπηρεσίες μέσω των οποίων επιτυγχάνεται σύνδεση του XMPP πρωτόκολλου με άλλα πρωτόκολλα. Σε θέματα ασφάλειας το πρωτόκολλο παρέχει αξιόπιστες υπηρεσίες για την διαπίστευση, εξουσιοδότηση και κρυπτογράφηση χρησιμοποιώντας το Transport Layer Security (TLS) και Simple Authentication and Security Layer (SASL). Έτσι εξασφαλίζεται ακεραιότητα κατά την μετάδοση των δεδομένων όπως επίσης προστασία ενάντια σε επιθέσεις και μη εξουσιοδοτημένη χρήση [23].

Μειονεκτήματα

Τα μειονεκτήματα που παρουσιάζει το πρωτόκολλο XMPP προκύπτουν από την χρήση του μορφότυπου XML για την μετάδοση των δεδομένων, ο οποίος θεωρείται ανεπαρκής σε σύγκριση με νεότερες τεχνολογίες όπως ο μορφότυπος JavaScript Object Notation (JSON) τον οποίο υποστηρίζουν πολλά σύγχρονα συστήματα. Η χρήση του μορφότυπου XML βοηθά στην διαχείριση του προβλήματος της ετερογένειας, αλλά επιφέρει επιπρόσθετο φόρτο στην μετάδοση δεδομένων λόγω της χρήσης πολλών επικεφαλίδων και tags, ο οποίος οδηγεί σε αύξηση της ενεργειακής κατανάλωσης των συσκευών. Ακόμη το μέγεθος των μηνυμάτων το οποίο προκύπτει από την χρήση του μορφότυπου XML, δυσκολεύει την χρήση του πρωτοκόλλου σε δίκτυα τα οποία χαρακτηρίζονται από περιορισμούς σε εύρος ζώνης (bandwidth).

3.2 Τεχνικές αποθήκευσης δεδομένων μεγάλης κλίμακας

Τα σημερινά IoT συστήματα καλούνται να διαχειριστούν σε πραγματικό χρόνο τεράστιους όγκους δεδομένων τα οποία παράγονται συνεχώς. Ένα από τα σημαντικότερα κομμάτια μιας αρχιτεκτονικής επεξεργασίας μεγάλου όγκου δεδομένων είναι η γρήγορη και αποδοτική αποθήκευση καθώς και ανάκτηση των δεδομένων αυτών. Η επιλογή της κατάλληλης τεχνικής αποθήκευσης αποτελεί κρίσιμο παράγοντα στην απόδοση του συστήματος αφού καλείται να διαχειριστεί τα προβλήματα ετερογένειας, όγκου και θορύβου που χαρακτηρίζουν τα δεδομένα που εισάγονται στο σύστημα, όπως επίσης και την εξασφάλιση αξιοπιστίας όσον αφορά θέματα αποθήκευσης και διάθεσης των δεδομένων. Στην παρακάτω ενότητα θα μελετηθούν τεχνικές αποθήκευσης δεδομένων.

3.2.1 HDFS (Hadoop Distributed File System)

Το HDFS (Hadoop Distributed File System) είναι ένα κλιμακώσιμο, κατανεμημένο σύστημα αρχείων και αποθήκευσης δεδομένων υλοποιημένο στην γλώσσα προγραμματισμού Java, το οποίο επιτρέπει την κατανεμημένη αποθήκευση δεδομένων σε ένα δίκτυο από υπολογιστικές μηχανές. Αποτελεί μέρος της πλατφόρμας επεξεργασίας δεδομένων Hadoop και ακολουθεί την αρχιτεκτονική master/worker χρησιμοποιώντας nodes δηλαδή λογισμικά τα οποίοι τρέχουν πάνω σε GNU/Linux λειτουργικό σύστημα και χωρίζονται σε 2 κατηγορίες:

- **Namenode (Master):** Κάθε HDFS cluster, δηλαδή ένα σύνολο υπολογιστικών μηχανών περιέχει έναν Namenode. Ο Namenode είναι υπεύθυνος για την διαχείριση του namespace του συστήματος εκτελώντας διεργασίες όπως άνοιγμα, κλείσιμο, μετονομασία αρχείων και φακέλων καθώς και ανάθεση άλλων εργασιών στους διάφορους Namenodes. Ακόμη διαθέτει μεταπληροφορίες (metadata) όπως τον αριθμό και μέγεθος των blocks, τοποθεσίες της αποθηκευμένης πληροφορίας στους διάφορους Datanodes όπως επίσης και των αντίγραφων (replicas) τους. Είναι υπεύθυνος για την αντιστοίχιση (mapping) των Datanodes με τα blocks καθώς και για την εξασφάλιση πρόσβασης του χρήστη στα αρχεία του συστήματος [24].

- **Datanode (Worker):** Ένα HDFS cluster περιέχει πολλούς Datanodes, συνήθως ένας για κάθε node-μηχάνημα στο cluster. Είναι υπεύθυνοι για την διαχείριση του αποθηκευτικού χώρου ο οποίος βρίσκεται στον node όπου τρέχουν. Οι Datanodes διαμοιράζονται και αποθηκεύουν τα δεδομένα μεταξύ τους, τα οποία σπάνε σε ένα ή περισσότερα blocks των 128MB συνήθως. Αποτελούν τους εργάτες του συστήματος, οι οποίοι εκτελούν τις διεργασίες διαβάσματος και γραψίματος κατ' απαίτηση των χρηστών. Ακόμη μετά από εντολή του Namenode μπορούν να δημιουργήσουν, να διαγράψουν ή να αντιγράψουν ένα συγκεκριμένο block. Κάθε Datanode στέλνει ένα heartbeat στον Namenode κάθε 3 δευτερόλεπτα για να αποδείξει ότι είναι σε λειτουργία. Εάν ένας Namenode δεν στείλει κάποιο heartbeat σε διάρκεια 2 λεπτών τότε ο Namenode τον ανακηρύσσει νεκρό και ξεκινά την διαδικασία αντιγραφής των block του σε κάποιον άλλο Namenode.

Λειτουργία

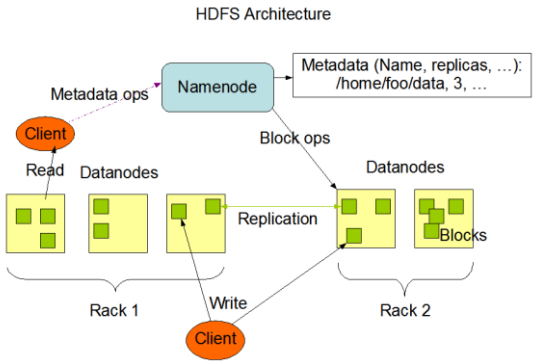
Το HDFS αποθηκεύει τα δεδομένα με ένα κατανεμημένο τρόπο διάφορα μηχανήματα-nodes, σπάζοντας τα σε μικρότερα μέρη, τα blocks, τα οποία έχουν default τιμή 128MB. Τα blocks αποτελούν την μικρότερη μονάδα δεδομένων στο σύστημα και το μέγεθος τους μπορεί να αυξηθεί ή να μειωθεί ανάλογα με τις ανάγκες του χρήστη. Εάν το μέγεθος των δεδομένων είναι μικρότερο από την τιμή μεγέθους ενός block, τότε υπάρχει προσαρμογή στο εκάστοτε block το οποίο αποκτά μέγεθος ίσο με το μέγεθος των δεδομένων. Για παράδειγμα εάν έχουμε ένα αρχείο με μέγεθος 180MB τότε θα δημιουργηθούν 2 block με μεγέθη 128MB και 52MB αντίστοιχα για την αποθήκευση του. Το 1^{ο} block θα γεμίσει πλήρως και τα υπολειπόμενα 76MB από το 2^{ο} block θα ανακατανεμηθούν αποδοτικά από το σύστημα για να μην σπαταληθεί ο χώρος αυτός. Το γεγονός ότι το HDFS σπάει τα αρχεία τα οποία ενδεχομένως να είναι τεραστίου μεγέθους σε μικρότερα κομμάτια, μειώνει τον χρόνο αναζήτησης, ανάκτησης και επεξεργασίας αφού οι διεργασίες αυτές μπορούν να γίνουν παράλληλα στα διάφορα blocks που αποτελούν το αρχείο αυτό. Ένα από τα κύρια χαρακτηριστικά του HDFS είναι η διαδικασία δημιουργίας πολλαπλών αντιγράφων ενός block και η αποθήκευση τους σε διαφορετικούς Datanodes γνωστή και ως Replication. Ο αριθμός αντιγράφων που θα δημιουργηθούν καθορίζεται από το replication factor το οποίο έχει συνήθως τιμή 3. Αυτό σημαίνει ότι κατά την δημιουργία ενός block θα δημιουργηθούν μαζί άλλα 2 αντίγραφα, έτσι ώστε να υπάρχουν συνολικά 3 blocks. Η πολιτική που ακολουθεί το HDFS όσον αφορά την τοποθέτηση των αντιγράφων στους διάφορους Datanodes στην περίπτωση όπου το replication factor είναι ίσο με 3, είναι τέτοια ώστε να υπάρχει τοποθέτηση των 2 από τα 3 block σε 2 Datanodes του ιδίου rack και το 3^{ο} block σε έναν Datanode ενός άλλου rack. Ένα rack αποτελεί ένα σύνολο από πολλούς nodes, επομένως ένα cluster από υπολογιστές δηλαδή nodes είναι διεσπαρμένο σε διάφορα racks. Έτσι κάθε Datanode έχει στην κατοχή του πολλά διαφορετικά blocks αλλά ποτέ 2 αντίγραφα του ιδίου block. Αντιθέτως κάθε rack μπορεί να περιέχει πολλαπλά αντίγραφα ενός block σε διαφορετικούς όμως Datanodes. Η επικοινωνία μεταξύ 2 Datanode σε διαφορετικά racks επιτυγχάνεται μέσω switches. Οι Datanodes μπορούν να επικοινωνούν και να μετακινούν αντίγραφα μεταξύ τους για την επίτευξη εξισορρόπησης των δεδομένων ανάμεσα στα διάφορα racks. Η διαδικασία διαβάσματος στο HDFS γίνεται μέσω του Namenode. Όταν κάποιος χρήστης θέλει να διαβάσει από ένα αρχείο τότε πρέπει να επικοινωνήσει με τον Namenode ο οποίος έχει τις πληροφορίες τοποθεσίας των δεδομένων στους αντίστοιχους Datanodes με τους οποίους θα επικοινωνήσει έπειτα ο χρήστης για να διαβάσει τα δεδομένα. Εάν κατά την διάρκεια του διαβάσματος ενός block, ο συγκεκριμένος Datanode παρουσιάσει κάποιο πρόβλημα, ο Namenode θα τον μεταφέρει σε κάποιον άλλο κοντινό Datanode ο οποίος κατέχει αντίγραφο του επιθυμητού block. Παρόμοια, κατά την διαδικασία γραψίματος σε αρχείο από κάποιο χρήστη πρέπει να υπάρξει επικοινωνία με τον Namenode ο οποίος θα παρέχει την διεύθυνση τοποθεσίας των επιθυμητών blocks. Αφού τελειώσει το γράψιμο σε κάποιο block, αρχίζει αυτόματα η διαδικασία δημιουργίας πολλαπλών αντιγράφων του τροποποιημένου αυτού block (με τον αριθμό να ορίζεται από το replication factor), το οποίο προωθείται από Datanode σε Datanode. Η επιλογή του κατάλληλου Datanode για την εξυπηρέτηση ενός αιτήματος ανάγνωσης ή εγγραφής γίνεται με την χρήση του αλγόριθμου Rack Awareness από τον Namenode οποίος επιλέγει τον κοντινότερο Datanode με βάση το id του κάθε rack.

Πλεονεκτήματα

Το HDFS προσφέρει πολλά πλεονεκτήματα όσον αφορά την αποθήκευση και επεξεργασία μεγάλου όγκου δεδομένων. Αρχικά η δημιουργία πολλαπλών αντιγράφων των δεδομένων στα διάφορα blocks εξασφαλίζει μεγάλη διαθεσιμότητα των δεδομένων, αφού ακόμη και σε περίπτωση όπου κάποιο block αντιμετωπίσει κάποιο πρόβλημα, τα δεδομένα του block αυτού μπορούν να ανακτηθούν από κάποιον άλλο Datanode ο οποίος έχει αντίγραφο του συγκεκριμένου block. Ακόμη το HDFS λόγο κατασκευής, μπορεί να λειτουργήσει πάνω σε υλικό το οποίο είναι φθηνό σε κόστος και ευκολά προσβάσιμο στην αγορά, χωρίς να υπάρχει φόβος απώλειας δεδομένων σε περίπτωση βλάβης κάποιου μηχανήματος, αφού διαθέτει πολλαπλά αντίγραφα των δεδομένων σε διάφορα racks και Datanodes από τα οποία μπορεί να γίνει ανάκτηση. Ένα HDFS cluster είναι πολύ εύκολα κλιμακώσιμο τόσο κάθετα όσο και οριζόντια, αφού μπορεί να γίνει προσθήκη είτε αποθηκευτικών δίσκων στους Datanodes (Vertical Scaling), είτε περισσότεροι nodes στο ίδιο το cluster (Horizontal Scaling). Ένα άλλο πλεονέκτημα που προσφέρει το HDFS είναι το υψηλό throughput, αφού η εκτέλεση μιας εργασίας διαιρείται και μοιράζεται σε πολλά μικρά υποσυστήματα τα οποία εργάζονται παράλληλα και ανεξάρτητα συμβάλλοντας στην πολύ γρήγορη διεκπεραίωση της όπως επίσης και χαμηλή συμφόρηση στο δίκτυο. Επίσης το HDFS μπορεί να χρησιμοποιήσει αρχεία με τεράστιο όγκο δεδομένων (gigabytes και terabytes) καθώς και πληθώρα τύπων δεδομένων, δομημένων και μη, όπως αρχεία CSV, XML, text κτλ.

Μειονεκτήματα

Ένα από τα μειονεκτήματα που αντιμετωπίζει το HDFS αφορά το μέγεθος των αρχείων που μπορεί να διαχειριστεί. Το HDFS αντιμετωπίζει προβλήματα όταν έχει να κάνει με αρχεία μικρού μεγέθους αφού δεν έχει σχεδιαστεί για αυτά. Ο μεγάλος αριθμός αρχείων τα οποία έχουν μέγεθος μικρότερο από το μέγεθος ενός block, οδηγούν στην υπερφόρτωση του Namenode ο οποίος αποθηκεύει το Namespace για το σύστημα. Ακόμη το HDFS δεν λειτουργεί αποδοτικά για εφαρμογές οι οποίες απαιτούν πολύ μικρό latency αφού είναι σχεδιασμένο να παρέχει υψηλό throughput, δηλαδή το πλήθος εκτέλεσης εργασιών στην μονάδα χρόνου, με το ενδεχόμενο ύπαρξης υψηλού latency. Επίσης το HDFS είναι κατάλληλο μόνο για batch processing και όχι για stream processing σε πραγματικό χρόνο, αφού δουλεύει μόνο για δεδομένα τα οποία έχουν αποθηκευτεί σε αρχεία πριν από την επεξεργασία τους.



Σχήμα 1.9 Αρχιτεκτονική HDFS

Πηγή: *http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html*

3.2.2 Amazon S3 (Amazon Simple Storage Service)

Η τεχνολογία Amazon S3 (Amazon Simple Storage Service), η οποία αποτελεί μέρος των Amazon Web Services, έκανε την εμφάνιση της το 2006. Αποτελεί μια διαδικτυακή υπηρεσία της εταιρίας Amazon, η οποία προσφέρει υπηρεσίες αποθήκευσης δεδομένων σε μορφή αντικειμένων οποιουδήποτε τύπου, μέσω μιας διαδικτυακής διεπαφής. Χρησιμοποιείται από πολλές μεγάλες εταιρείες όπως Netflix, Tumblr, Pinterest, Reddit κλπ.

Λειτουργία
Η αρχιτεκτονική της Amazon S3 αποτελείται από τα εξής κύρια χαρακτηριστικά:

- **Objects:** Ένα object αποτελεί την βασική μορφή αποθήκευσης των δεδομένων. Αντιπροσωπεύει ένα αντικείμενο το οποίο αποτελείται από τα κύρια δεδομένα καθώς και μεταδεδομένα. Τα μεταδεδομένα καθορίζουν πληροφορίες σχετικά με αντικείμενο όπως ημερομηνία τελευταίας μετατροπής, τύπο δεδομένων, μέγεθος κτλ. Παράλληλα δίνεται η δυνατότητα δημιουργίας και άλλων μεταδεδομένων εκτός από τα προκαθορισμένα ανάλογα με τις ανάγκες του χρήστη. Το μέγεθος των αντικειμένων μπορεί να φτάσει μέχρι και 5TB με 2KB μεταδεδομένα.

- **Keys:** Κάθε object το οποίο αποθηκεύεται σε κάποιο bucket περιέχει κάποιο μοναδικό αναγνωριστικό δηλαδή ένα κλειδί (key). Το κλειδί αυτό είναι μια αλληλουχία χαρακτήρων με μέγιστο μέγεθος UTF-8 κωδικοποίησης 1024 bytes. Κάθε αντικείμενο αναγνωρίζεται μέσω ενός συνδυασμού του κλειδιού, του bucket και ενός προαιρετικού version id. Η αρχιτεκτονική της Amazon S3 δεν παρέχει ιεραρχική δομή φακέλων και ευρετηρίων. Η δημιουργία μιας τέτοιας ιεραρχικής δομής μπορεί να επιτευχθεί μέσω της ονομασίας των κλειδιών, χρησιμοποιώντας το διαχωριστικό ‘/’ κατά την ονομασία ενός κλειδιού. Για παράδειγμα το κλειδί με όνομα «Reports/report.pdf» οδηγεί στην δημιουργία μίας ιεραρχικής δομής φακέλου με το όνομα «Reports» και περιεχόμενο το αντικείμενο «report.pdf».

- **Buckets:** Ένα bucket αποτελεί ένα αποθηκευτικό μέσο, μέσα στο οποίο αποθηκεύονται τα διάφορα objects όπως επίσης και ένα μέσο οργάνωσης του Amazon S3 namespace. Το όνομα κάθε ενός από τα bucket είναι μοναδικό και δεν μπορεί να χρησιμοποιηθεί από κάποιον άλλο. Κατά την δημιουργία ενός bucket καθορίζεται η περιοχή (Region) στην οποία θα ανήκει, έτσι κρίνεται απαραίτητη η επιλογή μιας περιοχής η οποία είναι κοντά στον χρήστη, ούτως ώστε να επιτευχθεί μείωση του κόστους και latency. Ένα object O, το οποίο δημιουργείται στην περιοχή R, στο bucket B, μπορεί να προσπελαστεί μέσω της διεύθυνσης https://B.s3.R.O.

- **Regions:** Για κάθε bucket το οποίο δημιουργείται, δίνεται η δυνατότητα επιλογής γεωγραφικής περιοχής (region) στην οποία και θα αποθηκευτεί. Τα δεδομένα τα οποία αποθηκεύονται σε κάποια συγκεκριμένη περιοχή παραμένουν σε αυτή συνεχώς. Αλλαγή περιοχής γίνεται μόνο μετά από απαίτηση του χρήστη.

Η Amazon S3 ακολουθεί ένα read-after-write μοντέλο συνοχής δεδομένων. Κάθε GET, PUT, DELETE request μπορεί να μην φέρει το αναμενόμενο αποτέλεσμα ανάλογα με την χρήση του. Για παράδειγμα, ένα GET request αμέσως μετά από ένα PUT σε κάποιο κλειδί μπορεί να επιστρέψει τα παλιά δεδομένα για το συγκεκριμένο κλειδί. Ακόμη ένα GET request για κάποιο αντικείμενο αμέσως μετά από κάποιο DELETE request στο συγκεκριμένο αντικείμενο μπορεί να επιστρέψει τα διαγραμμένα δεδομένα. Αυτό συμβαίνει λόγω του ότι η πληροφόρηση για κάθε αλλαγή που συμβαίνει στα δεδομένα πρέπει να αντιγραφεί και να μεταδοθεί σε πολλούς servers οι οποίοι ανήκουν σε ένα AWS data center, πράγμα που απαιτεί κάποιο χρονικό διάστημα. Μέσω της δημιουργίας πολλαπλών αντιγράφων των δεδομένων αυτών, στους διάφορους servers η Amazon S3 προσφέρει υψηλή διαθεσιμότητα των δεδομένων. Παρόμοιο μοντέλο συνοχής δεδομένων ακολουθούν και τα buckets. Ακόμη δίνεται η δυνατότητα αποθήκευσης πολλαπλών εκδόσεων κάποιου αντικειμένου σε ένα bucket. Η Amazon S3 προσφέρει 4 διαφορετικές κλάσεις αποθήκευσης, την Amazon S3 Standard η οποία είναι η προκαθορισμένη επιλογή, την Amazon S3 Standard Infrequent Access (IA) η οποία είναι κατάλληλη για δεδομένα τα οποία δεν θα προσπελάυνονται συχνά πχ για backup και recovery, την Amazon S3 One Zone-Infrequent Access η οποία είναι κατάλληλη για δεδομένα τα οποία δεν είναι απαραίτητα συχνά, αλλά στην περίπτωση που χρειάζονται πρέπει η προσπέλαση τους να γίνεται αστραπιαία και τέλος την κλάση Amazon Glacier η οποία είναι σχεδιασμένη για την μακροχρόνια αποθήκευση δεδομένων, όπου στην περίπτωση που χρειαστεί προσπέλαση τους, θα υπάρχει μεγάλη ανοχή στο χρόνο ανάκτησης τους ο οποίος μπορεί να κυμαίνεται μεταξύ λεπτών και ωρών. Επίσης υπάρχει υποστήριξη REST και SOAP διεπαφών [25].

Πλεονεκτήματα

Μερικά από τα πλεονεκτήματα που προσφέρει η Amazon S3 είναι η μεγάλη κλιμακωσιμότητα του συστήματος όπως επίσης και η υψηλή διαθεσιμότητα των δεδομένων (99.999999999%), η οποία επιτυγχάνεται μέσω τη δημιουργίας πολλαπλών αντιγράφων σε πολλούς servers έτσι ώστε τα δεδομένα να είναι προστατευμένα και διαθέσιμα συνεχώς ανεξαρτήτως σφαλμάτων. Ακόμη προσφέρει ασφάλεια και προστασία από μη εξουσιοδοτημένη πρόσβαση στα δεδομένα δίνοντας την δυνατότητα στον χρήστη να επιλέξει ποιος θα έχει πρόσβαση σε αυτά και αν τα δεδομένα θα είναι διαθέσιμα δημοσίως. Επίσης δίνεται η δυνατότητα χρησιμοποίησης διαφόρων χρήσιμων εργαλείων ανάλυσης δεδομένων πάνω στα αποθηκευμένα αντικείμενα. Ένα άλλο πλεονέκτημα που προσφέρει είναι η εύκολη ενσωμάτωση με άλλες υπηρεσίες της Amazon όπως EC2 και CloundFront.

Μειονεκτήματα

Ένα από τα κύρια μειονεκτήματα της υπηρεσίας Amazon S3, είναι η πολυπλοκότητα της διαδικασίας στησίματος και χρήσης, η οποία μπορεί να κριθεί ακατάλληλη για αρχάριους χρήστες. Ακόμη η λάθος διαχείριση της υπηρεσίας όσον αφορά τους διαθέσιμους πόρους μπορεί να οδηγήσει σε μεγάλο χρηματικό κόστος, λόγω του ότι η υπηρεσία δεν προσφέρεται εντελώς δωρεάν.

3.2.3 Βάσεις Δεδομένων

Οι πολύπλευρες και ποικίλες εφαρμογές των σύγχρονων IoT συστημάτων, έχουν ως αποτέλεσμα την παραγωγή δεδομένων μεγάλης κλίμακας από διάφορες ετερογενείς πηγές. Για τον λόγο αυτό, τα δεδομένα τα οποία παράγονται από ένα σύγχρονο IoT σύστημα, χαρακτηρίζονται από ετερογένεια, ποικιλία και έλλειψη αυστηρού μοντέλου δομής, αυξάνοντας την ανάγκη εύρεσης ευέλικτων και κλιμακώσιμων λύσεων, για την αποδοτική διαχείριση των δεδομένων μεγάλης κλίμακας, τα οποία χαρακτηρίζονται από τις ιδιότητες αυτές. Την ανάγκη αυτή δεν μπορούν να ικανοποιήσουν αποδοτικά οι μέχρι τώρα υπάρχουσες SQL βάσεις δεδομένων όπως είναι οι MySQL, Microsoft SQL Server κτλ οι οποίες αποτελούν σχεσιακές βάσεις, σχεδιασμένες για την διαχείριση δεδομένων τα οποία ακολουθούν ένα αυστηρό μοντέλο δομής. Λύση στην αντιμετώπιση των προβλημάτων, των οποίων δημιουργούνται κατά την διαχείριση δεδομένων μεγάλης κλίμακας, έρχεται να δώσει μια άλλη σύγχρονη προσέγγιση στην τεχνολογία των βάσεων δεδομένων, αυτή των NoSQL βάσεων. Οι NoSQL βάσεις δεδομένων, γνωστές και ως μη σχεσιακές βάσεις δεδομένων αποτελούν μια σύγχρονη λύση, προσφέροντας υψηλή απόδοση, κλιμακωσιμότητα και ευελιξία όσον αφορά την αποθήκευση και διαχείριση δεδομένων μεγάλης κλίμακας. Προσφέρουν δυνατότητα αποθήκευσης σε δεδομένα τα οποία δεν ακολουθούν κάποιο αυστηρό μοντέλο δομής. Σε αντίθεση με τις SQL βάσεις δεδομένων, οι NoSQL τεχνολογίες χρησιμοποιούν ένα δυναμικό μοντέλο δομής για την αποθήκευση των δεδομένων το οποίο μπορεί να είναι της μορφής εγγράφου, στηλών, γράφων ή ακόμη και KeyValue δομής. Στην παρακάτω ενότητα θα μελετηθούν NoSQL βάσεις δεδομένων, διαφόρων μοντέλων δομής δεδομένων, συγκρίνοντας τα πλεονεκτήματα που χαρακτηρίζουν την κάθε λύση, όσον αφορά την διαχείριση και αποθήκευση δεδομένων μεγάλης κλίμακας.

Document βάσεις δεδομένων

- **MongoDB:** Η βάση δεδομένων MongoDB αποτελεί μια NoSQL βάση δεδομένων η οποία αποθηκεύει δεδομένα σε συλλογές εγγράφων. Μία συλλογή αποτελείται από έγγραφα και αντιστοιχεί στην μορφή πίνακα την οποία χρησιμοποιούν οι SQL βάσεις δεδομένων. Κάθε έγγραφο αποτελείται από ζεύγη κλειδιών-τιμών της μορφής BSON (Binary JSON), των οποίων οι σχέσεις ορίζονται μέσω του εμφωλιασμού τους σε ένα έγγραφο. Η χρήση του μορφότυπου BSON καθιστά το μοντέλο δομής των δεδομένων δυναμικό, αφού κάθε έγγραφο αποτελείται από εμφωλιασμένα αντικείμενα της μορφής BSON, τα οποία δεν αποτελούνται απαραίτητα από τα ίδια πεδία και ιδιότητες. Η βάση δεδομένων MongoDB χρησιμοποιεί ευρετήρια παρόμοιας μορφής με την μορφή B-Tree διατηρώντας τις αντιστοιχίσεις των αρχείων στην κύρια μνήμη, στην οποία αρχικά γίνονται οι εγγραφές δεδομένων πριν γίνει η αντιγραφή τους στον δίσκο [26]. Μερικά από τα πλεονεκτήματα που προσφέρει η βάση δεδομένων MongoDB, είναι η μεγάλη ευελιξία και απλότητα που χαρακτηρίζει το δυναμικό μοντέλο αποθήκευσης δεδομένων λόγω του μορφότυπου BSON, όπως επίσης και η μεγάλη δυνατότητα κλιμακωσιμότητας που προσφέρει. Ακόμη προσφέρει μεγάλη ταχύτητα στην αποθήκευση και προσπέλαση των δεδομένων λόγω της χρήσης κύριας μνήμης.

Column Family βάσεις δεδομένων

- **Cassandra:** Η τεχνολογία Cassandra αποτελεί ένα κλιμακώσιμο κατανεμημένο σύστημα διαχείρισης NoSQL βάσεων δεδομένων το οποίο δημιουργήθηκε από τους Avinash Lakshman και Prashant Malik της εταιρίας Facebook και αποτελείται από λογισμικό ανοικτού κώδικα. Αποθηκεύει τα δεδομένα σε μορφή στηλών τα οποία προσπελάυνονται μέσω ενός κλειδιού γραμμής. Ο μηχανισμός αποθήκευσης δεδομένων της Cassandra, χρησιμοποιεί ένα δομημένο σε logs, Merge Tree όπου οι εγγραφές γίνονται πρώτα στην κύρια μνήμη και ακολούθως στον δίσκο προσφέροντας μεγάλη ταχύτητα στην αποθήκευση και προσπέλαση μεγάλης κλίμακας δεδομένων. Ακόμη η τεχνολογία Cassandra εξασφαλίζει μεγάλη ανεκτικότητα στα σφάλματα λόγω της αυτόματης δημιουργίας πολλαπλών αντιγράφων των δεδομένων τα οποία διανέμονται σε πολλαπλούς κόμβους ενός cluster και μπορούν να διατεθούν άμεσα σε περίπτωση σφάλματος του κόμβου λειτουργίας, από κάποιον άλλο γειτονικό κόμβο. Αποτελεί μια εξαιρετικά οριζόντια κλιμακώσιμη τεχνολογία μέσω της προσθήκης επιπλέων μηχανημάτων στο cluster που τρέχει, γεγονός το οποίο προσφέρει υψηλή απόδοση στην λειτουργία.

- **HBase:** Η βάση δεδομένων HBase αποτελεί μια κατανεμημένη βάση δεδομένων μεγάλης κλίμακας υλοποιημένη σε Java από τον οργανισμό Apache Software Foundation ως ένα κομμάτι το Apache Hadoop, το οποίο τρέχει πάνω στο HDFS. Το μοντέλο λειτουργίας της βάσης δεδομένων HBase είναι παρόμοιο με αυτό της βάσης δεδομένων Google Bigtable η οποία αποτελεί κομμάτι του GFS (Google File System) . Η βάση δεδομένων HBase μπορεί να υποστηρίξει τεράστιους πίνακες με δισεκατομμύρια γραμμές και εκατομμύρια στήλες στους οποίους υπάρχει ένα πεδίο ορισμένο ως primary key χωρίς όμως να αποτελεί μια SQL βάση δεδομένων, διατηρώντας ένα ευέλικτο μοντέλο δομής δεδομένων. Συγκεκριμένα κάθε στήλη ενός πίνακα αποτελεί μια συλλογή από key-value ζεύγη και κάθε συλλογή από στήλες αποτελεί μια column family. Μια γραμμή ενός πίνακα αποτελεί μια συλλογή από column families και ένας πίνακας ορίζεται ως μία συλλογή από γραμμές. Η χρήση της βάσης δεδομένων HBase προσφέρει δυνατότητες επεξεργασίας δεδομένων σε πραγματικό χρόνο όπως επίσης και προσπελάσεις εγγραφής/ανάγνωσης σε μεγάλο όγκο δεδομένων (petabytes) με πολύ χαμηλό latency χρησιμοποιώντας πίνακες κατακερματισμού. Ακόμη αποτελεί μια γραμμικά κλιμακώσιμη λύση η οποία είναι σχεδιασμένη να επεκτείνεται μεταξύ χιλιάδων server. Παρέχει μεγάλη ανεκτικότητα σε σφάλματα μέσω της δημιουργίας πολλαπλών αντιγράφων των πινάκων τα οποία διανέμονται μέσα στο cluster και μπορούν να διατεθούν άμεσα από κάποιον άλλο κόμβο σε περίπτωση σφάλματος του κόμβου λειτουργίας. Η αρχιτεκτονική λειτουργίας της βάσης δεδομένων HBase, προσφέρει μεγάλη συνέπεια όσον αφορά την εγγραφή κι ανάγνωση των δεδομένων σε σύγκριση με άλλες τεχνολογίες όπως η Cassandra η οποία είναι eventual consistent.

Graph βάσεις δεδομένων

- **Neo4j:** Η βάση δεδομένων Neo4j αποτελεί μια βάση δεδομένων υλοποιημένη σε Java, της οποίας το μοντέλο δεδομένων βασίζεται στην δομή των γράφων. Μια βάση δεδομένων η οποία λειτουργεί με δομές γράφων αποτελεί αναπαράσταση μια ενός συνόλου αντικειμένων των οποίων οι σχέσεις εκφράζονται μέσω της σύνδεσης τους με ακμές. Τα δεδομένα αποθηκεύονται σε μορφή ακμής, κόμβου ή κάποιου attribute το οποίο αντιστοιχεί σε ένα key-value ζεύγος, όπου οι μορφές αυτές αντιμετωπίζονται ως οντότητες πρώτης τάξης. Κάθε ακμή ή κόμβος μπορεί να περιέχει πολλά attributes όπως επίσης και κάποιο label το οποίο βοηθά στις αναζητήσεις. Η χρήση της βάσης δεδομένων Neo4j προσφέρει ένα ευέλικτο μοντέλο δεδομένων το οποίο μπορεί να τροποποιηθεί ανάλογα με τις ανάγκες της εφαρμογής υποστηρίζοντας όλες τις ACID ιδιότητες. Ακόμη η χρήση της Neo4j προσφέρει δυνατότητες κλιμακωσιμότητας μέσω της multi-clustering λειτουργίας η οποία επιτρέπει αύξηση των εγγραφών και αναγνώσεων υποστηρίζοντας μέχρι και 10 δισεκατομμύρια κόμβους και ακμές, χωρίς να επηρεάζεται η ταχύτητα λειτουργίας και η ακεραιότητα των δεδομένων. Μέσω της δημιουργίας αντιγράφων των δεδομένων εξασφαλίζεται υψηλή ασφάλεια και διαθεσιμότητα των δεδομένων. Η βάση δεδομένων Neo4j υποστηρίζει την Cypher Query Language, μια γλώσσα ερωτημάτων η οποία επιτρέπει την διαχείριση των σχέσεων μεταξύ των δεδομένων χωρίς την απαίτηση χρήσης σύνθετων join.

Key-Value βάσεις δεδομένων

- **Redis:** Η βάση δεδομένων Redis αποτελεί μια in-memory λύση η οποία δημιουργήθηκε από τον Salvatore Sanfilippo και χρησιμοποιείται ως βάση δεδομένων, cache και message broker. Η λειτουργία της υποστηρίζει διάφορες δομές δεδομένων όπως hashes, strings, sets, λίστες, bitmaps, hyperloglogs, streams πάνω στις οποίες δίνεται δυνατότητα εφαρμογής atomic λειτουργιών όπως συνένωση string, αύξηση μιας τιμής κτλ. Όλα τα δεδομένα βρίσκονται αποθηκευμένα στην κύρια μνήμη του server σε σύγκριση με άλλες βάσεις δεδομένων οι οποίες αποθηκεύουν τα δεδομένα σε στον δίσκο ή σε SSDs. Η βάση δεδομένων Redis υποστηρίζει ασύγχρονη δημιουργία πολλαπλών αντιγράφων των δεδομένων μέσω των replication trees, εξασφαλίζοντας συνέπεια, αξιοπιστία και διαθεσιμότητα των δεδομένων. Χαρακτηρίζεται από υψηλή απόδοση λόγω της in-memory φύσης του, σε αντίθεση με άλλα συστήματα βάσεων δεδομένων τα οποία καταγράφουν την κάθε αλλαγή στον δίσκο πριν να θεωρηθεί έγκυρη, προϋποθέτοντας έτσι συνεχείς προσπελάσεις στον δίσκο για την εκτέλεση διάφορων λειτουργιών. Ακόμη προσφέρει μεγάλες δυνατότητες κλιμακωσιμότητας ανάλογα με τις ανάγκες της εφαρμογής, μέσω της αρχιτεκτονικής της, η οποία υποστηρίζει clustered τοπολογίες.

3.2.4 Elasticsearch

Η τεχνολογία Elasticsearch αποτελεί μια κλιμακώσιμη κατανεμημένη μηχανή αναζήτησης και ανάλυσης text-based δεδομένων η οποία προσφέρει μια διαδικτυακή HTTP διεπαφή και χρησιμοποιεί schema-free JSON documents. Δημιουργήθηκε από τον Shay Banon τον Φεβρουάριο του 2010. Αποτελεί λογισμικό ανοικτού κώδικα βασισμένο στην βιβλιοθήκη Lucene, το οποίο χρησιμοποιείται για την αποθήκευση, αναζήτηση και ανάλυση μεγάλου όγκου δεδομένων σχεδόν σε πραγματικό χρόνο. Σχεδιάστηκε για εφαρμογές οι οποίες έχουν πολύπλοκες απαιτήσεις όσον αφορά την αναζήτηση δεδομένων. Χρησιμοποιείται από πολλές μεγάλες εταιρείες όπως Cisco, Netflix, Tinder κτλ.

Λειτουργία
Η αρχιτεκτονική λειτουργίας της τεχνολογίας Elasticsearch αποτελείται από τη εξής κύρια μέρη:

- **Nodes:** Servers (φυσικοί ή εικονικοί) οι οποίοι αποτελούν μέρος ενός cluster και είναι υπεύθυνοι για την αποθήκευση και αναζήτηση των δεδομένων. Κάθε Node περιλαμβάνει κάποιο Universally Unique Identifier (UUID) το οποίο του ανατίθεται κατά την εκκίνηση του και αποτελεί το αναγνωριστικό του. Υπάρχουν 3 είδη Node, master ο οποίος κατανέμει εργασίες στο cluster, data οι οποίοι αποθηκεύουν τα δεδομένα σε μορφή shard και εκτελούν εργασίες όπως αναζήτηση, indexing, aggregating και ο client ο οποίος λειτουργεί σαν load balancer, δρομολογώντας τα αιτήματα αναζήτησης των χρηστών. Κάθε Node διαχειρίζεται τα HTTP αιτήματα τα οποία στέλνονται από τον client μέσω HTTP Rest API.

- **Cluster:** Ομάδα από έναν ή περισσότερους Nodes οι οποίοι συνεργάζονται για την εκτέλεση των κατανεμημένων εργασιών όπως η αποθήκευση, αναζήτηση, αντιγραφή κτλ. των δεδομένων.

- **Index:** Αποτελεί μια συλλογή από δεδομένα (documents) τα οποία έχουν παρόμοια χαρακτηριστικά. Κάθε index διαθέτει κάποιο μοναδικό αναγνωριστικό το οποίο χρησιμοποιείται κατά την αναζήτηση, διαγραφή ή ενημέρωση των δεδομένων, παρόμοια με τα σχεσιακά μοντέλα βάσεων.

- **Document:** Αποτελεί την βασική μονάδα πληροφορίας του συστήματος. Η δομή ενός document αποτελείται από ένα JSON object.

- **Shard:** Η χρήση της τεχνολογίας Elasticsearch επιτρέπει την διάσπαση ενός index σε πολλά Shards. Κάθε shard αποτελεί ένα ανεξάρτητο και λειτουργικό index το οποίο βρίσκεται σε κάποιον Node του cluster. Η χρήση των Shards επιτρέπει την οριζόντια διάσπαση των δεδομένων με σκοπό τον παραλληλισμό των εργασιών, ο οποίος θα ενισχύσει την απόδοση του συστήματος. Ακόμη μέσω των Shards δίνεται η δυνατότητα

δημιουργίας πολλαπλών αντιγράφων προσφέροντας έτσι υψηλή διαθεσιμότητα των δεδομένων [27].

Η τεχνολογία Elasticsearch μπορεί να συνδυαστεί με άλλες τεχνολογίες όπως:

- **Beats:** Ελαφρύς μεταφορέας δεδομένων απευθείας στο Elasticsearch
- **APM server:** Μέτρηση απόδοσης εφαρμογών
- **Elasticsearch Hadoop:** Γρήγορος μεταφορέας δεδομένων από και προς το Hadoop
- **Kibana:** Πίνακας ελέγχου για εξερεύνηση και οπτικοποίηση δεδομένων
- **Logstash:** Μηχανή συλλογής δεδομένων με δυνατότητες real-time pipelining

υλοποιώντας έτσι το λεγόμενο Elastic Stack του οποίου η χρήση αυξάνει σε μεγάλο βαθμό τις δυνατότητες και λειτουργικότητα του όλου συστήματος.

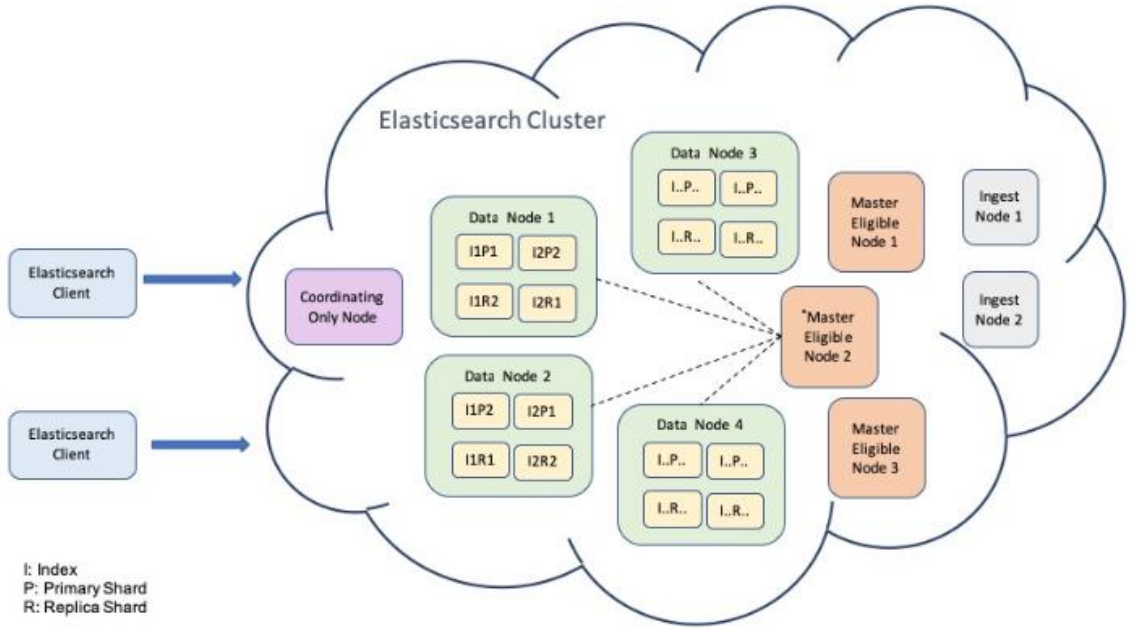
Πλεονεκτήματα

Μερικά από τα πλεονεκτήματα που προσφέρει η χρήση της τεχνολογίας Elasticsearch είναι η μεγάλη απόδοση λόγω της κατανεμημένης λειτουργίας η οποία επιτρέπει παραλληλισμό κατά την επεξεργασία και αναζήτηση μεγάλων όγκων δεδομένων. Έτσι αποτελεί μια σύγχρονη λύση η οποία είναι κατάλληλη για εφαρμογές οι οποίες απαιτούν αναζήτηση και επεξεργασία σε σχεδόν πραγματικό χρόνο. Ακόμη προσφέρει υποστήριξη σε πολλές γλώσσες όπως Java, Python, JavaScript, PHP, Ruby και τεχνολογίες όπως Nodejs κτλ. Εκτός από τις τεχνολογίες με τις οποίες υλοποιείται το Elastic Stack, προσφέρεται υποστήριξη σε διάφορα Elasticsearch plugins όπως αναλυτές γλώσσας κτλ. Ένα άλλο σημαντικό πλεονέκτημα το οποίο προσφέρει η χρήση της τεχνολογίας Elasticsearch είναι η υψηλή διαθεσιμότητα των δεδομένων. Μέσω της δημιουργίας πολλαπλών αντιγράφων (replicas) τα οποία κατανέμονται στους Nodes του cluster, εξασφαλίζεται διαθεσιμότητα των δεδομένων ακόμη και μετά από κάποια βλάβη ενός Node αφού γίνεται εκλογή νέου master σε περίπτωση σφάλματος κάποιου άλλου. Γενικά η τεχνολογία Elasticsearch αποτελεί μια πολύ απλή και ευέλικτη λύση, αφού προσφέρει ένα απλό REST API και μια απλή HTTP διεπαφή όπου σε συνδυασμό με την χρήση των JSON documents μπορούν να υλοποιηθούν εύκολα εφαρμογές για διάφορα σενάρια χρήσης.

Μειονεκτήματα

Παρά τον μεγάλο αριθμό πλεονεκτημάτων που προσφέρει η χρήση της τεχνολογίας Elasticsearch, υπάρχουν κάποια μειονεκτήματα όσον αφορά την λειτουργία της. Αρχικά υπάρχει περιορισμός όσον αφορά τον μορφότυπο αφού υπάρχει υποστήριξη μόνο για JSON μορφότυπο δεδομένων και όχι για CSV, XML κτλ. Ακόμη η τεχνολογία Elasticsearch μπορεί να αποτελέσει μια απαιτητική σε κόστος λύση, αφού για την επίτευξη βέλτιστης απόδοσης απαιτούνται servers εξοπλισμένοι με τουλάχιστον 64GB RAM και δίσκους SSD. Επίσης η βέλτιστη λειτουργία των ερωτημάτων απαιτεί σωστή οργάνωση των ιεραρχιών των indexes, IDs κτλ. Σε σενάρια χρήσης τα οποία απαιτούν streaming πολλών TB από

δεδομένα κάθε μέρα, οι τεχνικές αποθήκευσης των Hadoop και MongoDB πλεονεκτούν σε σχέση με την τεχνολογία Elasticsearch.



Σχήμα 1.10 Αρχιτεκτονική Elasticsearch

Πηγή: *https://subscription.packtpub.com/book/data/9781789957754/1/ch01lvl1sec04/elasticsearch-architectural-overview*

3.3 Τεχνικές streaming και επεξεργασίας δεδομένων σε πραγματικό χρόνο

Ένα άλλο σημαντικό κομμάτι ενός ολοκληρωμένου IoT συστήματος είναι η παροχή υπηρεσιών streaming, και επεξεργασίας δεδομένων σε πραγματικό χρόνο. Διάφορες εφαρμογές IoT συστημάτων απαιτούν άμεση επεξεργασία και διάθεση των δεδομένων με σκοπό την λήψη κρίσιμων αποφάσεων και εκτέλεση λειτουργιών, όπου μια πιθανή καθυστέρηση μπορεί να καταστήσει τα δεδομένα άχρηστα ή ανακριβή οδηγώντας σε αποτυχία λειτουργίας του συστήματος. Για παράδειγμα ένα σύστημα με έξυπνα οχήματα το οποίο επεξεργάζεται δεδομένα όπως συντεταγμένες άλλων οχημάτων, περιβάλλον κτλ. απαιτεί συνεχή επεξεργασία των εισερχομένων δεδομένων σε πραγματικό χρόνο και άμεση διάθεση τους στα εμπλεκόμενα συστήματα με σκοπό την λήψη κρίσιμων αποφάσεων για την διατήρηση της σωστής λειτουργίας του συστήματος. Για τον λόγο αυτό η επεξεργασία και διάθεση ροών δεδομένων σε πραγματικό χρόνο αποτελεί σημαντική λειτουργία για ένα σύγχρονο IoT σύστημα. Στην παρακάτω ενότητα θα μελετηθούν διάφορες τεχνικές real-time streaming και επεξεργασίας δεδομένων σε πραγματικό χρόνο.

3.3.1 Apache Kafka

Η τεχνολογία Apache Kafka είναι μια κατανεμημένη πλατφόρμα επεξεργασίας ροών δεδομένων υλοποιημένη σε Scala και Java. Αποτελεί λογισμικό ανοικτού κώδικα το οποίο αρχικά αναπτύχθηκε από τη εταιρεία LinkedIn και έπειτα δόθηκε στον οργανισμό Apache Software Foundation. Ακολουθεί την προσέγγιση του Native Streaming, δηλαδή τα εισερχόμενα δεδομένα επεξεργάζονται απευθείας μόλις φθάσουν στο σύστημα χωρίς καμία χρονοκαθυστέρηση. Είναι σχεδιασμένη για την διαχείριση και επεξεργασία ροών δεδομένων, δημιουργώντας αγωγούς δεδομένων οι οποίοι μεταφέρουν τα δεδομένα αυτά μεταξύ συστημάτων και εφαρμογών σε πραγματικό χρόνο, με αξιόπιστο τρόπο. Χρησιμοποιείται από πολλές μεγάλες εταιρείες όπως Twitter, AirBNB, Netflix, Uber κτλ.

Η τεχνολογία Apache Kafka αποτελείται από τα εξής 5 κύρια APIs:

- **Producer API:** Επιτρέπει την δημοσίευση ροών δεδομένων από μια εφαρμογή σε ένα ή περισσότερα Kafka topics.

- **Consumer API:** Επιτρέπει σε μια εφαρμογή να εγγραφεί σε κάποιο topic με σκοπό την επεξεργασία της εισερχόμενης ροής δεδομένων στο συγκεκριμένο topic.

- **Streams API:** Δίνει την δυνατότητα σε μια εφαρμογή να παράξει ροές δεδομένων ως έξοδο χρησιμοποιώντας ως είσοδο άλλες ροές δεδομένων από διάφορα topics, τις οποίες επεξεργάζεται.

- **Connector API:** Δίνει την δυνατότητα δημιουργίας και χρήσης επαναχρησιμοποιήσιμων producer και consumer με σκοπό την σύνδεση μεταξύ διάφορων εφαρμογών και Kafka topics.

- **Admin API:** Επιτρέπει την διαχείριση και εποπτεία διάφορων οντοτήτων της πλατφόρμας Kafka όπως brokers, topics κτλ.

Λειτουργία

Η αρχιτεκτονική της τεχνολογίας Apache Kafka ακολουθεί 2 μοντέλα διαχείρισης δεδομένων, το queuing και το publish-subscribe μοντέλο. Το μοντέλο queuing επιτρέπει την ανάγνωση ενός κομματιού δεδομένων από έναν μόνο consumer, ακριβώς μια φορά, δίνοντας μεγάλες δυνατότητες κλιμακωσιμότητας στο μοντέλο αυτό. Στην περίπτωση του μοντέλου publish-subscribe τα δεδομένα διατίθενται σε όλους τους subscriber μέσω της εγγραφής τους σε ανάλογα topics. Οι subscribers μπορούν να διαβάσουν τα δεδομένα μόνο μια φορά, έτσι δεν υπάρχει δυνατότητα κλιμακωσιμότητας. Η επικοινωνία client – server γίνεται μέσω TCP πρωτοκόλλου. Η τεχνολογία Apache Kafka είναι σχεδιασμένη να τρέχει σε cluster ενός ή περισσότερων server οι οποίοι ονομάζονται brokers. Αποθηκεύει τα δεδομένα τα οποία προέρχονται από τους producers, σε μορφή κλειδιού-τιμής τα οποία και χωρίζονται σε topics. Για κάθε topic η πλατφόρμα Apache Kafka κρατάει ένα log το οποίο αποτελεί μία ακολουθία εγγραφών, μοιρασμένη σε κομμάτια τα οποία αναλογούν σε διαφορετικούς subscribers, εξασφαλίζοντας την δυνατότητα ύπαρξης πολλαπλών subscriber σε κάθε topic. Κάθε topic χωρίζεται σε partitions στα οποία τα δεδομένα μπαίνουν σε σειρά ανάλογα με κάποιο offset και αποθηκεύονται μαζί με κάποια χρονοσφραγίδα. Κάθε partition αντιστοιχεί σε κάποιον subscriber ενισχύοντας τις δυνατότητες κλιμακωσιμότητας του συστήματος μέσω της παράλληλης κατανάλωσης των δεδομένων. Υπάρχει συνεχόμενη προσθήκη εγγραφών σε πραγματικό χρόνο στο log, κατά την διάρκεια προώθησης δεδομένων από τους producers. Η τεχνολογία Apache Kafka υποστηρίζει 2 ειδών topic, τα regular και compacted. Στην περίπτωση των regular topic, αν ο χρόνος ζωής κάποιας εγγραφής είναι μεγαλύτερος από κάποιο χρονικό όριο ή ο διαθέσιμος χώρος ενός partition έχει εξαντληθεί, υπάρχει δυνατότητα διαγραφής των παλιών εγγραφών για την απελευθέρωση χώρου. Το προκαθορισμένο χρονικό όριο ζωής συνήθως είναι 7 ημέρες με δυνατότητα αύξησης του. Στην περίπτωση των compacted topic τα νεότερα δεδομένα διαχειρίζονται σαν ενημερώσεις των παλιών δεδομένα με το ίδιο κλειδί εξασφαλίζοντας ύπαρξη της νεότερης έκδοσης ανά κλειδί. Τα διάφορα partitions ενός topic διανέμονται με κατανεμημένο τρόπο στους διάφορους κόμβους του cluster, οι οποίοι διαθέτουν επίσης πολλαπλά αντίγραφα από άλλα partitions. Κάθε partition έχει κάποιον server σαν αρχηγό και μηδέν ή περισσότερους servers ως ακόλουθους οι οποίοι αντιγράφουν τις ενέργειες του αρχηγού. Ο αρχηγός είναι υπεύθυνος για την εξυπηρέτηση των αιτημάτων ανάγνωσης-εγγραφής και σε περίπτωση όπου αντιμετωπίσει κάποιο πρόβλημα λειτουργίας, κάποιος από τους ακόλουθους τους θα γίνει ο νέος αρχηγός. Οι producers δημοσιεύουν δεδομένα σε topics της επιλογής τους και είναι υπεύθυνοι για την επιλογή των εγγραφών που θα ανατεθούν σε κάθε partition ενός topic. Οι consumers ομαδοποιούνται σε γκρουπ και κάθε εγγραφή η οποία δημοσιεύεται σε κάποιο topic δίνεται σε μόνο έναν consumer από κάθε γκρουπ καθιστώντας τον αποκλειστικό κάτοχο της δεδομένης εγγραφής για το

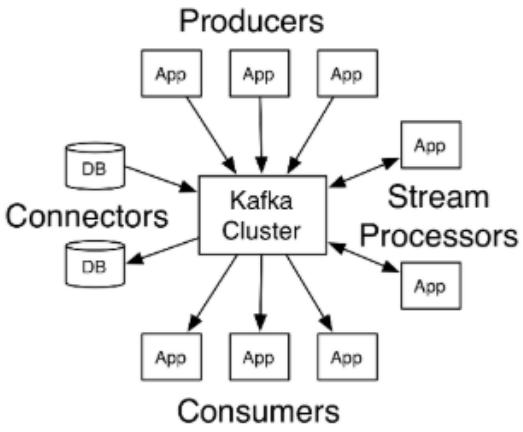
συγκεκριμένο γκρουπ. Αν όλοι οι consumer βρίσκονται σε διαφορετικά γκρουπ τότε η εγγραφή δίνεται σε όλους τους consumer. Κάθε νέος consumer σε κάποιο γκρουπ, αναλαμβάνει μερικά partitions και για κάθε consumer που πεθαίνει γίνεται διαμερισμός των partitions στους υπόλοιπους consumers [28]. Η κύρια λειτουργία της πλατφόρμας Apache Kafka είναι η επεξεργασίας των ροών δεδομένων σε πραγματικό χρόνο. Με την βοήθεια του Streams API δίνεται η δυνατότητα σε μία εφαρμογή να δέχεται ως είσοδο συνεχόμενες ροές δεδομένων μέσω των topics, να τις επεξεργάζεται και να δίνει ως έξοδο τις επεξεργασμένες αυτές ροές δεδομένων. Για παράδειγμα μια εφαρμογή μπορεί να δέχεται ως είσοδο ροές δεδομένων αποτελούμενες από διάφορες τιμές θερμοκρασίας ή υγρασίας και να δίνει σαν έξοδο διάφορους στατιστικούς δείκτες, οι οποίοι προκύπτουν από την επεξεργασία των εισερχομένων ροών δεδομένων σε πραγματικό χρόνο.

Πλεονεκτήματα

Μερικά από τα πλεονεκτήματα που προσφέρει η πλατφόρμα Apache Kafka αφορούν την δυνατότητα επίτευξης μεγάλου throughput, χιλιάδων μηνυμάτων ανά δευτερόλεπτο με πολύ χαμηλό latency. Ακόμη παρέχει μεγάλη ανοχή σε σφάλματα λόγω της δημιουργία πολλαπλών αντιγράφων των partitions μέσα σε ένα cluster όπως επίσης και δυνατότητες κλιμακωσιμότητας μέσω της προσθήκης επιπλέον κόμβων στο cluster. Παρά το γεγονός ότι προσφέρει μεγάλο throughput και χαμηλό latency, εγγυάται συνέπεια στην εγγραφή και ανάγνωση δεδομένων. Εκτός από την δυνατότητα διαχείρισης ροών δεδομένων σε πραγματικό χρόνο μπορεί να χρησιμοποιηθεί και για επεξεργασία δεδομένων σε παρτίδες (batch processing).

Μειονεκτήματα

Σε κάποιες περιπτώσεις μπορεί να χρειαστεί μετατροπή των μηνυμάτων για να μπορέσουν να μεταδοθούν. Αυτό οδηγεί σε μεγάλη μείωση στην απόδοση της πλατφόρμας Apache Kafka. Ακόμη μείωση στην απόδοση μπορεί να παρατηρηθεί λόγω της συμπίεσης και αποσυμπίεσης της ροής δεδομένων όπως επίσης και κατά την αύξηση των ουρών στο cluster. Επίσης η πλατφόρμα δεν υποστηρίζει μεθόδους ανταλλαγής μηνυμάτων όπως point-to-point queues ή request/reply οι οποίες μπορεί να είναι αναγκαίες σε μερικές περιπτώσεις.



Σχήμα 1.11 Αρχιτεκτονική Apache Kafka
Πηγή: *https://kafka.apache.org/intro.html*

3.3.2 Apache Storm

Η τεχνολογία Apache Storm αποτελεί ένα real-time κατανεμημένο υπολογιστικό σύστημα, κατάλληλο για την επεξεργασία μεγάλου όγκου ροών δεδομένων σε πραγματικό χρόνο, υλοποιημένο στις γλώσσες προγραμματισμού Clojure και Java. Δημιουργήθηκε από τον Nathan Marz και την ομάδα του στην εταιρεία BackType, όπου έπειτα καθιερώθηκε ως λογισμικό ανοικτού κώδικα από το Twitter. Η τεχνολογία Apache Storm λειτουργεί παρόμοια με την διαδικασία του batch processing της τεχνολογίας Hadoop, χρησιμοποιώντας όμως ροές δεδομένων σε πραγματικό χρόνο, σε μία «event by event» επεξεργασία ακολουθώντας την προσέγγιση του Native Streaming. Η τεχνολογία Apache Storm χρησιμοποιείται από διάφορες μεγάλες εταιρείες όπως Twitter, Yahoo!, Spotify κτλ.

Η τεχνολογία Apache Storm σχεδιάστηκε να λειτουργεί ως μια τοπολογία σε σχήμα ενός ακυκλικού κατευθυνόμενου γράφου (DAG). Η αρχιτεκτονική της αποτελείται από τα εξής κύρια επιμέρους στοιχεία:

- **Topology:** Αποτελεί ένα δίκτυο από spout και bolt το οποίο αναπαριστά την συνολική εικόνα της επεξεργασίας μέσα στο σύστημα. Κάθε κόμβος περιέχει υπολογιστική λογικά όπως επίσης και ακμές με άλλους κόμβους οι οποίες ορίζουν την κατεύθυνση μεταφοράς των δεδομένων και σειρά εκτέλεσης των διεργασιών.

- **Stream:** Είναι αγωγοί δεδομένων τα οποία είναι της μορφής tuple. Η μορφή αυτή αποτελεί κύριο συστατικό της αρχιτεκτονικής αυτής. Αποτελούν τις ακμές στον γράφο τοπολογίας της αρχιτεκτονικής.

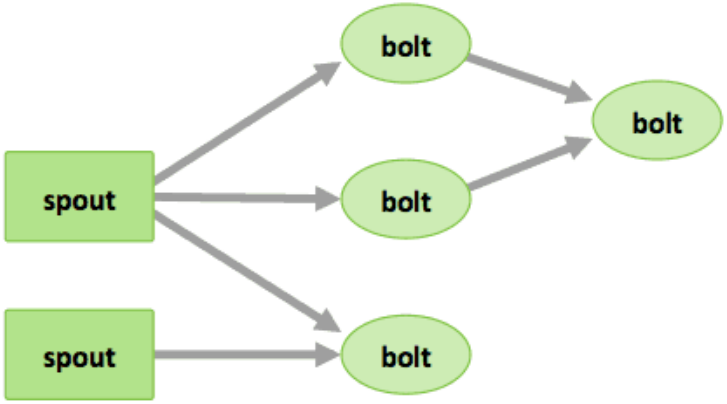
- **Spout:** Αποτελούν τις πηγές εισόδου των δεδομένων στο σύστημα. Είναι υπεύθυνα για την συνεχόμενη λήψη δεδομένων, μετατροπή τους σε streams από tuples και αποστολή τους στους κόμβους (bolts) για να επεξεργαστούν.

- **Bolt:** Αποτελούν τους υπολογιστικούς κόμβους της τοπολογίας. Λαμβάνουν τις εισερχόμενες ροές δεδομένων, τις οποίες είτε αποθηκεύουν σε βάση δεδομένων, είτε επεξεργάζονται εφαρμόζοντας διάφορες τεχνικές φιλτραρίσματος, συνένωσης ή αθροίσματος, έτσι ώστε να παράξουν νέες ροές δεδομένων ως έξοδο.

Λειτουργία

Η γενική λειτουργία ενός Storm cluster βασίζεται στην δημιουργία τοπολογιών επεξεργασίας οι οποίες επεξεργάζονται συνεχώς τις εισερχόμενες ροές δεδομένων μέχρι κάποιος να τις τερματίσει. Η αρχιτεκτονική αυτή περιλαμβάνει 3 είδη κόμβων, τον Master ή αλλιώς Nimbus, τους Supervisors και τους Workers. Ο Nimbus node είναι υπεύθυνος για την ανάθεση των διαφόρων εργασιών στους Workers μέσω των Supervisors όπως επίσης και για την επιτήρηση τους συστήματος για τυχόν σφάλματα. Οι Supervisors κόμβοι είναι υπεύθυνοι για την έναρξη και παύση λειτουργίας των Worker, με σκοπό την εκτέλεση των διεργασιών που τους ανατίθενται από τον Nimbus. Οι Workers είναι υπεύθυνοι για την εκτέλεση των διαφόρων εργασιών. Κάθε διεργασία Worker εκτελεί κάποιο υποσύνολο μίας

τοπολογίας, έτσι κάθε τοπολογία περιέχει πολλές διεργασίες Workers κατανεμημένες σε πολλά μηχανήματα [29]. Η επίτευξη επικοινωνίας μεταξύ Nimbus και Supervisor κόμβων γίνεται μέσω του ZooKeeper, μιας υπηρεσίας παροχής συντονισμού και διαχείρισης κατανεμημένων συστημάτων. Η τοπολογία υποβάλλεται στον Nimbus κόμβο ο οποίος με την βοήθεια ZooKeeper εντοπίζει τους διαθέσιμους Supervisors έτσι ώστε να αναθέσει εργασίες στους διάφορους Workers. Λόγω της χρήσης του ZooKeeper, υπάρχει δυνατότητα συνεχούς προσθήκης κόμβων (Worker ή Supervisor) οι οποίοι ενσωματώνονται αυτόματα στο cluster. Αν ένας Worker κόμβος πεθάνει τότε ο Supervisor του θα τον επανεκκινήσει και σε περίπτωση που η επανεκκίνηση αποτυγχάνει συνεχώς τότε θα ανατεθεί σε άλλο μηχάνημα. Ακόμη σε περίπτωση θανάτου κάποιου Worker κόμβου, το μερίδιο δουλειάς του θα ανατεθεί σε κάποιον άλλο Worker κόμβο. Σε περίπτωση που ο Nimbus κόμβος πεθάνει, οι Workers παραμένουν ανεπηρέαστοι ωστόσο δεν μπορεί να τους ανατεθεί νέα εργασία πριν την επανεκκίνηση του Nimbus.



Σχήμα 1.12 Τοπολογία αρχιτεκτονικής Apache Storm
Πηγή: *https://www.cloudera.com/products/open-source/apache-hadoop/apache-storm.html*

Πλεονεκτήματα

Μερικά από τα πλεονεκτήματα που προσφέρει η τεχνολογία Apache Storm είναι η δυνατότητα αυξημένης κλιμακωσιμότητας και η ικανότητα παράλληλης επεξεργασίας δεδομένων σε υψηλές ταχύτητες ακόμη και με μεγάλο όγκο δεδομένων (δυνατότητα επεξεργασία ενός εκατομμυρίου μηνυμάτων των 100byte κάθε δευτερόλεπτο σε κάθε κόμβο). Ακόμη προσφέρει αξιοπιστία στην επεξεργασία δεδομένων μέσω της προσέγγισης «fail fast, auto restart», η οποία επιτρέπει την επανεκκίνηση κόμβων μετά από κάποιο σφάλμα χωρίς να επηρεάζεται το σύστημα. Έτσι εξασφαλίζεται η επεξεργασία κάθε tuple τουλάχιστον ή ακριβώς μια φορά, ακόμη και μετά από σφάλματα. Επίσης χαρακτηρίζεται από ευκολία στην λειτουργία, αφού ένα Storm cluster είναι έτοιμο να λειτουργήσει σε περιβάλλον παραγωγής, αμέσως μετά την ρύθμιση του.

Μειονεκτήματα

Έναν περιορισμό που αντιμετωπίζει η τεχνολογία Apache Storm, είναι η ανικανότητα εκτέλεσης χρονοπρογραμματισμένων (scheduled) εργασιών, αφού παρέχει μόνο δυνατότητα επεξεργασίας ροών δεδομένων σε πραγματικό χρόνο.

3.3.3 Apache Flink

Η τεχνολογία Apache Flink αποτελεί μια μηχανή/ κατανεμημένης επεξεργασίας περιορισμένων και μη ροών δεδομένων, δηλαδή ροών οι οποίες έχουν αρχή αλλά όχι πάντα τέλος, σε πραγματικό χρόνο. Είναι σχεδιασμένο να τρέχει σε όλα τα κοινότυπα cluster περιβάλλοντα, εκτελώντας in-memory επεξεργασία στις εισερχόμενες ροές δεδομένων μέσω αγωγών με παράλληλο τρόπο, προσφέροντας δυνατότητες τόσο για stream processing, όσο και για batch processing. Το Apache Flink δημιουργήθηκε από τον οργανισμό Apache Software Foundation το 2009 και προσφέρεται ως λογισμικό ανοικτού κώδικα υλοποιημένο σε Java και Scala. Χρησιμοποιείται από πολλές μεγάλες εταιρείες όπως eBay, Huawei, Amazon, Uber κτλ.

Η αρχιτεκτονική της λειτουργία τους Apache Flink ακολουθεί το master/worker μοντέλο και αποτελείται από τις εξής 3 κύριες οντότητες:

- **Job Client:** Ο Job Client αποτελεί το αρχικό στάδιο εκτέλεσης της επεξεργασίας. Είναι υπεύθυνος για την λήψη του προγράμματος εκτέλεσης από τον client και προώθησης του στον Job Manager. Μετά την ολοκλήρωση εκτέλεσης μιας εργασίας, μεταφέρει τα αποτελέσματα πίσω στον χρήστη.

- **Job Manager:** Ο Job Manager είναι υπεύθυνος για την ενορχήστρωση της κατανομής των πόρων και εκτέλεση των εργασιών. Αφού κατανέμει τους απαιτούμενους πόρους για την εκτέλεση των εργασιών, μοιράζει τις εργασίες στους Task Managers για να εκτελεστούν. Ακόμη ο Job Manager διαχειρίζεται λειτουργίες όπως χρονοπρογραμματισμό εργασιών, διαχείριση των checkpoints και ανάνηψη από σφάλματα. Σε ένα σύστημα μπορούν να υπάρχουν πολλοί Job Managers αλλά ένας είναι ο αρχηγός. Σε περίπτωση που ο αρχηγός αντιμετωπίσει κάποιο πρόβλημα, εκλέγεται ένας standby κόμβος ως νέος αρχηγός.

- **Task Manager:** Οι Task Managers είναι υπεύθυνοι για την εκτέλεση των διάφορων εργασιών που τους ανατίθενται από τον Job Manager. Κατά την διάρκεια της εκτέλεσης των εργασιών οι Task Managers δίνουν συνεχή αναφορά στον Job Manager όσον αφορά τις αλλαγές μεταξύ των καταστάσεων εκτέλεσης. Κάθε Task Manager node καλείται να διαχειριστεί ένα η περισσότερα Task slot τα οποία αντιστοιχούν σε νήματα εκτέλεσης διεργασιών. Το μέγεθος παραλληλισμού της εκτέλεσης των εργασιών καθορίζεται από τον αριθμό των διαθέσιμων Task slots. Για παράδειγμα ένας Task Managers με 5 διαθέσιμα Task slots κατανέμει το 20% της μνήμης τους σε κάθε slot, για την παράλληλη εκτέλεση των threads.

Λειτουργία

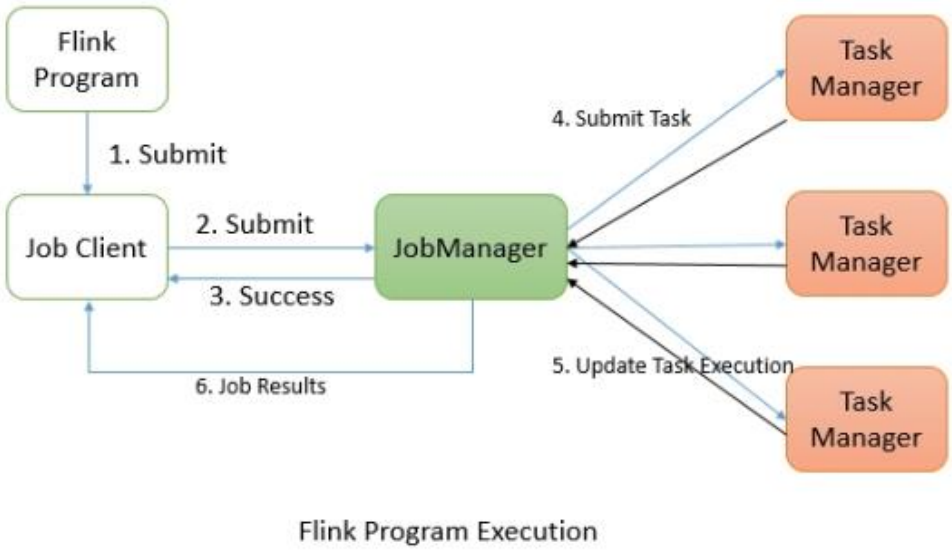
Το Apache Flink είναι σχεδιασμένο να τρέχει streaming εφαρμογές μεγάλης κλίμακας εφαρμόζοντας τεχνικές παράλληλης επεξεργασίας δεδομένων, μέσω της διαίρεσης της εφαρμογής σε πολλές εργασίες οι οποίες κατανέμονται μεταξύ των πόρων ενός cluster. Ένα σημαντικό χαρακτηριστικό του Apache Flink είναι ότι αποτελεί ένα μια stateful λύση, δηλαδή μπορεί να γίνει επανάκτηση παλαιότερων event για την σύνθεση δεδομένων. Η επικοινωνία μεταξύ των Job Manager και των Task Manager επιτυγχάνεται μέσω του συστήματος Akka. Το σύστημα Akka αποτελεί ένα σύστημα ρόλων το οποίο προσφέρει υπηρεσίες χρονοπρογραμματισμού, καταγραφής log κτλ. Στο Flink ένας ρόλος αντιστοιχεί σε ένα container το οποίο περιλαμβάνει ένα νήμα επεξεργασίας το οποίο επεξεργάζεται τα εισερχόμενα δεδομένα. Ένα σημαντικό χαρακτηριστικό της λειτουργίας του Apache Flink είναι η διαδικασία του check pointing μέσω του οποίου επιτυγχάνεται συνέπεια των δεδομένων και ανεκτικότητα στα σφάλματα. Ο μηχανισμός του check pointing δημιουργεί συνεχώς snapshots τα οποία αποθηκεύουν την τρέχουσα κατάσταση της ροής δεδομένων σε ανθεκτικό αποθηκευτικό χώρο . Η διαδικασία ομαδοποίησης των εγγραφών σε snapshots γίνεται μέσω των stream barriers τα οποία αποτελούν «λογικά φράγματα» που εισάγονται μέσα στη ροή δεδομένων και περιλαμβάνουν κάποιο μοναδικό ID. Σε περίπτωση σφάλματος του συστήματος, το Apache Flink σταματά τους executors, τους επαναφέρει ξεκινώντας την διαδικασία από το τελευταίο διαθέσιμο checkpoint χωρίς να υπάρχει ενδεχόμενο επανεπεξεργασίας δεδομένων που έχουν ήδη επεξεργαστεί.. Ο Job Manager ενημερώνεται συνεχώς για την κατάσταση κάθε ενός από τα snapshot. Η κατάσταση ενός task είναι αποθηκευμένη στην μνήμη και στην περίπτωση όπου υπερβεί το μέγεθος του διαθέσιμου χώρου η κατάσταση αποθηκεύεται στον δίσκο. Έτσι η επεξεργασία γίνεται μέσω προσπελάσεων στην τοπική μνήμη των κόμβων (in-memory επεξεργασία), γεγονός το οποίο οδηγεί σε πολύ χαμηλούς χρόνους επεξεργασίας. Όσον αφορά την διαχείριση και κατανομή πόρων το Apache Flink μπορεί να χρησιμοποιήσει όλους τους κοινούς διαχειριστές πόρων όπως το Hadoop YARN, Apache Mesos, Kubernetes κτλ όπως επίσης και να λειτουργήσει ως ένα αυτούσιο cluster [30]. Το Apache Flink αποτελείται από 2 κύρια APIs, το DataStream API το οποίο δίνει την δυνατότητα εφαρμογής μετασχηματισμών όπως φίλτρα, aggregations, window functions κτλ σε περιορισμένες και μη ροές δεδομένων, όπως επίσης και το DataSet API το οποίο προσφέρει δυνατότητα εφαρμογής μετασχηματισμών όπως φίλτρα, μετασχηματισμούς mapping, joining, grouping κτλ. Και τα δύο APIs περιλαμβάνουν περισσότερους από 20 μετασχηματισμούς και είναι διαθέσιμα σε Java και Scala. Ακόμη το Apache Flink περιλαμβάνει ένα Table API το οποίο αποτελείται από μια γλώσσα παρόμοιας μορφής με την γλώσσα SQL, το οποίο είναι κατάλληλο για batch processing και relational stream, δίνοντας την δυνατότητα δημιουργίας σχεσιακών πινάκων από εξωτερικές πηγές ή από τα υφιστάμενα DataStreams και DataSets, όπως επίσης και εφαρμογής σχεσιακών τελεστών όπως select, join, aggregate κτλ σε αυτούς.

Πλεονεκτήματα

Το Apache Flink προσφέρει πολλά πλεονεκτήματα όσον αφορά την επεξεργασία ροών δεδομένων σε πραγματικό χρόνο. Αρχικά αποτελεί ένα σύστημα με πολύ χαμηλό latency λόγω της in-memory επεξεργασίας και υψηλό ρυθμό απόδοσης, το οποίο προσφέρει 2 ειδών επεξεργασίες δεδομένων, stream και batch processing διατηρώντας πολύ υψηλές ταχύτητες. Είναι ένα σύστημα το οποίο είναι ανεκτικό στα σφάλματα διατηρώντας την ασφάλεια και συνέπεια των δεδομένων μέσω των τεχνικών του check pointing και των stand by node. Ακόμη αποτελεί ένα υψηλά κλιμακώσιμο σύστημα με δυνατότητες επέκτασης ενός cluster σε χιλιάδες nodes, το οποίο μπορεί εύκολα να συνδυαστεί με άλλες τεχνολογίες επεξεργασίας δεδομένων μεγάλης κλίμακας όπως Apache Hadoop, Spark, MapReduce κτλ. Ένα άλλο πλεονέκτημα που προσφέρει είναι η υποστήριξη τριών ευρεία χρησιμοποιημένων γλωσσών προγραμματισμού από τα API του όπως είναι η Java, Scala και Python.

Μειονεκτήματα

Ένα από τα μειονεκτήματα που αντιμετωπίζει το Apache Flink είναι ότι παρά το γεγονός ότι προσφέρει δυνατότητες batch processing, η τεχνολογία αυτή προτιμάται μόνο για τις υπηρεσίες επεξεργασίας ροών δεδομένων σε πραγματικό χρόνο (stream processing). Ακόμη το Apache Flink δεν διαθέτει τόσο μεγάλη κοινότητα όσο αυτή του Apache Spark, έτσι υπάρχει λιγότερη διαθέσιμη βοήθεια όσον αφορά θέματα εγκατάστασης, αντιμετώπισης προβλημάτων και διαχείρισης του συστήματος.



Σχήμα 1.13 Τοπολογία αρχιτεκτονικής Apache Flink

Πηγή*:https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781786466228/1/ch01lvl1sec9/distributed-execution*

3.4 Τεχνικές διαμοιρασμού πόρων και παράλληλης επεξεργασίας δεδομένων

Η διαχείριση και επεξεργασία μεγάλου όγκου δεδομένων αποτελεί μια χρονοβόρα και δαπανηρή σε πόρους διαδικασία για ένα IoT σύστημα. Οι σύγχρονες εφαρμογές των IoT συστημάτων βασίζουν την επιτυχία της λειτουργία τους στην γρήγορη και αποδοτική επεξεργασία του μεγάλου αυτού όγκου δεδομένων. Για το λόγο αυτό κρίνεται απαραίτητη η χρήση τεχνολογιών, οι οποίες επιτρέπουν σε ένα σύστημα να διαχειρίζεται μεγάλους όγκους δεδομένων, μέσω του αποδοτικού διαμοιρασμού πόρων οποίος δίνει δυνατότητες παράλληλης επεξεργασίας των δεδομένων αυτών, χρησιμοποιώντας ένα μεγάλο αριθμό υπολογιστικών πυρήνων, αυξάνοντας την συνολική απόδοση του συστήματος. Στην παρακάτω ενότητα θα μελετηθούν διάφορες τεχνικές διαμοιρασμού πόρων και παράλληλης επεξεργασίας δεδομένων.

3.4.1 Apache Spark

Η τεχνολογία Apache Spark αποτελεί ένα framework κατανεμημένης επεξεργασίας και ανάλυσης δεδομένων μεγάλης κλίμακας υλοποιημένο σε γλώσσα προγραμματισμού Scala. Είναι λογισμικό ανοικτού κώδικα, το οποίο αναπτύχθηκε αρχικά από το Berkeley's AMPLab του University of California το 2009, και μετέπειτα δωρήθηκε στον οργανισμό Apache Software Foundation. Παρέχει μια διεπαφή μέσω της οποίας δίνεται η δυνατότητα προγραμματισμού ολόκληρων clusters για την επίτευξη παράλληλης επεξεργασίας, batch processing όπως επίσης και real-time processing. Ακόμη υποστηρίζει πολλές γλώσσες προγραμματισμού όπως Python, Java, Scala, R, SQL καθώς και διάφορες βιβλιοθήκες οι οποίες αφορούν λειτουργίες όπως streaming, machine learning, graph processing κτλ. Η τεχνολογία αυτή χρησιμοποιείται από πολλές μεγάλες εταιρείες όπως Microsoft, Apple, Facebook και IBM.

Η αρχιτεκτονική του Apache Spark αποτελείται από τα εξής κύρια συστατικά στοιχεία:

- **Spark Core:** Αποτελεί τον βασικό πυρήνα της κατανεμημένης και παράλληλης επεξεργασίας των δεδομένων, πάνω στον οποίο είναι κτισμένη όλη η λειτουργικότητα του συστήματος. Είναι υπεύθυνο για θέματα διαχείρισης μνήμης, ανάνηψης από σφάλματα, αλληλεπίδρασης με τα αποθηκευτικά μέσα, όπως επίσης για τον χρονοπρογραμματισμό, διαμοιρασμό και παρακολούθηση των εργασιών στο cluster, καθώς και για τις βασικές I/O λειτουργίες.

- **Spark Streaming:** Είναι το στοιχείο το οποίο επιτρέπει την επεξεργασία ροών δεδομένων σε σχεδόν πραγματικό χρόνο μέσω της διαδικασίας micro-batching. Χρησιμοποιεί την ικανότητα γρήγορου χρονοπρογραμματισμού των εργασιών από το Spark Core επεκτείνοντας την ήδη υπάρχουσα λειτουργία του batch processing. Κατά την διαδικασία του micro-batching γίνεται διάσπαση της εισερχόμενης ροής δεδομένων σε συνεχόμενες μικρές παρτίδες οι οποίες επεξεργάζονται με την βοήθεια των RDDs

(Resilient Distributed Datasets) μια μία πολύ μικρή χρονοκαθυστέρηση (delay). Οι RDDs αποτελούν συλλογές αντικειμένων ή στοιχείων, οι οποίες μπορούν να σπάσουν σε μικρότερα κομμάτια, επιτρέποντας την παράλληλη επεξεργασία τους στους κόμβους ενός cluster. Έτσι δίνεται η δυνατότητα επαναχρησιμοποίησης κώδικα ο οποίος αφορά το batch processing, για stream processing, κάνοντας πιο εύκολη την υλοποίηση μιας αρχιτεκτονικής λ. Στη έκδοση 2.x του Apache Spark έχει προστεθεί η λειτουργικότητα του Structured Streaming μέσω του οποίου δίνεται η δυνατότητα δημιουργίας απείρων dataframes και datasets για την αντιμετώπιση προβλημάτων που αφορούν την συνένωση και παράδοση των δεδομένων.

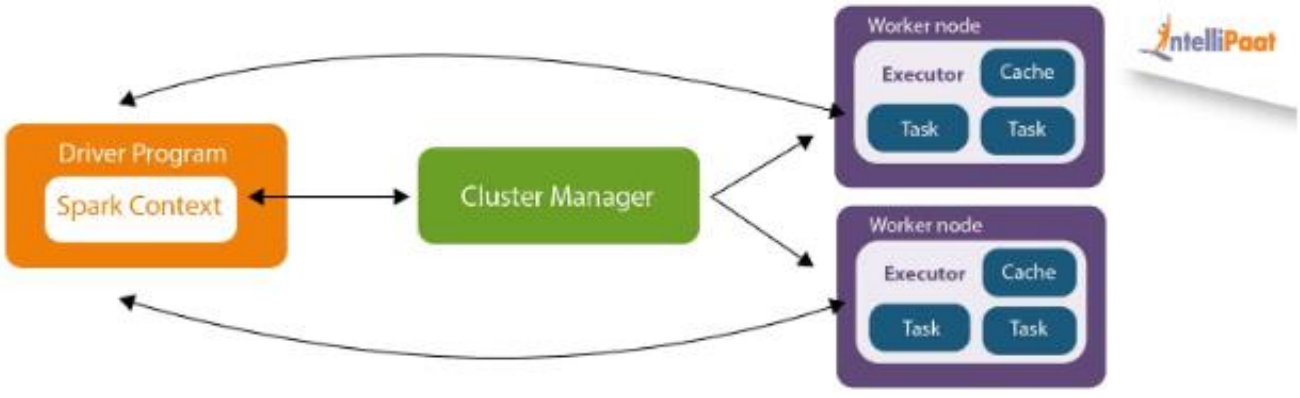
- **Spark SQL:** Αποτελεί το στοιχείο της αρχιτεκτονικής το οποίο υποστηρίζει διαχείριση των Dataframes μέσω της γλώσσας DSL και των γλωσσών προγραμματισμού Scala, Java, Python, προσφέροντας υποστήριξη σε δομημένα και μη δεδομένα. Ακόμη παρέχει υποστήριξη για την γλώσσα προγραμματισμού SQL μέσω ενός CLI και των ODBC/JDBC servers καθώς και δυνατότητα συμβατότητας με άλλες τεχνολογίες όπως JSON, HDFS, Apache Hive, Apache Parquet κτλ.

- **MLlib Machine Learning Library:** Είναι ένα κατανεμημένο framework μηχανικής μάθησης στο οποίο είναι ενσωματωμένοι διάφοροι αλγόριθμοι μηχανικής μάθησης και στατιστικής όπως k-means, random forests, LDA (Latent Dirichlet Allocation), ALS (Alternate Least Squares), PCA (Principal Component Analysis) κτλ. Παρέχει δυνατότητα εκμάθησης μοντέλων χρησιμοποιώντας τις γλώσσες προγραμματισμού Python ή R, καλύπτοντας τα βασικά μέρη της μηχανικής μάθησης δηλαδή, classification, regression, clustering και filtering. Η μοντελοποίηση και εκπαίδευση νευρωνικών δικτύων γίνεται μέσω των Deep Learning Pipelines.

- **GraphX:** Αποτελεί ένα κατανεμημένο framework επεξεργασίας γράφων το οποίο παρέχει 2 διαφορετικά APIs, ένα που ακολουθεί το Pregel μοντέλο και ένα με πιο γενικό MapReduce στυλ, για την υλοποίηση παράλληλων αλγορίθμων όπως ο PageRank της Google. Οι αλγόριθμοι αυτοί χρησιμοποιούν τους RDD για την μοντελοποίηση των δεδομένων καθιστώντας το framework αυτό κατάλληλο για γράφους οι οποίοι δεν χρειάζονται συνεχείς ενημερώσεις.

Λειτουργία

Η αρχιτεκτονική εκτέλεσης των εργασιών του Apache Spark βασίζεται τοπολογία DAG. Το Apache Spark δημιουργεί μια τοπολογία DAG με βάση τις εντολές που ορίζει ο χρήστης όσον αφορά την επεξεργασία των δεδομένων. Η τοπολογία αυτή λειτουργεί ως επίπεδο χρονοπρογραμματισμού, δηλαδή ορίζει ποιες εργασίες θα εκτελεστούν, σε ποιους κόμβους θα γίνει η εκτέλεση αυτή και με ποια σειρά. Το Apache Spark ακολουθεί το μοντέλο αρχιτεκτονικής master-worker. Αποτελείται από τον Driver ο οποίος έχει τον ρόλο του Master και από διάφορους Workers nodes οι οποίοι εκτελούν τις εργασίες που τους ανατίθενται. Το Driver πρόγραμμα καλεί το main πρόγραμμα της εφαρμογής και δημιουργεί ένα SparkContext το οποίο περιέχει όλες τις κύριες λειτουργίες του Spark. Μέσω του Spark Driver γίνεται η μετάφραση του κώδικα ο οποίος έχει γραφτεί από κάποιο χρήστη, σε εργασίες οι οποίες θα εκτελεστούν στο cluster [31]. Γενικά ο Spark Driver και το SparkContext παρακολουθούν την εξέλιξη των εργασιών και το Spark μπορεί να χειριστεί μόνο του θέματα που αφορούν την διαχείριση των πόρων και του cluster. Παρόλα αυτά κρίνεται απαραίτητη η ύπαρξη ενός Cluster Manager μεταξύ Driver και Workers για την διαχείριση των πόρων. Μερικά παραδείγματα αυτών είναι ο YARN (Yet Another Resource Negotiator), Apache Mesos, Kubernetes κτλ. Έτσι εργασία σπάζει σε μικρότερα κομμάτια με την βοήθεια των RDD και διανέμεται από τον Cluster Manager στους Worker nodes για εκτέλεση. Η διάρκεια ζωής των κόμβων που εκτελούν τις εργασίες είναι ίδια με αυτή της εφαρμογής του Spark. Αφού οι Workers εκτελέσουν τις εργασίες που τους ανατέθηκαν από τον Cluster Manager, επιστρέφουν τα αποτελέσματα πίσω στο SparkContext. Υπάρχει δυνατότητα αύξησης του αριθμού των Workers, έτσι ώστε μια εργασία να μπορεί να σπάσει σε περισσότερα λογικά κομμάτια αυξάνοντας την συνολική απόδοση του συστήματος. Επίσης η λειτουργία του Apache Spark απαιτεί κάποιο κατανεμημένο σύστημα αρχείων όπως το HDFS, Amazon S3, Cassandra, MapR-FS, OpenStack Swift κτλ. Το Apache Spark αποτελεί αναπόσπαστο μέρος διάφορων υπηρεσιών από μεγάλες εταιρίες όπως το Amazon EMR, Google Cloud Dataproc, Microsoft Azure HDInsight.



Σχήμα 1.14 Αρχιτεκτονική Apache Spark

Πηγή: *https://intellipaat.com/blog/tutorial/spark-tutorial/spark-architecture/*

Πλεονεκτήματα

Μερικά από τα πλεονεκτήματα που προσφέρει το Apache Spark είναι η αυξημένη απόδοση και μεγάλη ταχύτητα στην επεξεργασία δεδομένων η οποία μπορεί να φτάσει μέχρι και 100 φορές γρηγορότερη στην μνήμη και 10 φορές γρηγορότερη στον δίσκο σε σύγκριση με το Hadoop. Μέσω της κατανεμημένης αποθήκευσης των δεδομένων στην μνήμη (cache) κατά την διάρκεια της επεξεργασίας μπορεί να διαχειριστεί αποδοτικά μεγάλο όγκο δεδομένων χωρίς ιδιαίτερο πρόβλημα. Ακόμη προσφέρει μεγάλη ευκολία στην χρήση όπως επίσης και μεγάλη συμβατότητα αφού υποστηρίζει ένα μεγάλο εύρος γλωσσών προγραμματισμού και άλλων τεχνολογιών. Η ικανότητα υποστήριξης διάφορων λειτουργιών όπως streaming, τεχνικές μηχανικής μάθησης, επεξεργασία γράφων κτλ. καθιστά το Apache Spark μια πιο ολοκληρωμένη λύση για ένα σύστημα σε σύγκριση με άλλες τεχνολογίες. Επίσης μπορεί να λειτουργήσει σε πολλά περιβάλλοντα αφού υποστηρίζει EC2, Hadoop YARN, Mesos, Kubernetes καθώς και πολλές τεχνικές αποθήκευσης (HDFS, Alluxio, Apache Cassandra, Apache HBase, Apache Hive).

Μειονεκτήματα

Ένα από τα μειονεκτήματα που αντιμετωπίζει το Spark αφορά την διαχείριση μικρού μεγέθους αρχείων σε περίπτωση που συνδυαστεί με κάποιο σύστημα διαχείρισης αρχείων όπως το HDFS, το οποίο δεν είναι σχεδιασμένο για αυτά. Έτσι η εξάρτηση αυτή από κάποιο σύστημα διαχείρισης αρχείων αποτελεί και αυτή από μόνη της ένα μειονέκτημα. Ακόμη η ιδιότητα που έχει το Spark να κρατάει τα δεδομένα στην μνήμη RAM επιφέρει μεγάλη κατανάλωση σε μνήμη, πράγμα το οποίο μπορεί να αποτελέσει πρόβλημα στην περίπτωση όπου υπάρχει ανάγκη για μια λύση με χαμηλό κόστος αφού η μνήμη είναι σχετικά ακριβή. Επίσης μερικές διαδικασίες όπως το partitioning και caching για εκτελούνται σωστά και αποδοτικά, πρέπει να ελέγχονται χειροκίνητα.

3.4.2 Apache Hadoop

Το Apache Hadoop περιλαμβάνει μια συλλογή λειτουργιών αποτελούμενες από λογισμικό ανοικτού κώδικα υλοποιημένο σε Java, οι οποίες είναι κατάλληλες για την κατανεμημένη επεξεργασία, ανάλυση και αποθήκευση δεδομένων μεγάλης κλίμακας. Δημιουργήθηκε από τους Doug Cutting και Mike Cafarella το 2006 και είναι σχεδιασμένο να τρέχει σε clusters υπολογιστών, οι οποίοι αποτελούνται κυρίως από φθηνό hardware επιτυγχάνοντας υψηλή απόδοση. Ο πυρήνας του Apache Hadoop αποτελείται από το επεξεργαστικό μέρος το οποίο λειτουργεί με βάση το προγραμματιστικό μοντέλο MapReduce, όπως επίσης και από ένα κατανεμημένο σύστημα αρχείων το HDFS (Hadoop Distributed File System) του οποίου η λειτουργία έχει αναλυθεί στην ενότητα **«3.2-Τεχνικές αποθήκευσης δεδομένων μεγάλης κλίμακας».** Το Apache Hadoop χρησιμοποιείται από πολλές μεγάλες εταιρείες όπως Facebook, Google, Twitter, LinkedIn, Yahoo κτλ. Παρακάτω θα αναλυθεί η λειτουργία του επεξεργαστικού επιπέδου του Hadoop, το Hadoop MapReduce.

Όπως αναφέρθηκε κατά την ανάλυση της λειτουργίας του HDFS, το Hadoop ακολουθεί την αρχιτεκτονική master/worker. Στην περίπτωση του MapReduce επιπέδου έχουμε τους εξής 2 ειδών nodes:

- **Job Tracker (Master):** Ο κόμβος αυτός αποτελεί τον αρχηγό μέσα στο cluster ο οποίος δίνει διάφορες εντολές προς τους υπόλοιπους κόμβους. Είναι υπεύθυνος για την λήψη των αιτημάτων εκτέλεσης μιας MapReduce διαδικασίας από τους χρήστες, όπως επίσης και για την επικοινωνία με τον Namenode του HDFS με σκοπό την εξασφάλιση της τοποθεσίας των δεδομένων που θα χρειαστούν κατά την εκτέλεση της εργασίας καθώς και άλλων μεταδεδομένων.

- **Task Tracker (Worker):** Οι κόμβοι αυτοί αποτελούν τους εργάτες μέσα στο cluster, οι οποίοι εκτελούν τις διάφορες εργασίες που τους ανατίθενται. Η αρχιτεκτονική του Apache Hadoop περιλαμβάνει πολλούς τέτοιους κόμβους οι οποίοι είναι υπεύθυνοι για την λήψη των MapReduce εργασιών από τον Job Tracker, την εκτέλεση τους καθώς και την λήψη του κώδικα ο οποίος θα εφαρμοστεί πάνω στα αρχεία με τα δεδομένα κατά την εκτέλεση μιας MapReduce εργασίας.

Λειτουργία

Η αρχιτεκτονική της λειτουργίας του επεξεργαστικού επιπέδου του Apache Hadoop βασίζεται στο προγραμματιστικό μοντέλο MapReduce το οποίο είναι κατάλληλο για την παράλληλη επεξεργασία δεδομένων μεγάλης κλίμακας. Η επεξεργασία των δεδομένων γίνεται μέσω της δημιουργία πολλών ζευγαριών της μορφής key-value όπου κάθε κλειδί key λειτουργεί σαν ένα αναγνωριστικό της τιμής value. Μια MapReduce εργασία αποτελείται από πολλές μικρότερες Map και Reduce εργασίες οι οποίες εφαρμόζονται σε συγκεκριμένα κομμάτια των δεδομένων, διαμοιράζοντας έτσι τις εργασίες σε όλο το υπολογιστικό cluster.

Μια MapReduce εργασία κατά την εκτέλεση της περνά από τις εξής 2 κύρια φάσεις:

- **Map Phase:** Αποτελεί το πρώτο στάδιο εκτέλεσης μια MapReduce εργασίας. Κατά την διάρκεια της φάσης αυτής γίνεται η αρχική επεξεργασία των δεδομένων, τα οποία βρίσκονται συνήθως σε κάποιο αρχείο ή φάκελο μέσα στο HDFS. Τα δεδομένα σπάζουν σε πολλά input splits τα οποία αποτελούν λογικές αναπαραστάσεις των δεδομένων οι οποίες μετέπειτα θα επεξεργαστούν από κάποιον Mapper. Ακολούθως ο RecordReader μετατρέπει τα input splits σε key-value ζεύγη στα οποία θα γίνει η περαιτέρω επεξεργασία. Τα ζεύγη αυτά επεξεργάζονται από τον Mapper μέσω συναρτήσεων οι οποίες είναι ορισμένες από τον χρήστη, δημιουργώντας νέα, διαφορετικά key-value ζεύγη χρησιμοποιώντας τα παλιά, τα οποία και προωθούνται στον Combiner για περαιτέρω επεξεργασία. Για κάθε Mapper υπάρχει ένας Combiner ο οποίος συμπεριφέρεται σαν ένας μικρός Reducer, ο οποίος δεν εκτελείται κάθε φορά. Επεξεργάζεται περαιτέρω τα δεδομένα με σκοπό την μείωση του μεγέθους των δεδομένων που πρέπει να μεταδοθούν μέσα στο δίκτυο. Ένα σημαντικό κομμάτι της φάσης αυτής είναι ο Partitioner ο οποίος παίρνει τα key-value ζεύγη από τον Mapper και κατανέμει ισάξια τα κλειδιά σε partitions χρησιμοποιώντας το modulus key.hashcode()%(αριθμός από Reducers). Αυτό εξασφαλίζει ότι ένα κλειδί με την ίδια τιμή αλλά από διαφορετικό Mapper θα καταλήξει στον ίδιο Reducer. Τέλος τα δεδομένα αποθηκεύονται στο τοπικό σύστημα περιμένοντας να χρησιμοποιηθούν από τους Reducers.

- **Reduce Phase:** Πριν τα δεδομένα να περάσουν για επεξεργασία στους Reducers περνούν μέσα από μια ενδιάμεση διαδικασία η οποία ονομάζεται «shuffle and sort». Κατά την διάρκεια της διαδικασίας αυτής γίνεται λήψη των δεδομένων των οποίον γράφτηκαν από τον Partitioner και ταξινόμηση τους σε μια μεγάλη λίστα δεδομένων. Αυτό αποσκοπεί στην ομαδοποίηση των όμοιων κλειδιών έτσι ώστε να υπάρχει εύκολη προσπέλαση τους από τις reduce εργασίες. Έπειτα τα δεδομένα περνούν σαν είσοδος στους Reducers οι οποίοι εφαρμόζουν τις reducer συναρτήσεις μια φορά ανά ομάδα κλειδιών. Ένας Reducer μπορεί να φιλτράρει, συνενώσει και συνδυάσει δεδομένα με πολλούς τρόπους, παράγοντας μηδέν ή περισσότερα νέα key-value ζεύγη ανάλογα με τον ορισμό της reduce συνάρτησης. Το τελικό στάδιο της διαδικασίας πραγματοποιείται από τον RecordWriter ο οποίος παίρνει τα key-value ζεύγη από τους Reducers, τα οποία και γράφει σαν αποτέλεσμα σε κάποιο αρχείο στο HDFS με τρόπο ο οποίος ορίζεται από το OutputFormat.

Παρομοίως με το Apache Spark, το Hadoop χρησιμοποιεί και αυτό έναν cluster manager και συγκεκριμένα τον YARN, για την διαχείριση των πόρων καθώς και χρονοπρογραμματισμό και παρακολούθηση των εργασιών. Ο YARN αποτελείται από 2 μέρη, τον ResourceManager και τον NodeManager. Ο NodeManager είναι υπεύθυνος για την παρακολούθηση των πόρων (CPU, memory, network, disk capacity κτλ.) και αναφορά στον ResourceManager, ενώ ο ResourceManager είναι υπεύθυνος για τον διαμερισμό τον πόρων στις διάφορες εφαρμογές. Ο Application Master διαπραγματεύεται πόρους από τον ResourceManager και συνεργάζεται με τον NodeManager για την εκτέλεση των εργασιών.

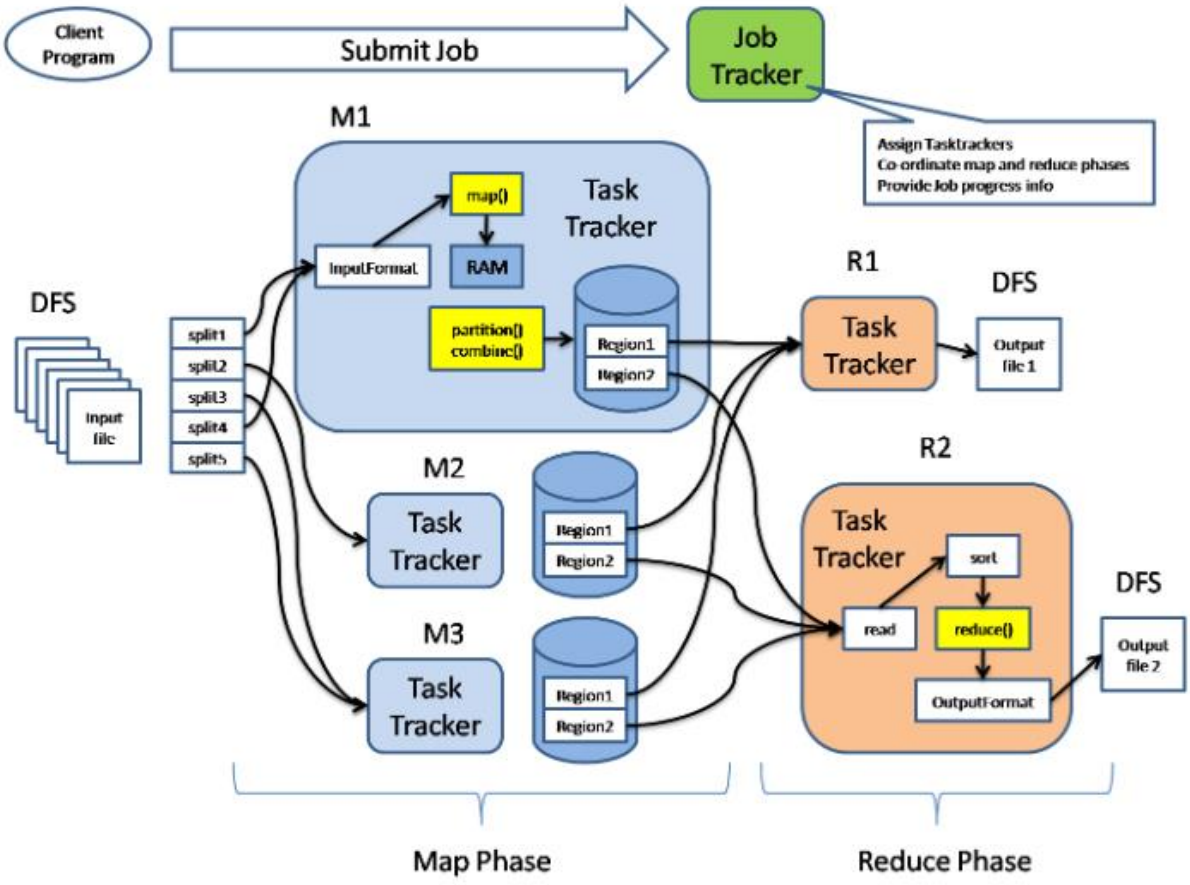
Πλεονεκτήματα

Το Apache Hadoop προσφέρει πολλά πλεονεκτήματα όσον αφορά την γρήγορη και αποδοτική επεξεργασία δεδομένων μεγάλης κλίμακας. Είναι ικανό να προσφέρει αποδοτική και ταχεία επεξεργασία δεδομένων λόγω της κατανεμημένης αρχιτεκτονικής του και της δυνατότητας για παράλληλη επεξεργασία σε πολλούς πυρήνες ενός cluster. Ακόμη αποτελεί μια φθηνή λύση αφού η λειτουργία του δεν απαιτεί ακριβό hardware, προσφέροντας μεγάλη συμβατότητα με άλλες τεχνολογίες όπως Apache Spark, Flint όπως επίσης και με διάφορες γλώσσες προγραμματισμού όπως C, C++, Python, Perl, Ruby, Groovy κτλ. Ένα από τα μεγαλύτερα πλεονεκτήματα του Hadoop είναι η δυνατότητα κλιμακωσιμότητας που προσφέρει μέσω της προσθήκης μηχανημάτων σε ένα cluster ή RAM και δίσκους σε ένα μηχάνημα. Επίσης προσφέρει ευκολία στην χρήση αφού οι προγραμματιστές χρειάζεται να γνωρίζουν μόνο το μοντέλο MapReduce χωρίς να ανησυχούν για την κατανεμημένη και παράλληλη επεξεργασία των δεδομένων η οποία γίνεται αυτόματα από το σύστημα. Λόγω της αξιοπιστίας που προσφέρει η χρήση του HDFS ως σύστημα διαχείρισης αρχείων, υπάρχει μεγάλη ανεκτικότητα στα σφάλματα από το Hadoop, αφού λόγω των πολλαπλών αντιγράφων που δημιουργούνται τα δεδομένα διατηρούνται ασφαλή και διαθέσιμα ανεξαρτήτως σφαλμάτων. Ακόμη η χρήση του HDFS προσφέρει μεγάλο εύρος όσον αφορά τους τύπους δεδομένων που μπορεί το σύστημα να διαχειριστεί, δίνοντας δυνατότητα υποστήριξης αρχείων τύπου XML, CSV, text, εικόνες κτλ. Επίσης λόγω του σχεδιασμού λειτουργίας του Hadoop, παρατηρείται μείωση στην συμφόρηση του δικτύου αφού κάθε εργασία σπάει σε μικρότερες εργασίες οι οποίες διανέμονται στους κόμβους οδηγώντας σε μεταφορά μικρής ποσότητας κώδικα κοντά στα δεδομένα παρά μεγάλο όγκο δεδομένων κοντά στον κώδικα. Ένα άλλο πλεονέκτημα που προσφέρει η χρήση του Hadoop είναι το γεγονός ότι αποτελεί λογισμικό ανοικτού κώδικα έτσι ο πηγαίος του κώδικας διατίθεται δωρεάν στους χρήστες, δίνοντας την δυνατότητα τροποποίησης του ανάλογα με τις υφιστάμενες ανάγκες [32].

Μειονεκτήματα

Παρά το πολλά πλεονεκτήματα που προσφέρει το Hadoop υπάρχουν κάποια μειονεκτήματα όσον αφορά την λειτουργία του. Ένα από τα προβλήματα που αντιμετωπίζει ως σύστημα προέρχονται από την χρήση του HDFS. Το πρόβλημα αυτό αφορά την διαχείριση αρχείων μικρού μεγέθους, τα οποία το Hadoop δεν είναι σχεδιασμένο να διαχειρίζεται. Στην περίπτωση ύπαρξης τεράστιου αριθμού αρχείων μικρού μεγέθους, δηλαδή μικρότερου από το default μέγεθος του block (128MB), ο Namenode θα υπερφορτωθεί αφού είναι ο υπεύθυνος διαχείρισης του namespace στο HDFS. Παρά το γεγονός ότι το Hadoop αποτελεί μία πολύ γρήγορη και αποδοτική λύση όσον αφορά την επεξεργασία δεδομένων μεγάλης κλίμακας οι εργασίες Map και Reduce απαιτούν κάποιο χρονικό διάστημα για να εκτελεστούν λόγω της διανομής των δεδομένων στο cluster. Για το λόγο αυτό συνήθως κρίνεται απαραίτητος ο συνδυασμός του με το Apache Spark το οποίο έχει αποδειχτεί 100 φορές γρηγορότερο λόγω της in-memory επεξεργασίας η οποία δεν προϋποθέτει μετακίνηση των δεδομένων. Ακόμη το Hadoop αυτούσιο προσφέρει υποστήριξη μόνο για batch processing και όχι για stream processing και επεξεργασία σε πραγματικό χρόνο. Η υποστήριξη για stream processing και επεξεργασία σε πραγματικό χρόνο προϋποθέτει συνδυασμό με άλλες τεχνολογίες όπως το Apache Spark για να επιτευχθεί. Επίσης το

Hadoop υποστηρίζει την τεχνολογία διαπίστευσης Kerberos η οποία είναι πολύπλοκη όσον αφορά την διαχείριση της. Επιπλέον το Hadoop παρουσιάζει προβλήματα ασφαλείας τα οποία προκύπτουν από την απουσίας κρυπτογράφησης σε επίπεδο δικτύου και αποθήκευσης δεδομένων.



Σχήμα 1.15 Αρχιτεκτονική MapReduce

Πηγή: *http://a4academics.com/tutorials/83-hadoop/840-map-reduce-architecture*

3.5 Τεχνικές ενορχήστρωσης, συντονισμού συστήματος και απομόνωσης εφαρμογών

Λόγω της μεγάλης κλίμακας και ποικιλομορφίας των δεδομένων των οποίων καλούνται να διαχειριστούν τα σύγχρονα IoT συστήματα, υπάρχει αυξημένη ανάγκη για κατανεμημένη επεξεργασία, χρησιμοποιώντας απαιτητικές εφαρμογές και αλγορίθμους σε μεγάλα cluster υπολογιστών. Για τον λόγο αυτό κρίνεται απαραίτητη η σωστή ενορχήστρωση και συντονισμός των επιμέρους κομματιών ενός κατανεμημένου συστήματος επεξεργασίας δεδομένων, για την εξασφάλιση της ομαλής και αποδοτικής λειτουργίας του. Στην παρακάτω ενότητα θα μελετηθούν τεχνικές ενορχήστρωσης και συντονισμού συστήματος όπως επίσης και τεχνικές απομόνωσης εφαρμογών.

3.5.1 Docker

Η τεχνολογία Docker αποτελεί μια πλατφόρμα PaaS (platform as a service) η οποία χρησιμοποιείται ως μέσω ανάπτυξης, κατανομής και εκτέλεσης εφαρμογών σε ένα κατανεμημένο περιβάλλον. Σχεδιάστηκε και αναπτύχθηκε το 2013 από την Docker, Inc. και αποτελείται από λογισμικό ανοικτού κώδικα, του οποίου η λειτουργία προσφέρει δυνατότητες εικονικοποίησης (Virtualization) σε επίπεδο λειτουργικού συστήματος. Χρησιμοποιεί ειδικές οντότητες τους containers, οι οποίοι ενθυλακώνουν τις εφαρμογές μαζί με όλα τα απαραίτητα στοιχεία που χρειάζονται για την λειτουργία τους όπως βιβλιοθήκες και άλλες εξαρτήσεις, σε ένα απομονωμένο πακέτο το οποίο ανεξαρτητοποιείται από την υποδομή του host.

Λειτουργία

Όπως έχει προ αναφερθεί η αρχιτεκτονική λειτουργίας της πλατφόρμας Docker βασίζεται στην χρήση των containers. Τα Docker containers αποτελούν ελαφριά, αυτόνομα, απομονωμένα περιβάλλοντα τα οποία διαθέτουν οτιδήποτε χρειάζονται για την εκτέλεση εφαρμογών. Δεν εξαρτώνται από το λειτουργικό σύστημα στο οποίο τρέχει ο host προσφέροντας έτσι, την δυνατότητα δημιουργίας και εκτέλεσης εφαρμογών των οποίων οι containers τρέχουν διαφορετικά OS images (Linux, Windows, Mac κτλ), χρησιμοποιώντας από κοινού τον πυρήνα λειτουργικού του host. Διαφέρουν από τις κοινές εικονικές μηχανές (VMs) αφού τρέχουν απευθείας μέσα στον πυρήνα του host χωρίς την βοήθεια κάποιου hypervisor. Για τον λόγο αυτό χρησιμοποιούν πολύ λιγότερους πόρους, πράγμα που σημαίνει ότι με το ίδιο υλικό μπορούμε να τρέξουμε περισσότερα containers από ότι εικονικές μηχανές. Ακόμη λόγο των πολύ μικρών απαιτήσεων των Docker containers σε πόρους, γίνεται εφικτό να τρέξουμε Docker containers πάνω σε κάποιον host ο οποίος αποτελεί ο ίδιος μια εικονική μηχανή. Τα Docker containers δημιουργούνται μέσω των Docker images οι οποίες αποτελούν ειδικά templates με οδηγίες και πληροφορίες για την δημιουργία ενός container. Για παράδειγμα μπορούμε να δημιουργήσουμε ένα image βασισμένο σε κάποιο λειτουργικό σύστημα, το οποίο image θα εγκαθιστά αυτόματα κάποιο server ή λογισμικό που είναι απαραίτητο για την εκτέλεση της εφαρμογής μας. Έτσι δίνεται η δυνατότητα δημιουργίας πολλών containers από το ίδιο image τα οποία θα έχουν τα

χαρακτηριστικά και λειτουργίες που ορίζονται από το υφιστάμενο Docker image. Η κεντρική ιδέα της αρχιτεκτονικής της πλατφόρμας Docker υλοποιείται μέσω της client – server αρχιτεκτονικής. Ο Docker client επικοινωνεί μέσω ενός REST API με τον Docker daemon ο οποίος μπορεί να βρίσκεται στο ίδιο σύστημα ή και σε κάποιο απομακρυσμένο.

Τα κύρια μέρη της αρχιτεκτονικής της πλατφόρμας Docker είναι:

- **Docker client:** Αποτελεί το μέσω επικοινωνίας του χρήστη με την πλατφόρμα Docker. Προσφέρει την δυνατότητα χρήσης διάφορων εντολών όπως 'docker build', 'docker run', 'docker pull' κτλ. οι οποίες μεταφέρονται μέσω ενός REST API σε έναν ή περισσότερους dockerd προς εκτέλεση.

- **Docker daemon (dockerd):** Είναι υπεύθυνος για την λήψη των API αιτημάτων από τον Docker client και εκτέλεση τους. Τα αιτήματα αφορούν την διαχείριση των Docker images, containers, δικτύου κτλ.

- **Docker registry:** Αποτελεί έναν αποθηκευτικό χώρο για τα Docker images. Κατά την εκτέλεση διαφόρων εντολών, γίνεται έλεγχος στο Docker registry για την εύρεση και λήψη των επιθυμητών Docker images. Το προεπιλεγμένο Docker registry είναι το Docker Hub αλλά προσφέρεται δυνατότητα δημιουργίας και χρήσης κάποιου άλλου δημοσίου ή ιδιωτικού registry [33].

Ακόμη προσφέρεται η δυνατότητα χρήσης του Docker Swarm mode, το οποίο επιτρέπει ένα σύνολο φυσικών ή εικονικών μηχανών να ενταχθούν μαζί σε ένα cluster προσφέροντας έτσι δυνατότητες ενορχήστρωσης και διαχείρισης στο cluster αυτό. Ένα Docker Swarm αποτελείται από πολλούς hosts οι οποίοι ενεργούν τόσο σαν managers, όσο και σαν workers οι οποίοι εκτελούν το σύνολο των εργασιών (swarm services) που τους ανατίθενται.

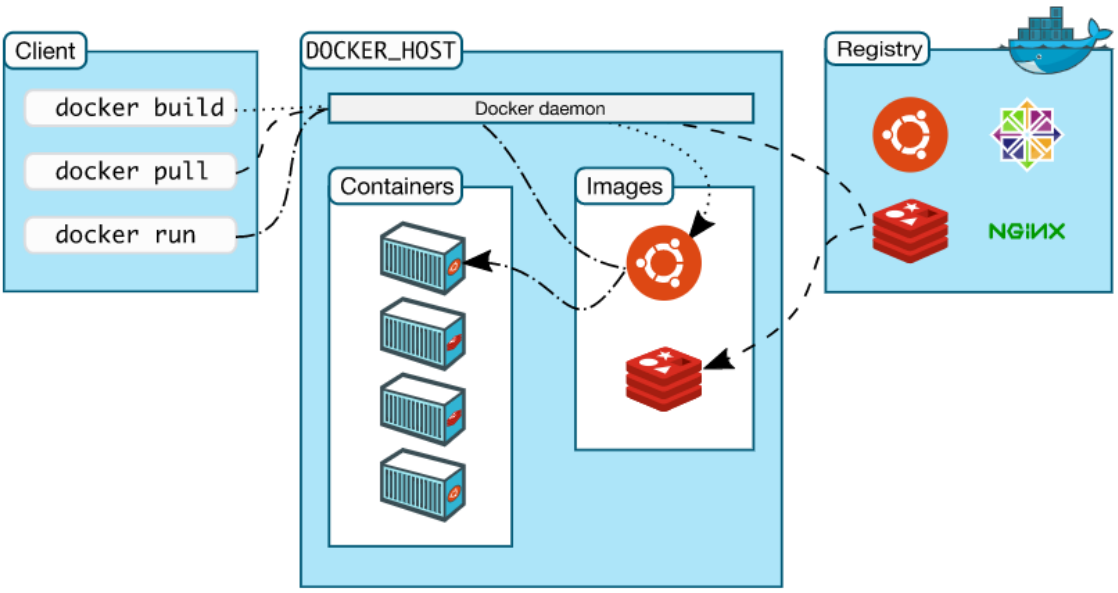
Πλεονεκτήματα

Η χρήση της πλατφόρμας Docker προσφέρει πολλά πλεονεκτήματα όσον αφορά την ανάπτυξη και εκτέλεση εφαρμογών. Αρχικά η χρήση των containers τα οποία αποτελούν μια πολύ πιο ελαφριά προσέγγιση σε σύγκριση με τις κοινές εικονικές μηχανές οδηγούν σε μεγάλη μείωση του χρόνου και κόστους ανάπτυξης και εκτέλεσης εφαρμογών, ειδικά σε περιπτώσεις όπου η χρήση εικονικών μηχανών θα αποτελούσε υπερβολή ή σε περιπτώσεις όπου το σύστημα υπάγεται σε περιορισμούς πόρων. Ακόμη η χρήση της τεχνολογίας Docker προσφέρει μεγάλη απλότητα, ευκολία και ευελιξία στον χρήστη αφού δίνει την δυνατότητα σχεδιασμού ενός Docker image με βάση τις απαιτήσεις του χρήστη, μεσώ του οποίου μπορούν γρήγορα και εύκολα να δημιουργηθούν και να εκτελεστούν πολλά containers χωρίς την ανάγκη επαναρύθμισης ή επανασχεδιασμού μειώνοντας έτσι και τον απαιτούμε χρόνο από την φάση σχεδιασμού μέχρι την φάση ανάπτυξης και παράδοσης. Το γεγονός ότι τα Docker containers δεν εξαρτώνται από το λειτουργικό σύστημα στο οποίο τρέχει ο host, προσφέρει την δυνατότητα μεταφοράς εφαρμογών οι οποίες ενσωματώνονται μαζί με τις εξαρτήσεις και ρυθμίσεις τους σε containers, σε άλλα μηχανήματα ανεξαρτήτως λειτουργικού συστήματος τους, στα οποία τρέχει μια πλατφόρμα Docker χωρίς καμία

ανάγκη προσαρμογής στο νέο σύστημα. Επίσης δίνεται η δυνατότητα εύκολης πρόσβασης σε δημόσια registries τα οποία διαθέτουν διάφορα Docker images, κάνοντας έτσι απλούστερη και ευκολότερη την μετακίνηση κώδικα και εφαρμογών έτοιμων για παραγωγή. Ένα πλεονέκτημα που προσφέρει η χρήση του Docker σε Swarm mode είναι το γεγονός ότι δεν χρειάζεται χειροκίνητη επανεκκίνηση ενός swarm service σε περίπτωση όπου υπάρξει αλλαγή στις ρυθμίσεις του. Η διαδικασία αυτή γίνεται αυτόματα από το σύστημα μέσω της δημιουργίας και εκτέλεσης νέων εργασιών ενός service, οι οποίες ικανοποιούν τις απαιτήσεις των νέων ρυθμίσεων. Τέλος η πλατφόρμα Docker μπορεί να εξασφαλίσει πλήρη απομόνωση μεταξύ των εφαρμογών, μέσω της παροχής πλήρους ελέγχου στον χρήστη όσον αφορά την διαχείριση επικοινωνίας μεταξύ των containers, γεγονός το οποίο ενισχύει το επίπεδο ασφαλείας και προστασίας του συστήματος και των ιδίων των εφαρμογών.

Μειονεκτήματα

Παρά τον μεγάλο αριθμό πλεονεκτημάτων προσφέρει η χρήση της πλατφόρμας Docker, αντιμετωπίζει μερικά μειονεκτήματα ως προς την λειτουργία της, όπως το γεγονός ότι δεν είναι σχεδιασμένη για αποδοτική ανάπτυξη και εκτέλεση εφαρμογών μέσα σε containers οι οποίες απαιτούν γραφικό περιβάλλον. Ακόμη μετά την καταστροφή ενός container όλα τα δεδομένα του χάνονται έτσι υπάρχει ανάγκη για κάποιο χώρο αποθήκευσης για δεδομένα μακράς διαρκείας. Η επιλογή που προσφέρεται από την πλατφόρμα Docker είναι τα Docker Data Volumes, αλλά η ανάγκη για κάποια καλύτερη λύση ακόμη υφίσταται. Ένα άλλο μειονέκτημα που αντιμετωπίζει η τεχνολογία Docker είναι το γεγονός ότι δεν οφελούνται όλοι οι τύποι εφαρμογών από την χρήση των container. Αν οι εφαρμογές δεν αφορούν ξεχωριστά microservices, το μόνο όφελος που προκύπτει από την χρήση των container, είναι η απλότητα στην εκτέλεση και μεταφορά των εφαρμογών η οποία προσφέρεται μέσω της ενσωμάτωσης των εφαρμογών και των εξαρτήσεων τους σε ένα container.



Σχήμα 1.16 Αρχιτεκτονική Docker

Πηγή: *https://docs.docker.com/get-started/overview/*

3.5.2 Kubernetes

Η τεχνολογία Kubernetes αποτελεί μια ευέλικτη πλατφόρμα ενορχήστρωσης και συντονισμού κατανεμημένων συστημάτων η οποία λόγω της λειτουργίας της πήρε την ονομασία της από την ελληνική λέξη «Κυβερνήτης». Σχεδιάστηκε από την Google όπου αποτελεί λογισμικού ανοικτού κώδικα από το 2014, το οποίο σήμερα συντηρείται από τον οργανισμό Cloud Native Computing Foundation υλοποιημένο σε γλώσσα προγραμματισμού Golang. Χρησιμοποιείται ως μέσο αυτόματης διαχείρισης και εκτέλεσης απομονωμένων εφαρμογών χρησιμοποιώντας ειδικούς εικονικούς χώρους μέσα στο cluster οι οποίοι μοιάζουν με VMs και ονομάζονται containers. Η πλατφόρμα Kubernetes προσφέρει την δυνατότητα στον χρήστη να ορίζει τον τρόπο με τον οποίο τρέχουν και αλληλοεπιδρούν οι εφαρμογές μεταξύ τους ή με το εξωτερικό περιβάλλον.

Λειτουργία

Η αρχιτεκτονική της τεχνολογίας Kubernetes αποτελεί μια αρχιτεκτονική της οποίας η λειτουργία είναι βασισμένη σε μια cluster τοπολογία από φυσικές ή εικονικές μηχανές οι οποίες χρησιμοποιούν ένα κοινό δίκτυο για να επικοινωνούν μεταξύ τους. Ο διαχειριστής δημιουργεί και ανεβάζει ένα αρχείο manifest της μορφής JSON ή YAML, στο οποίο ορίζει διάφορες ρυθμίσεις ως προς το τι θα δημιουργηθεί και πως θα διαχειριστεί από το σύστημα. Αφού δημιουργηθεί η κατάλληλη υποδομή με βάση τις οδηγίες του αρχείου αυτού, η πλατφόρμα Kubernetes παρακολουθεί συνεχώς τα στοιχεία του cluster έτσι ώστε να εξασφαλίζεται πλήρης συνέπεια μεταξύ της τρέχουσας κατάστασης των εφαρμογών και της επιθυμητής. Η αρχιτεκτονική αυτή εμπεριέχει 2 ειδών nodes, τον master server (control plane) και τους workers (Nodes). Ο master server ή αλλιώς control plane είναι υπεύθυνος για την λήψη αιτημάτων από τους χρήστες, διαπίστευση, διαχείριση δικτύου και κλιμακωσιμότητας, όπως επίσης και για τον έλεγχο της βιωσιμότητας του συστήματος. Λαμβάνει το αρχείο manifest και βρίσκει τρόπο να υλοποιήσει τις ρυθμίσεις που ορίζονται στο αρχείο αυτό, με βάση την τρέχουσα κατάσταση του συστήματος και τις απαιτήσεις του αρχείου.

Ο control plane αποτελείται από τα εξής κύρια μέρη:

- **etcd:** Αποτελεί έναν κατανεμημένο key-value χώρο αποθήκευσης ο οποίος χρησιμοποιείται ως backup για τα δεδομένα ρυθμίσεων και την κατάσταση του cluster.

- **kube-apiserver:** Είναι κομμάτι του control plane, το οποίο αποτελεί το frontend της πλατφόρμας εκτείθοντας τις λειτουργίες του Kubernetes API στους χρήστες.

- **kube-controller-manager:** Είναι υπεύθυνος για την εκτέλεση των controller διεργασιών. Οι διεργασίες αυτές περιλαμβάνουν έλεγχο και ανταπόκριση σε περίπτωση θανάτου ενός από τους Nodes, διαχείριση πολλαπλών αντιγράφων των Pods, δημιουργία λογαριασμών και API tokens καθώς και για την διαχείριση του Endpoints object.

- **kube-scheduler:** Είναι υπεύθυνος για την ανάθεση διαθέσιμων Nodes σε Pods τα οποία έχουν δημιουργηθεί έτσι ώστε να αρχίσει η εκτέλεση τους. Η χρονοδρομολόγηση των διεργασιών λαμβάνει υπόψιν διάφορους παράγοντες όπως απαιτήσεις σε πόρους, περιορισμούς υλικού και λογισμικού, τοπικότητα δεδομένων κτλ.

- **cloud-controller-manager:** Είναι υπεύθυνος για την σύνδεση του cluster με το API κάποιου cloud παροχέα καθώς και για την διαχείριση των controller οι οποίοι είναι ειδικοί για το cloud. Σε περίπτωση όπου η πλατφόρμα τρέχει τοπικά τότε δεν υπάρχει κάποιος cloud-controller-manager [34].

Οι worker nodes (Node servers) αποτελούν τους κόμβους οι οποίοι είναι υπεύθυνοι για την εκτέλεση των απομονωμένων σε containers εφαρμογών. Κάθε Node server εμπεριέχει ένα ή περισσότερα Pods τα οποία αποτελούνται από ένα ή περισσότερα containers. Τα containers αυτά λειτουργούν μαζί και μοιράζονται κοινή διεύθυνση IP, hostname, πόρους, κύκλο ζωής κτλ. για αυτό και μπορούν να αντιμετωπιστούν σαν ένα ενιαίο στιγμιότυπο μίας εφαρμογής.

Οι Node servers αποτελούνται από τα εξής κύρια μέρη:

- **kubelet:** Τρέχει σε κάθε Node του cluster εξασφαλίζοντας μέσω ενός συνόλου από PodSpecs, ότι τα containers τρέχουν σε ένα Pod και είναι υγιή. Επικοινωνεί με τον master με σκοπό την λήψη εντολών εργασιών. Αλληλοεπιδρά με το etcd διαβάζοντας το αρχείο manifest με σκοπό την λήψη πληροφοριών όσον αφορά τις ρυθμίσεις των containers.

- **kube-proxy:** Αποτελεί ένα network proxy το οποίο τρέχει σε κάθε node μέσα στο cluster διατηρώντας τους κανόνες του δικτύου σε κάθε κόμβο. Οι κανόνες αυτοί επιτρέπουν την επικοινωνία μεταξύ Pods μέσα ή έξω από το cluster. Χρησιμοποιεί το packet filtering layer του λειτουργικού συστήματος στο οποίο τρέχει και σε περίπτωση που αυτό δεν είναι διαθέσιμο, προωθεί το ίδιο τα πακέτα του δικτύου.

- **Container runtime:** Αποτελεί το λογισμικό το οποίο είναι υπεύθυνο για το τρέξιμο και διαχείριση του κύκλου ζωής των containers (πχ Docker).

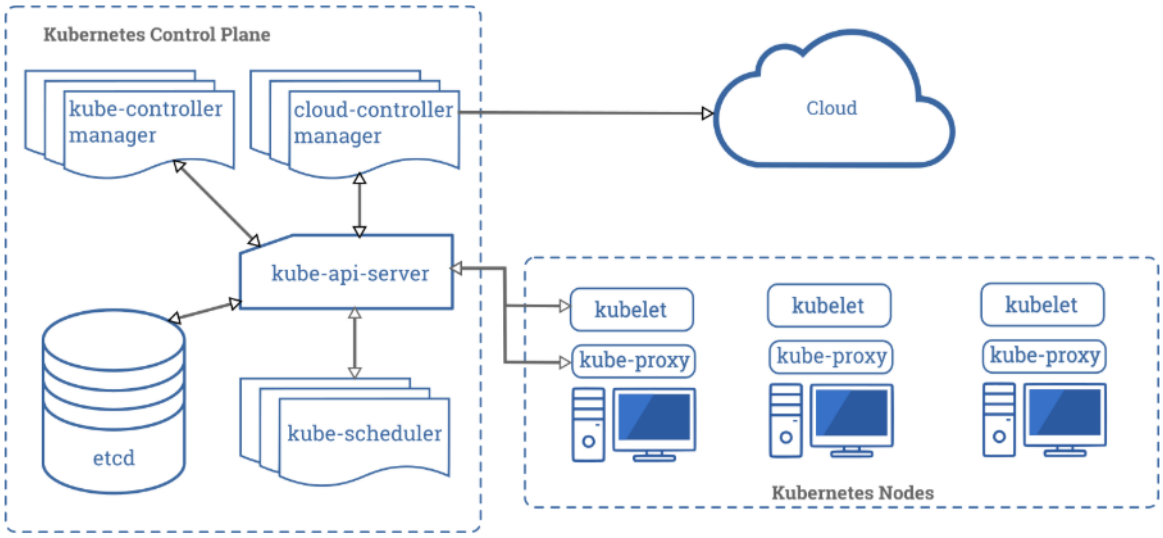
Πλεονεκτήματα

Μερικά από τα πλεονεκτήματα που προσφέρει η πλατφόρμα Kubernetes είναι η αξιοπιστία στην διαθεσιμότητα των εφαρμογών και υπηρεσιών αφού όταν ένα container σταματήσει να λειτουργεί, ένα άλλο αυτόματα παίρνει την θέση του χωρίς ο χρήστης να το προσέξει (self healing). Ακόμη προσφέρει μεγάλη δυνατότητα κλιμακωσιμότητας μέσα στο cluster μέσω της εύκολης προσθήκης νέων Pods σε έναν Node καθώς και άλλων Node μέσα στο cluster. Η τεχνολογία αυτή διαθέτει μια μεγάλη κοινότητα σε σύγκριση με άλλα εργαλεία ενορχήστρωσης η οποία προσφέρει σημαντική βοήθεια σε θέματα υποστήριξης, αναβάθμισης κτλ. Προσφέρει δυνατότητα υποστήριξης πολλών και διαφορετικών τύπων εφαρμογών χωρίς της ύπαρξη περιορισμών όπως επίσης και ισορρόπηση του φόρτου μέσω της ανάθεσης ατομικών IP διευθύνσεων για κάθε Pod και ενός DNS για κάθε σύνολο από

Pods. Επίσης προσφέρει μεγάλη ευελιξία στην διαχείριση των πόρων μέσω της δυνατότητας ρύθμισης μέγιστης και ελάχιστης τιμής κατά την ανάθεση πόρων όπως CPU, μνήμη κτλ. καθώς και δυνατότητα διαχείρισης και επίβλεψης πολλών ενεργών containers ταυτόχρονα από έναν μόνο διαχειριστή.

Μειονεκτήματα

Παρόλα τα πλεονεκτήματα που προσφέρει η τεχνολογία Kubernetes, παρουσιάζει κάποια μειονεκτήματα όσον αφορά την λειτουργία της. Η χρήση της τεχνολογίας αυτής για απλές εφαρμογές οι οποίες δεν είναι υψηλές σε απαιτήσεις πόρων, κρίνεται ως υπερβολή. Η τεχνολογία Kubernetes αποτελεί μια περίπλοκη τεχνολογία, ειδικά για νέους χρήστες και μηχανικούς, γεγονός το οποίο καθιστά δύσκολη, χρονοβόρα και ακριβή την διαδικασία μετάβασης από κάποια άλλη παρόμοιας μορφής τεχνολογία.



Σχήμα 1.17 Αρχιτεκτονική Kubernetes

Πηγή: *https://kubernetes.io/docs/concepts/overview/components/*

3.5.3 Apache Mesos

Η τεχνολογία Apache Mesos αποτελεί ένα σύστημα διαχείρισης cluster το οποίο χειρίζεται τον φόρτο εργασίας ενός κατανεμημένου περιβάλλοντος μέσω του δυναμικού καταμερισμού και απομόνωσης πόρων. Δημιουργήθηκε από το University of California, Berkeley και αποτελεί λογισμικό ανοικτού κώδικα. Η τεχνολογία Apache Mesos είναι κατάλληλη για ανάπτυξη και διαχείριση εφαρμογών μεγάλης κλίμακας σε κατανεμημένα περιβάλλοντα όπως spark, Hadoop κτλ. οι οποίες απαιτούν κατανεμημένους πόρους. Βρίσκεται μεταξύ application layer και λειτουργικού συστήματος έχοντας αντίθετο ρόλο σε σχέση με την διαδικασία της εικονικοποίησης αφού συγκεντρώνει πλήθος φυσικών πόρων σε μια μεγάλη εικονική μηχανή με πληθώρα πόρων, αφαιρώντας έτσι την ανάγκη για δέσμευση διάφορων υπολογιστικών μηχανημάτων για συγκεκριμένες λειτουργίες (static partitioning). Έχει την δυνατότητα απομόνωσης πόρων όπως CPU, memory, file system κτλ. οι οποίοι δεν αλληλοεπιδρούν μεταξύ τους, επιτρέποντας έτσι την εύκολη κατανομή των ελεύθερων πόρων σε διάφορες υπο εκτέλεση εργασίες μέσα στο cluster. Χρησιμοποιείται από πολλές μεγάλες εταιρείες όπως Twitter, Airbnb, Uber, Netflix κτλ λόγω της υποστήριξης που προσφέρει όσον αφορά θέματα microservices, big data, real time analytics και elastic scaling.

Λειτουργία

Η αρχιτεκτονική λειτουργίας της τεχνολογίας Apache Mesos αποτελείται από τα εξής κύρια μέρη:

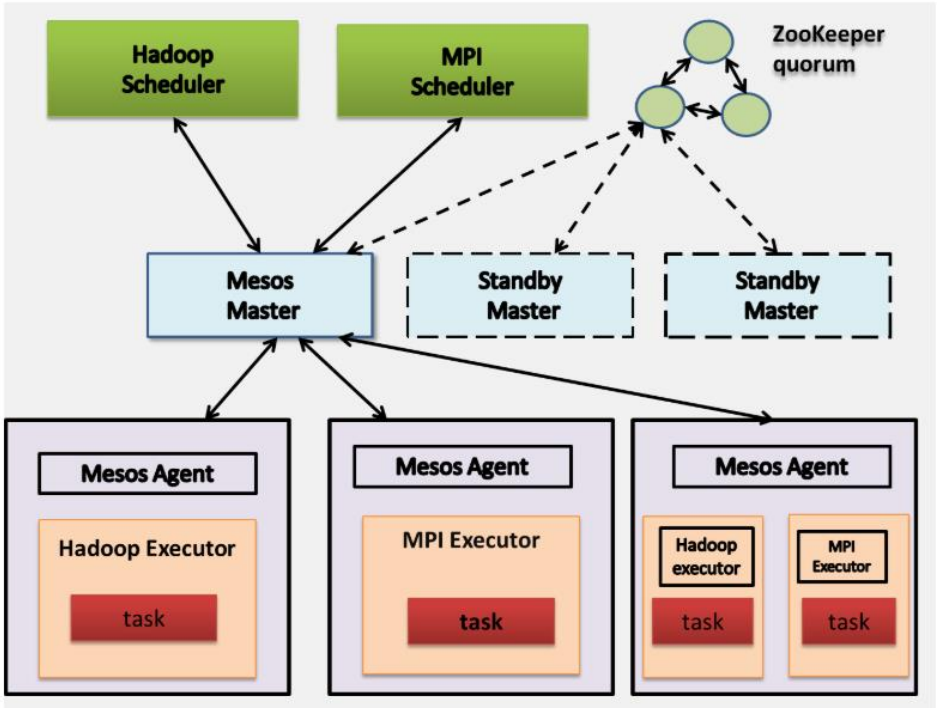
- **Mesos Master:** Αποτελεί τον πυρήνα του cluster όπου μέσω μιας διεπαφής παρέχει πληροφορίες όσον αφορά τους διαθέσιμους πόρους μέσα στο cluster. Εξασφαλίζει υψηλή διαθεσιμότητα του cluster μέσω της συνεργασίας με την τεχνολογία ZooKeeper η οποία ενεργοποιεί ένα νέο Master σε περίπτωση σφάλματος του τρέχοντος. Ακόμη ο Mesos Master αποθηκεύει όλα τα απαραίτητα δεδομένα που σχετίζονται με τις τρέχουσες εργασίες. Προσφέρει τους διαθέσιμους πόρους στα Frameworks μέσω του DRF (Dominant Resource Fairness) αλγορίθμου.

- **Mesos Agent:** Κάθε μηχάνημα μέσα στο cluster τρέχει κάποιον Agent ο οποίος διαχειρίζεται το container το οποίο περιέχει τον τοπικό executor. Αποτελεί τον μεσάζων παράγοντα μεταξύ του Master και του τοπικού executor λαμβάνοντας τις προς εκτέλεση εργασίες και ενημερώνοντας τους schedulers για την κατάσταση των εκτελέσιμων εργασιών. Ακόμη παρέχει πληροφορίες όσον αφορά τον host στον οποίο τρέχουν, τις εργασίες και τους executors, τους διαθέσιμους πόρους και άλλα μεταδεδομένα.

- **Mesos Framework:** Αποτελούνται από 2 μέρη τον Scheduler και τον Executor. Ο Scheduler είναι υπεύθυνος για την έναρξη της διαδικασίας εκτέλεσης μιας εργασίας εφόσον οι απαιτήσεις σε πόρους και άλλοι περιορισμοί έχουν ικανοποιηθεί από το Master. Ακόμη είναι υπεύθυνος για την διαχείριση σφαλμάτων κατά την εκτέλεση των εργασιών. Ο Executor εκτελεί την εργασία την οποία έχει δρομολογήσει ο Scheduler και ενημερώνει πίσω για την κατάσταση κάθε εργασίας που του έχει ανατεθεί. [35]

Κάθε Agent περιέχει κάποιον Executor. Μερικά παραδείγματα από Frameworks είναι Chronos, Marathon, Aurora, Hadoop, Spark, Jenkins κτλ.

Η γενική λειτουργία της τεχνολογίας Apache Mesos αποτελείται από τις εξής ενέργειες. Αρχικά όταν κάποιος Mesos Executor ελευθερώσει κάποιον πόρο, γίνεται ενημέρωση του Mesos Master επί του γεγονότος αυτού. Ο Mesos Master μέσω της πολιτικής η οποία αφορά την κατανομή πόρων εξακριβώνει τον αριθμό διαθέσιμων πόρων με σκοπό την προσφορά τους στα αντίστοιχα Frameworks. Έτσι ο Mesos Master στέλνει διάφορες προσφορές (offers) στα Frameworks, οι οποίες αποτελούν απεικονίσεις των διαθέσιμων πόρων. Αν οι προσφορές αυτές ικανοποιούν τις απαιτήσεις του Framework τότε αυτό στέλνει πίσω στον Master μια λίστα από τις εργασίες τις οποίες επιθυμεί να εκτελέσει χρησιμοποιώντας τους διαθέσιμους αυτούς πόρους. Έπειτα ο Mesos Master στέλνει τις εργασίες προς εκτέλεση στους Executors, οι οποίοι δεσμεύουν τους απαραίτητους πόρους που χρειάζονται για την εκτέλεση των εργασιών αυτών. Τέλος γίνεται έναρξη της διαδικασίας εκτέλεσης των επιθυμητών εργασιών από τα Frameworks.



Σχήμα 1.18 Αρχιτεκτονική Apache Mesos

Πηγή: *https://data-flair.training/blogs/apache-mesos-tutorial/*

Πλεονεκτήματα

Μερικά από τα πλεονεκτήματα που προσφέρει η χρήση της τεχνολογίας Apache Mesos ως μέσο διαχείρισης ενός cluster είναι η δυνατότητα κλιμακωσιμότητας σε μεγάλο βαθμό αφού μπορούν εύκολα να ενσωματωθούν νέοι κόμβοι στο σύστημα, επεκτείνοντας έτσι τον διαθέσιμο αριθμό πόρων σε ένα ανεκτικό σε σφάλματα σύστημα. Ακόμη μέσω της ομαδοποίησης των πόρων σε ένα κοινό σύνολο, απλοποιείται σε μεγάλο βαθμό η διαδικασία κατανομής τους στις διάφορες εργασίες οι οποίες πρέπει να εκτελεστούν. Επιπρόσθετα η χρήση της τεχνολογίας Apache Mesos επιτρέπει την συνύπαρξη και εκτέλεση διάφορων ειδών ανεξάρτητων διεργασιών μέσα στο cluster, όπως microservices, analytics, κατανεμημένες εργασίες ή κοινότυπες εφαρμογές κτλ. διευρύνοντας έτσι το φάσμα χρησιμοποίησης του συστήματος, αυξάνοντας έτσι τον βαθμό χρησιμοποίησης μειώνοντας παράλληλα το κόστος. Επίσης το γεγονός ότι η τεχνολογία Apache Mesos υποστηρίζει πληθώρα από Frameworks, την καθιστά μια ευέλικτη λύση, η οποία μπορεί να συνδυαστεί με πολλές σύγχρονες άλλες τεχνολογίες.

Μειονεκτήματα

Παρά την πληθώρα πλεονεκτημάτων που προσφέρει η χρήση της τεχνολογίας Apache Mesos, υπάρχουν μερικά μειονεκτήματα τα οποία αντιμετωπίζει όπως η έλλειψη ενός στοιχείου διαχείρισης δικτύου. Ακόμη η χρήση της τεχνολογίας Apache Mesos σε clusters τα οποία αποτελούνται από μικρό αριθμό κόμβων, μπορεί να αποτελέσει μια πολύπλοκη λύση για το σύστημα αυτό.

4. Υλοποίηση αρχιτεκτονικής επεξεργασίας δεδομένων μεγάλης κλίμακας

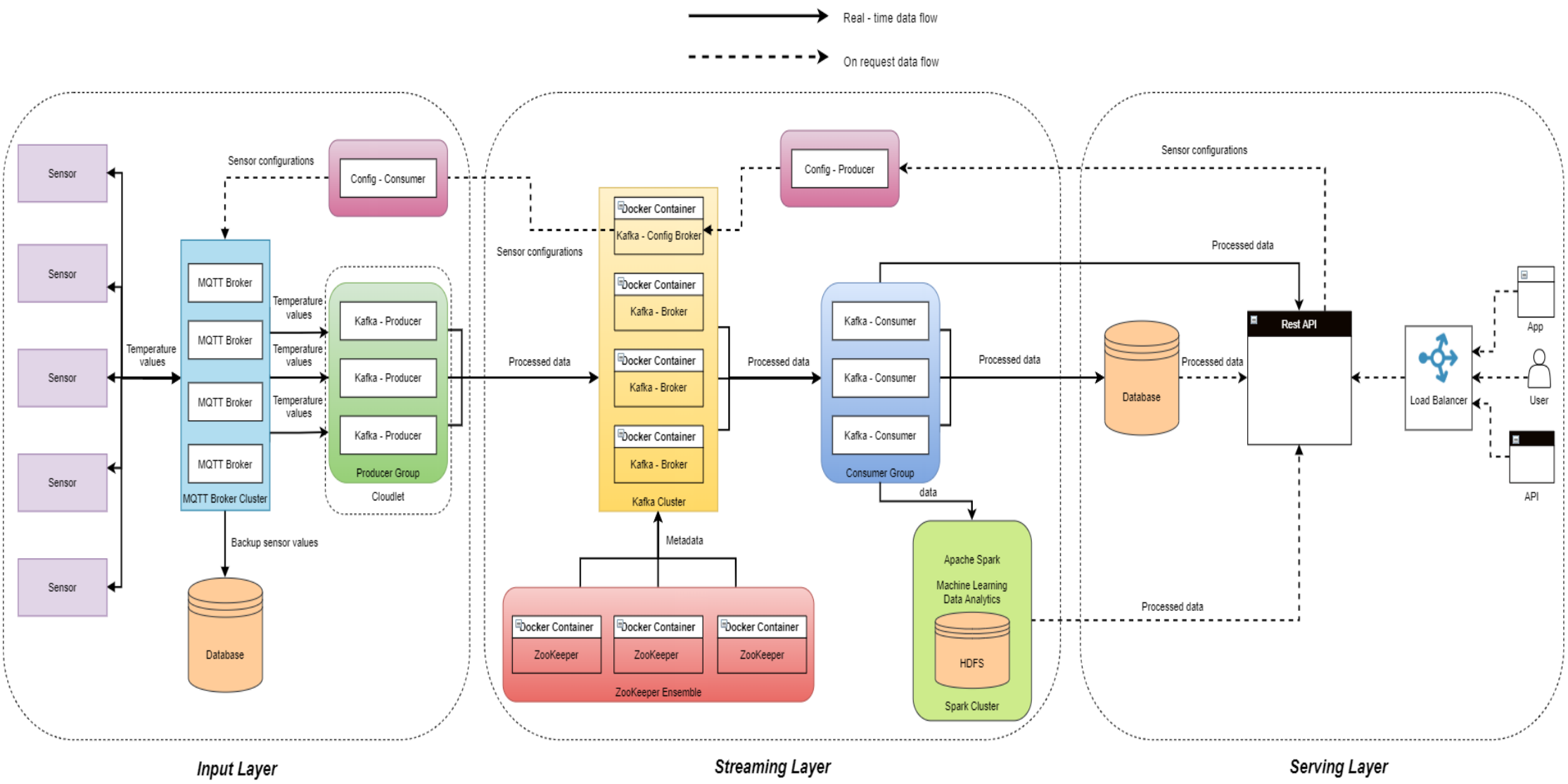
4.1 Σενάριο χρήσης

Στην ενότητα αυτή θα γίνει ανάλυση της υλοποίησης μιας μικρής κλίμακας αρχιτεκτονικής επεξεργασίας δεδομένων, μέσω της οποίας μπορούν να εξυπηρετηθούν διάφορα σενάρια χρήσης ενός σύγχρονου IoT συστήματος όπως monitoring, λήψη αποφάσεων βάση στατιστικών, πρόβλεψη σφαλμάτων και βλαβών κτλ. Συγκεκριμένα στην περίπτωση αυτή το σενάριο χρήσης αφορά την λήψη συνεχούς ροών δεδομένων από διάφορους αισθητήρες, με σκοπό την εφαρμογή διαφόρων συναρτήσεων επεξεργασίας (aggregate, sum, machine learning, clustering κτλ), αποθήκευση και διάθεση τους στον τελικό χρήστη τόσο σε πραγματικό χρόνο όσο και κατά απαίτηση μέσω μίας διεπαφής. Τα δεδομένα αυτά μπορεί να αφορούν διάφορες τιμές θερμοκρασιών από βαρύ βιομηχανικό εξοπλισμό, κτηριακές εγκαταστάσεις, θερμοκήπια και γεωργικές καλλιέργειες, datacentres, υπολογιστικά συστήματα κτλ.

4.2 Σχεδιασμός και ανάλυση αρχιτεκτονικής

Η IoT αρχιτεκτονική επεξεργασίας δεδομένων μεγάλης κλίμακας η οποία έχει υλοποιηθεί στην παρακάτω ενότητα, αποτελεί έναν συνδυασμό cloud και edge computing τεχνολογιών μέσω των οποίων υλοποιείται ένα πλήρως λειτουργικό και σύγχρονο IoT σύστημα. Συγκεκριμένα γίνεται υλοποίηση μιας αρχιτεκτονικής επεξεργασίας δεδομένων κ (Kappa Architecture) μικρής κλίμακας, με στοιχεία edge computing. Ο σχεδιασμός της αρχιτεκτονικής αυτής υλοποιεί 3 επίπεδα, τα Input Layer, Streaming Layer και Serving Layer μέσω των οποίων δίνεται η δυνατότητα μεταφοράς, επεξεργασίας και αποθήκευσης δεδομένων μεγάλης κλίμακας σε πραγματικό χρόνο. Κάθε επίπεδο της αρχιτεκτονικής αποτελείται από συνδυασμούς μερικών από τις τεχνικές και τεχνολογίες μετάδοσης δεδομένων - πρωτοκόλλων επικοινωνίας μεταξύ των συσκευών, αποθήκευσης δεδομένων μεγάλης κλίμακας, streaming και επεξεργασίας δεδομένων σε πραγματικό χρόνο, διαμοιρασμού πόρων και παράλληλης επεξεργασίας δεδομένων όπως επίσης και τεχνικές ενορχήστρωσης, συντονισμού συστήματος και απομόνωσης εφαρμογών οι οποίες αναλύθηκαν στις προηγούμενες ενότητες. Στην παρακάτω ενότητα θα γίνει μια εκτενής ανάλυση του σχεδιασμού και λειτουργίας του κάθε επιπέδου της αρχιτεκτονικής, παρουσιάζοντας τις τεχνικές και τεχνολογίες που έχουν χρησιμοποιηθεί για την υλοποίηση των επιπέδων αυτών.

4.2.1 Διάγραμμα Λειτουργίας Αρχιτεκτονικής



4.2.2 Input Layer

Η λειτουργία του επιπέδου Input Layer αφορά την συνεχή λήψη των δεδομένων από τους διάφορους αισθητήρες και την διάθεση τους στο επόμενο επίπεδο της αρχιτεκτονικής για περαιτέρω επεξεργασία. Αποτελεί το πρώτο στάδιο του κύκλου ζωής των δεδομένων, μέσα στο οποίο γίνεται η κατάλληλη εξόρυξη, προετοιμασία και προεπεξεργασία των δεδομένων σε πραγματικό χρόνο. Η διαδικασία αυτή θα διευκολύνει την μεταγενέστερη διαδικασία επεξεργασίας των μετέπειτα επιπέδων, αφαιρώντας σημαντικό φόρτο εργασίας ο οποίος τώρα εκτελείται σε τοπικό επίπεδο, όπως προϋποθέτει η ιδέα της αρχιτεκτονικής edge computing.

Τα κύρια μέρη της αρχιτεκτονικής του Input Layer είναι:

- **Αισθητήρες:** Αποτελούν τις κύριες πηγές δεδομένων οι οποίες είναι ρυθμισμένες να μεταδίδουν τιμές θερμοκρασιών ανά κάποιο χρονικό διάστημα το οποίο έχει προσδιοριστεί. Χρησιμοποιούν το πρωτόκολλο επικοινωνίας και μετάδοσης δεδομένων MQTT το οποίο αποτελεί μια αξιόπιστη και ελαφριά λύση, για την μετάδοση δεδομένων μεταξύ συσκευών. Θεωρούμε ότι το σύστημα μας λαμβάνει τιμές θερμοκρασιών από έναν μεγάλο αριθμό αισθητήρων οι οποίοι θα είναι ομαδοποιημένοι ανά περιοχή/μηχάνημα/δωμάτιο/όροφο. Το πρωτόκολλο MQTT το οποίο χρησιμοποιείται για την μετάδοση των δεδομένων βασίζεται στο πρωτόκολλο επικοινωνίας TCP, το οποίο δεν χρησιμοποιεί κρυπτογραφημένη επικοινωνία από προεπιλογή. Για τον λόγο αυτό θα χρησιμοποιηθεί κάποιο TLS security για την αποφυγή κακόβουλων επιθέσεων και εξασφάλιση περεταίρω προστασίας κατά την μετάδοση των δεδομένων.

- **MQTT Broker cluster:** Αποτελεί τον μεσάζοντα παράγοντα μεταξύ των αισθητήρων και των Kafka Producer. Είναι υπεύθυνος για την συνεχή λήψη των δεδομένων από τους αισθητήρες και διάθεση τους στους Kafka Producer. Αυτό θα επιτευχθεί μέσω της διαδικασίας κοινοποίησης των δεδομένων από τους αισθητήρες με την βοήθεια του πρωτοκόλλου επικοινωνίας MQTT, σε διάφορα topics τα οποία δημιουργούνται στον broker. Έτσι οι Kafka Producers έχουν την δυνατότητα να εγγραφούν στα κατάλληλα topics, μέσω των οποίων θα λαμβάνουν τις τιμές των θερμοκρασιών μόλις αυτές κοινοποιηθούν στο αντίστοιχο topic. Ο MQTT broker μπορεί να ρυθμιστεί ανάλογα έτσι ώστε να διατηρεί μέρος των μεταδιδόμενων δεδομένων για κάποιο χρονικό διάστημα. Ακόμη δίνεται η δυνατότητα ρύθμισης του qos (Quality of Service), εξασφαλίζοντας μετάδοση των δεδομένων ακριβώς μια, τουλάχιστον μια ή το πολύ μια φορά. Όπως έχει προαναφερθεί για την κρυπτογράφηση της μετάδοσης των δεδομένων θα χρησιμοποιηθεί κάποιο TLS security. Η χρήση των broker αυτών προσφέρει μεγάλες δυνατότητες κλιμακωσιμότητας στο σύστημα, μέσω της χρήσης του κατανεμημένου cluster από MQTT brokers στους οποίους οι αισθητήρες θα ανατίθενται ανά ομάδες, ανάλογα με διάφορες παραμέτρους διαχωρισμού όπως για παράδειγμα η γεωγραφική τοποθεσία, ενισχύοντας έτσι το επίπεδο παραλληλισμού κατά την μετάδοση των δεδομένων.

- **Βάση δεδομένων:** Χρησιμοποιείται μια γρήγορη και αξιόπιστη βάση δεδομένων η οποία θα λειτουργήσει ως μέσο ασφαλείας σε περίπτωση σφάλματος κατά την μετάδοση των δεδομένων από τον Mosquitto Broker στους Kafka Producers. Έτσι σε περίπτωση ύπαρξης κάποιου σφάλματος κατά την μετάδοση των δεδομένων, οι τιμές οι οποίες δεν κατάφεραν να μεταδοθούν θα υπάρχουν μέσα στην βάση δεδομένων ως backup, προσφέροντας δυνατότητα αναμετάδοσης όταν το σφάλμα διορθωθεί. Προτείνεται μια ελαφριά και γρήγορη NoSQL βάση δεδομένων για την γρήγορη ανάκτηση των δεδομένων, έτσι ώστε να μην προκληθεί καθυστέρηση στην αναμετάδοση των δεδομένων.

- **Configuration Consumer:** Αποτελεί μέρος μιας αντίστροφης ροής μέσα στην αρχιτεκτονική αυτή, η οποία έχει ως σκοπό λειτουργίας την διαχείριση των ρυθμίσεων των αισθητήρων μέσω αιτημάτων του χρήστη στο REST API του Serving Layer. Ο Configuration consumer θα είναι εγγεγραμμένος σε κάποιο ειδικό configuration topic ενός συγκεκριμένου Kafka broker, στο οποίο θα κοινοποιούνται δεδομένα τα οποία θα αφορούν εφαρμογή αλλαγών στις ρυθμίσεις των αισθητήρων, όπως για παράδειγμα αλλαγή στον ρυθμό μετάδοσης, απενεργοποίηση αισθητήρα, αλλαγή topic κτλ. Με την σειρά του θα κοινοποιεί σε κάποιο αντίστοιχο configuration topic στο Mosquitto broker έτσι ώστε ο κατάλληλος αισθητήρας να λάβει το μήνυμα που πρέπει και να εκτελέσει τις αλλαγές στις ρυθμίσεις του.

- **Kafka Producers:** Αποτελούν τον συνδετικό κρίκο με το επόμενο επίπεδο επεξεργασίας, το Streaming Layer. Εδώ θα γίνει όλη προεπεξεργασία που χρειάζεται για να προχωρήσουν τα δεδομένα στο επόμενο στάδιο του κύκλου ζωής τους. Οι Kafka producers είναι εγγεγραμμένοι στα ανάλογα topics των MQTT broker λαμβάνοντας έτσι κάθε τιμή θερμοκρασίας που παράγουν οι αισθητήρες σε πραγματικό χρόνο μέσω του πρωτοκόλλου επικοινωνίας MQTT. Έτσι έχουμε ένα σύστημα παράλληλης κατανάλωσης δεδομένων από τους αισθητήρες σε πραγματικό χρόνο. Εφόσον κάποιος Kafka producer λάβει μια τιμή θερμοκρασίας, μπορεί σε πραγματικό χρόνο να εφαρμόσει σε αυτήν μερική επεξεργασία και να την προωθήσει στο επόμενο στάδιο επεξεργασίας. Η μερική επεξεργασία μπορεί να περιλαμβάνει πρόσθεση επιπλέον χρήσιμων δεδομένα όπως χρονοσφραγίδες και άλλα πεδία με πληροφορίες του αισθητήρα καθώς και εφαρμογή διάφορων φίλτρων ή μετασχηματισμών στα δεδομένα. Έτσι η παράλληλη αυτή προεπεξεργασία προετοιμάζει τα δεδομένα για το επόμενο επίπεδο εφαρμόζοντας στοιχεία της αρχιτεκτονικής edge computing με σκοπό την μείωση του φόρτου εργασίας του επομένου επιπέδου όπως επίσης και την επίτευξη συνέπειας και καθολικότητας στα δεδομένα. Αφού ολοκληρωθεί η κατάλληλη προεπεξεργασία, τα δεδομένα προωθούνται από τους Kafka producers στο επόμενο επίπεδο της αρχιτεκτονικής, μέσω της κοινοποίησης τους στα αντίστοιχα topics των Kafka broker του cluster. Οι Kafka Producers χρησιμοποιούν την τεχνολογία των cloudlet μεταφέροντας υπολογιστικό φόρτο κοντά στις πηγές παραγωγής των δεδομένων, μειώνοντας έτσι το χρόνο απόκρισης δίνοντας παράλληλα την δυνατότητα λήψης σε τοπικό επίπεδο και όχι από το cloud, των επεξεργασμένων τιμών θερμοκρασίας.

4.2.3 Streaming Layer

Η αρχιτεκτονική λειτουργίας του Streaming Layer επιπέδου αφορά την κύρια επεξεργασία η οποία θα εφαρμοστεί πάνω στα δεδομένα μας. Στο επίπεδο αυτό θα εφαρμοστεί πιο βαριά και εξειδικευμένη επεξεργασία σε πραγματικό χρόνο, όπως εφαρμογή διάφορων αλγορίθμων μηχανικής μάθησης, φιλτράρισμα, aggregators κτλ. μέσω ενός κατανεμημένου συστήματος επεξεργασίας, έτσι ώστε να ικανοποιούνται να ικανοποιούνται όλα τα ερωτήματα των χρηστών όπως επίσης και οι απαιτήσεις των εφαρμογών οι οποίες θα κάνουν χρήση του REST API που προσφέρεται από το Serving Layer. Ακόμη στο επίπεδο αυτό γίνεται μακροχρόνια αποθήκευση των δεδομένων σε κάποιο κατανεμημένο σύστημα αποθήκευσης για γρήγορη, αποδοτική και παράλληλη προσπέλαση και διαχείριση των δεδομένων. Το επίπεδο αυτό, λόγω της βαριάς και εξιδεικευμένης επεξεργασίας που εκτελεί, υλοποιείται σε κάποιο κεντρικό σύστημα επεξεργασίας στο cloud.

Τα κύρια μέρη της αρχιτεκτονικής του Streaming Layer είναι:

- **Kafka cluster:** Το cluster αυτό θα αποτελείται από πολλούς κατανεμημένους Kafka brokers στους οποίους θα είναι εγγεγραμμένοι οι producers και consumers. Κάθε broker στο cluster θα περιέχει διάφορα topics στα οποία οι producers θα κοινοποιούν δεδομένα σε πραγματικό χρόνο, έτσι ώστε οι consumers να μπορούν να καταναλώσουν τα δεδομένα αυτά για περαιτέρω επεξεργασία. Οι Kafka brokers θα μπορούν να ρυθμιστούν έτσι ώστε να διατηρούν τα εισερχόμενα δεδομένα για καθορισμένο χρονικό διάστημα δίνοντας έτσι την δυνατότητα επανεπεξεργασίας παλαιότερων ροών δεδομένων σε περίπτωση που υπάρξουν αλλαγές τον κώδικα επεξεργασίας των δεδομένων. Ακόμη η χρήστη του Kafka cluster επιτρέπει την δημιουργία αντιγράφων των topics τα οποία διαμοιράζονται στους broker του cluster, εξασφαλίζοντας έτσι συνεχή διαθεσιμότητα των δεδομένων μετά από σφάλμα κάποιου εκ των Kafka broker. Εκτός από τους broker οι οποίοι διαχειρίζονται τα topics των δεδομένων, θα υπάρχει ειδικός broker ο οποίος θα εξυπηρετεί μέσω κατάλληλων topic, αιτήματα τα οποία αφορούν αλλαγές στις ρυθμίσεις των αισθητήρων. Η χρήση του Kafka cluster εκτός από την συνεχή διαθεσιμότητα των δεδομένων και την αντιμετώπιση σφαλμάτων, προσφέρει παραλληλισμός ως προς την συνεχή κοινοποίηση και κατανάλωση δεδομένων ενισχύοντας έτσι σε μεγάλο βαθμό την απόδοση του συστήματος.

- **Kafka Consumers:** Οι Kafka consumers είναι εγγεγραμμένοι στα κατάλληλα topic των Kafka broker, για την συνεχή λήψη των δεδομένων από τους producers και προώθηση τους στο επόμενο στάδιο επεξεργασίας. Κάθε φορά που λαμβάνουν κάποια τιμή εφαρμόζουν μερική επεξεργασία σε αυτή και την προωθούν σε 3 διαφορετικές κατευθύνσεις στην αρχιτεκτονική. Αφού μια τιμή ληφθεί από κάποιον Kafka consumer, αυτή θα μπορεί να διατεθεί στον χρήστη μέσω της διαδικτυακής επαφής σε πραγματικό χρόνο, να αποθηκευτεί στην βάση δεδομένων η οποία βρίσκεται στο Serving Layer με σκοπό την ασύγχρονη προσπέλαση της μετά από απαίτηση του χρήστη και να προωθεί στο Spark cluster όπου θα γίνει εφαρμογή των απαιτητικών αλγορίθμων ανάλυσης δεδομένων και μηχανικής μάθησης.

- **ZooKeeper ensemble:** Η τεχνολογία ZooKeeper αποτελεί ένα απαραίτητο στοιχείο για την λειτουργία και διαχείριση του Kafka cluster. Η χρήση του ZooKeeper έχει ως σκοπό την διαχείριση θεμάτων όπως η εκλογή νέου controller σε περίπτωση σφάλματος του παλιού. Ο controller αποτελεί την οντότητα η οποία είναι υπεύθυνη για την ανάθεση νέων αρχηγών σε partitions, των οποίων οι αρχηγοί ανήκαν σε κάποιο κόμβο ο οποίος έχει τερματίσει την λειτουργία του, χρησιμοποιώντας τα αντίγραφα αυτών των partitions. Ακόμη η τεχνολογία ZooKeeper διαχειρίζεται τους ενεργούς broker του cluster, τα υπάρχοντα topics (αριθμός partitions, τοποθεσία αντιγράφων, leaders κτλ), την ποσότητα δεδομένων που μπορεί κάποιος client να διαβάσει και να γράψει (quotas), καθώς και τις Access Control Lists οι οποίες ορίζουν ποιος μπορεί να διαβάσει/γράψει και σε ποιο topic, ποια consumer groups υπάρχουν, offsets κτλ.

- **Configuration Producer:** Αποτελεί μέρος της αντίστροφής ροής δεδομένων στην αρχιτεκτονική η οποία αφορά την διαχείριση των ρυθμίσεων και σφαλμάτων που παρουσιάζουν οι αισθητήρες. Ο Configuration producer μετά από αίτημα του χρήστη στο REST API μέσω της διαδικτυακής διεπαφής, κοινοποιεί στο ειδικό topic του Kafka broker ο οποίος είναι υπεύθυνος για την προώθηση δεδομένων που αφορούν αλλαγές στις ρυθμίσεις των αισθητήρων όπως αλλαγή στον ρυθμό μετάδοσης, απενεργοποίηση αισθητήρα, αλλαγή topic κτλ. Έτσι ο αρμόδιος Configuration consumer θα μπορέσει να λάβει το μήνυμα αλλαγής ρυθμίσεων, προωθώντας το μέσω του Mosquitto broker στον κατάλληλο αισθητήρα, ο οποίος θα προβεί σε αλλαγή των ρυθμίσεων του.

- **Spark Cluster**: Η χρήση του Spark cluster στο επίπεδο αυτό έχει ως σκοπό την υποστήριξη εκτέλεσης διάφορων απαιτητικών αλγορίθμων επεξεργασίας και ανάλυσης δεδομένων όπως επίσης και αλγορίθμους μηχανικής μάθησης. Το κατανεμημένο αυτό σύστημα επεξεργασίας δεδομένων μπορεί να προσφέρει πολύ μεγάλη αύξηση στην συνολική απόδοση της αρχιτεκτονικής λόγω της αποθήκευσης των δεδομένων στην μνήμη (cache) κατά την διάρκεια της επεξεργασίας τους, προσφέροντας υψηλή ταχύτητα επεξεργασίας σε μεγάλο όγκο δεδομένων. Η δυνατότητα υποστήριξης διάφορων λειτουργιών όπως streaming, τεχνικές μηχανικής μάθησης, επεξεργασία γράφων κτλ. καθιστά το Apache Spark μια εξαιρετική λύση για τις ανάγκες μιας πιο εξειδικευμένης επεξεργασίας και ανάλυσης των δεδομένων. Τα δεδομένα θα προωθούνται από τους Kafka consumers στο HDFS το οποίο θα είναι συνδεδεμένο με το Spark Cluster. Έτσι το Spark θα μπορεί να λαμβάνει με αποδοτικό τρόπο τα δεδομένα από το κατανεμημένο σύστημα αποθήκευσης για την εκτέλεση των επιθυμητών λειτουργιών επεξεργασίας και ανάλυσης, αξιοποιώντας τις δυνατότητες κατανεμημένης επεξεργασίας που προσφέρει η master-worker αρχιτεκτονική της τεχνολογίας Apache Spark. Για την διαχείριση, κατανομή πόρων και scheduling των εργασιών στο Spark cluster, χρησιμοποιείται η τεχνολογία διαχείρισης cluster Apache Mesos. Ο χρήστης μέσω των απαραίτητων κλήσεων στο REST API που προσφέρεται από το Serving Layer, θα μπορεί να προσπελάσει δεδομένα τα οποία έχουν υποστεί εξειδικευμένη επεξεργασία. Τα δεδομένα αυτά θα αποστέλνονται από το HDFS στην διαδικτυακή διεπαφή για παρουσίαση στον χρήστη.

4.2.4 Serving Layer

Το Serving Layer επίπεδο είναι υπεύθυνο για την παρουσίαση των επεξεργασμένων και μη δεδομένων στον χρήστη. Μέσω μιας διαδικτυακής διεπαφής ή κατευθείαν από το backend του συστήματος, ο χρήστης θα μπορεί να εφαρμόσει ερωτήματα στο σύστημα χρησιμοποιώντας ένα REST API για την ανάκτηση, ανάλυση και παρουσίαση των επιθυμητών δεδομένων.

Η αρχιτεκτονική του Serving Layer επιπέδου αποτελείται από τα εξής κύρια μέρη:

- **REST API:** Ο χρήστης μέσω μιας διαδικτυακής διεπαφής θα έχει την δυνατότητα να εφαρμόσει διάφορα ερωτήματα στο σύστημα. Αυτό γίνεται μέσω των κλήσεων σε διάφορα endpoints του API τα οποία εξυπηρετούν διάφορα αιτήματα όσον αφορά την παρουσίαση και ανάλυση των δεδομένων από τους αισθητήρες. Η διαδικτυακή διεπαφή, μέσω του REST API θα παρέχει παρουσίαση των δεδομένων με διάφορους τρόπους ανάλογα με τις απαιτήσεις του χρήστη. Αρχικά παρέχονται διάφορες γραφικές παραστάσεις οι οποίες παρουσιάζουν μεταβολές στις τιμές των αισθητήρων, οι οποίες ενημερώνονται σε πραγματικό χρόνο κατά την μετάδοση νέων τιμών από τους αισθητήρες. Ακόμη παρέχεται η μέγιστη και ελάχιστη τιμή κάθε αισθητήρα για την μελέτη ακραίων περιπτώσεων, τιμές οι οποίες ενημερώνονται σε πραγματικό. Κάθε χρήστης μπορεί μέσω ασύγχρονων κλήσεων στο API, να ανακτήσει δεδομένα από την βάση δεδομένων, όπως η τιμή της θερμοκρασίας, πληροφορίες για τον αισθητήρα, κάποιο timestamp κτλ. Για τους χρήστες οι οποίοι επιθυμούν μια πιο εξειδικευμένη ανάλυση και επεξεργασία των δεδομένων δίνεται η δυνατότητα εφαρμογής διάφορων αλγορίθμων ανάλυσης και επεξεργασίας, μηχανικής μάθησης κτλ. στα δεδομένα. Αυτό επιτυγχάνεται μέσω κατάλληλων κλήσεων στο σύστημα για την χρησιμοποίηση του ενσωματωμένου Spark cluster στο Streaming Layer, στο οποίο εκτελείται με κατανεμημένο τρόπο η πιο βαριά επεξεργασία των δεδομένων τα οποία σερβίρονται στην διεπαφή από το HDFS. Επιπρόσθετα κάποιος χρήστης μπορεί μέσω κατάλληλων κλήσεων στο API να διαχειριστεί τις ρυθμίσεις των αισθητήρων από μακριά αλλάζοντας διάφορες παραμέτρους όπως ο ρυθμός μετάδοσης, εκκίνηση, τερματισμός κτλ. Η λήψη των δεδομένων σε πραγματικό χρόνο επιτυγχάνεται μέσω της χρήσης web sockets, για την επικοινωνία του client με το backend, ενώ η λήψη των ασύγχρονων δεδομένων από τις βάσεις όπως επίσης και η αποστολή των αιτημάτων από τον client στο backend, επιτυγχάνεται μέσω της χρήσης HTTP αιτημάτων.

- **Βάση δεδομένων:** Αποτελεί τον αποθηκευτικό χώρο των δεδομένων που αφορούν τις τιμές θερμοκρασιών και άλλων στοιχείων των αισθητήρων. Στην βάση αυτή αποθηκεύονται αυτούσια τα δεδομένα (τιμή, αισθητήρας, χρονοσφραγίδα κτλ) χωρίς ιδιαίτερη επεξεργασία όπως επίσης και διάφορα στοιχεία που αφορούν τους λογαριασμούς των χρηστών. Χρησιμοποιείται μια βάση όπως για παράδειγμα μια MongoDB ή μια Redis, η οποία θα επιτρέπει γρήγορη προσπέλαση στα στοιχεία τα οποία θα διατίθενται στην διεπαφή μέσω ασύγχρονων κλήσεων στο σύστημα. Τα δεδομένα τα οποία τυγχάνουν πιο εξειδικευμένης και βαριάς επεξεργασίας, θα

διατίθενται στην διεπαφή μέσω του HDFS του Spark cluster, απ' όπου και θα εκτελούνται οι απαιτητικοί αλγόριθμοι ανάλυσης δεδομένων.

- **Load Balancer:** Ο Load Balancer αποτελεί λογισμικό ή υλικό το οποίο είναι υπεύθυνο για την εξισορρόπηση του φόρτου στο δίκτυο μέσω την ισορροπημένης κατανομής του φόρτου αυτού στους διάφορους server του backend. Δρομολογεί τα αιτήματα των χρηστών με τρόπο τέτοιο ώστε να μεγιστοποιείται η ταχύτητα εξυπηρέτησης, καθώς και ο βαθμός χρησιμοποίησης του συστήματος μειώνοντας τους χρόνους απόκρισης, χωρίς να υπερφορτώνεται κάποιος server. Ακόμη εξασφαλίζει ανακατεύθυνση των αιτημάτων σε περίπτωση που κάποιος server τερματιστεί ή αντιμετωπίσει κάποιο πρόβλημα κατά την λειτουργία του ενισχύοντας έτσι την διαθεσιμότητα του συστήματος. Η χρήση του Load Balancer αυξάνει σε μεγάλο βαθμό την κλιμακωσιμότητα της αρχιτεκτονικής αφού επιτρέπει την εύκολη πρόσθεση επιπρόσθετων server για μεγαλύτερη και αποδοτικότερη εξυπηρέτηση των εισερχομένων ερωτημάτων. Υπάρχουν διάφοροι αλγόριθμοι κατανομής φόρτου οι οποίοι μπορούν να χρησιμοποιηθούν όπως ο Round Robin (κατανομή αιτημάτων στους servers με την σειρά), Least Connections (προτεραιότητα στον server με τις λιγότερες ενεργές συνδέσεις), Least Time (προτεραιότητα στον server με τον μικρότερο χρόνο απόκρισης και τις λιγότερες συνδέσεις), Hash (κατανομή με βάση κάποιο κλειδί πχ IP address) κτλ. Η ισορροπημένη κατανομή του φόρτου στο Serving Layer αποτελεί ένα σημαντικό κομμάτι της αρχιτεκτονικής, αφού μέσω της ικανότητας για αποδοτική εξυπηρέτηση μεγάλου όγκου αιτημάτων, προσφέρεται η δυνατότητα σύνδεσης και συνεργασίας του συστήματος αυτού με άλλα σύγχρονα IoT συστήματα, προσφέροντας έτσι μεγάλη ευελιξία και δυνατότητες, όσον αφορά την λειτουργία του συστήματος.

- **End Users:** Υπάρχει ποικιλία όσον αφορά τους τελικούς χρήστες ενός σύγχρονού IoT συστήματος. Ένας τελικός χρήστης εκτός από κάποιο φυσικό πρόσωπο, μπορεί να είναι μία εφαρμογή η οποία χρησιμοποιεί το REST API του υπάρχοντος συστήματος ή ακόμη ένα άλλο API το οποίο συνδέεται με το υπάρχον σύστημα. Ακόμη λόγω της γρήγορης εξέλιξης και των υψηλών απαιτήσεων που παρουσιάζει ο τομέας του Internet of Things, μπορεί να κριθεί αναγκαίος ο συνδυασμός και η συνεργασία μεταξύ πολλών σύγχρονων IoT συστημάτων. Έτσι ένας άλλος πιθανός χρήστης του συστήματος αυτού μπορεί να είναι ένα άλλο IoT σύστημα, του οποίου η λειτουργία βασίζεται στην συνεργασία με το υπάρχον σύστημα.

4.3 Μετρικές – Άξονες αξιολόγησης

Dataset

Το dataset το οποίο χρησιμοποιήθηκε για την μελέτη του συστήματος όσον αφορά την απόδοση (throughput), κλιμακωσιμότητα και χρόνο απόκρισης αποτελείται από απλά JSON objects της μορφής:

```
{
        “Sensor”: sensor, “Topic”: topic, “Value”: value, “Timestamp”: timestamp
}
```

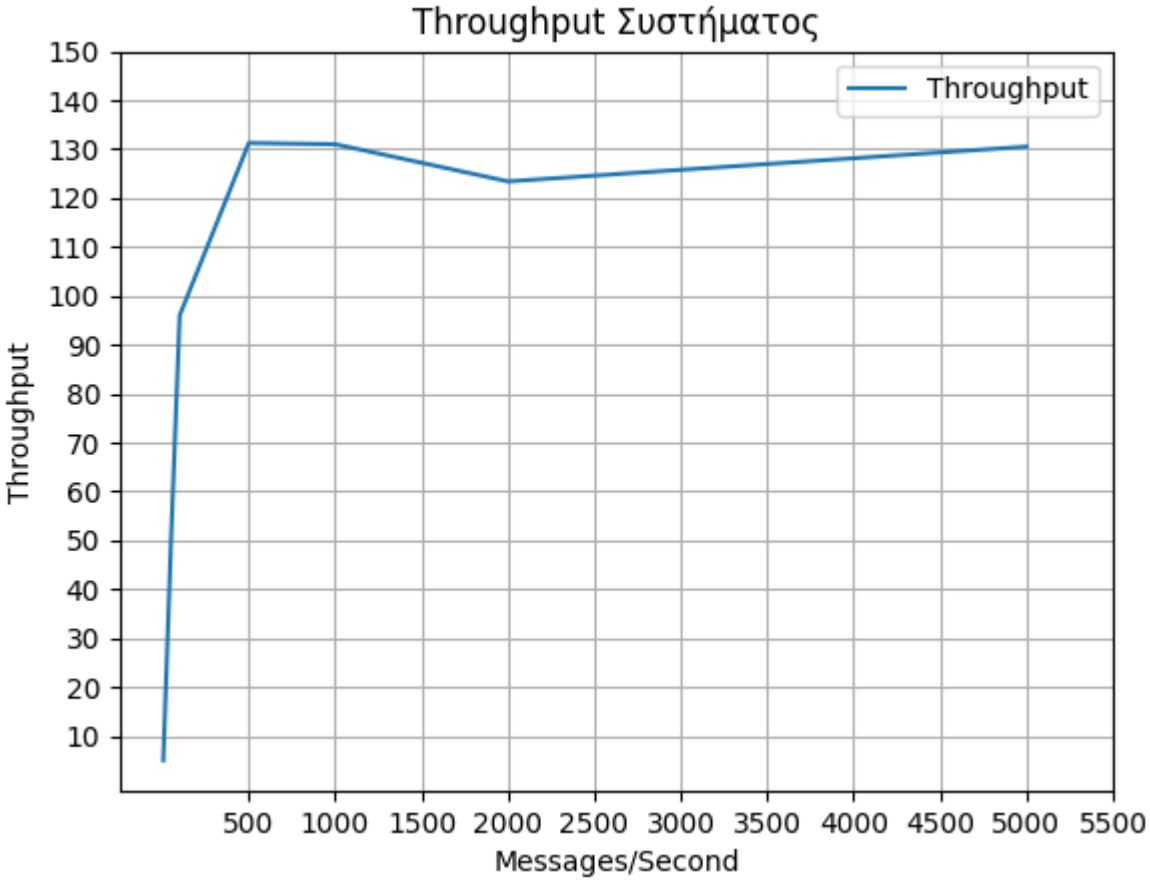
όπου το πεδίο Sensor αντιστοιχεί στον κωδικό του αισθητήρα, το πεδίο Topic στο ανάλογο topic του broker στον οποίο κοινοποιούνται τα δεδομένα, το πεδίο Value εμπεριέχει την τιμή της θερμοκρασίας η οποία παράγεται από τον αισθητήρα και το πεδίο Timestamp το οποίο χρησιμοποιείται κατά την εκτέλεση της πειραματικής διαδικασίας για την λήψη των μετρικών.

Μετρικές Throughput – Response Time

Κατά την πειραματική διαδικασία αξιολόγησης της απόδοσης (throughput) και του χρόνου απόκρισης (response time) του συστήματος λήφθηκαν οι παρακάτω μετρήσεις για διάφορες τιμές ρυθμού (μηνύματα/δευτερόλεπτο) δημιουργίας και αποστολής τιμών θερμοκρασίας από τους αισθητήρες εστιάζοντας σε έναν broker του Kafka cluster. Οι τιμές αυτές λήφθηκαν σε κατάλληλα σημεία του συστήματος με σκοπό την αξιολόγηση της λειτουργίας των κύριων επιμέρους στοιχείων τα οποία συνθέτουν μια σύγχρονη IoT αρχιτεκτονική. Συγκεκριμένα οι τιμές αυτές αντιστοιχούν στο throughput και response time του συστήματος λαμβάνοντας υπόψη την χρονική στιγμή κατά την οποία παράγονται οι τιμές στους αισθητήρες μέχρι και την χρονική στιγμή που θα προωθηθούν στο Serving Layer για διάθεση στους χρήστες, έχοντας εφαρμόσει πάνω στις τιμές των θερμοκρασιών μερική επεξεργασία όπως φίλτρα, συναρτήσεις καθαρισμού, στρογγυλοποίησης κτλ .

Throughput

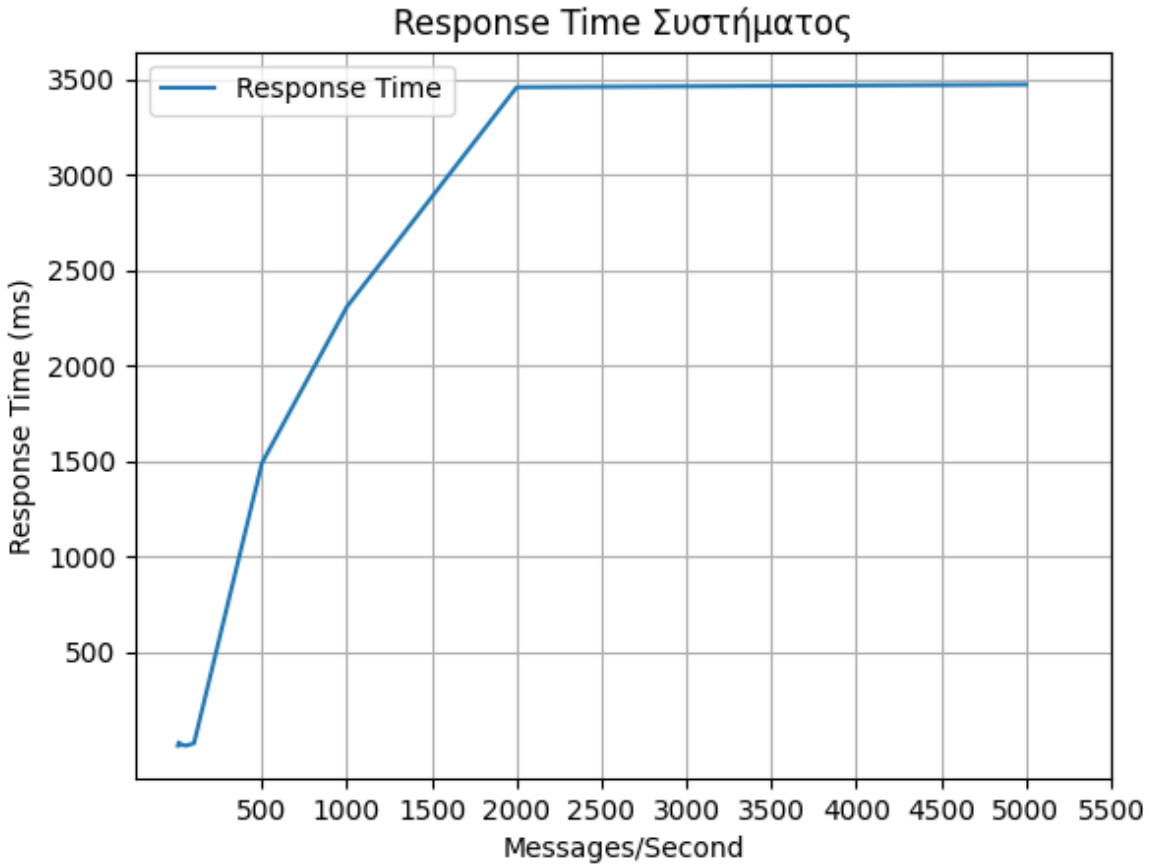
Μηνύματα / Δευτερόλεπτο	**Throughput**
5	5
10	10
20	20
50	50
100	96
500	131.25
1000	131
2000	123.4
5000	130.5



Διάγραμμα 1 – Throughput Συστήματος

Response Time

Μηνύματα / Δευτερόλεπτο	**Response Time (ms)**
5	8.596
10	25.43
20	17.288
50	8.058
100	21.382
500	1489.65
1000	2307.852
2000	3458.71
5000	3472.322



Διάγραμμα 2 – Response Time Συστήματος

Συμπεράσματα

Όπως παρατηρούμε η απόδοση του συστήματος αρχικά αυξάνεται ανάλογα με τον ρυθμό μετάδοσης των μηνυμάτων. Από κάποιο σημείο και μετά δεν παρατηρούνται μεγάλες μεταβολές, αλλά υπάρχει μια σχετική σταθεροποίηση στην τιμή του throughput. Ακόμη μέσω του διαγράμματος το οποίο αφορά τον χρόνο απόκρισης του συστήματος παρατηρούμε μια αρχική αύξηση καθώς αυξάνεται ο ρυθμός μετάδοσης των μηνυμάτων η οποία φαίνεται να σταθεροποιείται παρόμοια με το throughput για μεγάλες τιμές ρυθμού μετάδοσης. Παρά το γεγονός ότι υπήρξαν οι περιορισμοί αυτοί, παρατηρείται ότι το σύστημα διατηρεί σχετικά σταθερό throughput και χρόνο απόκρισης καθώς αυξάνεται ο ρυθμός μετάδοσης των δεδομένων. Το φαινόμενα αυτά είναι απολύτως λογικά και οφείλονται στους περιορισμούς που παρουσιάζει τόσο το hardware όσο και το δίκτυο μεταφοράς των δεδομένων μεταξύ των επιπέδων του συστήματος, το οποίο χρησιμοποιήθηκε για την υλοποίηση του εν λόγω συστήματος και στο οποίο λήφθηκαν οι παραπάνω μετρήσεις.

Πλεονεκτήματα Αρχιτεκτονικής

Κλιμακωσιμότητα	Το σύστημα προσφέρει μεγάλες δυνατότητες οριζόντιας κλιμακωσιμότητας μέσω της εύκολης προσθήκης κόμβων στα διάφορα cluster των επιπέδων όπως το cluster των MQTT broker, των Kafka Broker, του Spark καθώς και στο cloudlet των Kafka Producer. Έτσι προσθέτοντας κόμβους στα διάφορα cluster δίνεται η δυνατότητα στο σύστημα να διαχειριστεί αποδοτικότερα μεγαλύτερο υπολογιστικό φόρτο.
Υψηλή Διαθεσιμότητα	Μέσω της ικανότητας δημιουργίας πολλαπλών αντιγράφων των δεδομένων μέσα στα cluster των MQTT broker, των Kafka Broker και του Spark καθώς και με την χρήση του HDFS στο Streaming Layer, εξασφαλίζεται υψηλή διαθεσιμότητα των δεδομένων, χωρίς αυτή να επηρεάζεται από τυχόν σφάλματα του δικτύου ή των κόμβων στο σύστημα.
Ανοχή στα σφάλματα	Το υπάρχον σύστημα παρουσιάζει μεγάλη ανοχή σε σφάλματα όσον αφορά τα μηχανήματα – κόμβους του συστήματος. Σε περίπτωση σφάλματος κάποιου κόμβου ο οποίος αποτελεί leader κάποιου partition σε έναν Kafka broker, γίνεται αυτόματη εκλογή νέου έτσι ώστε να μην επηρεαστεί η λειτουργία του συστήματος σε περίπτωση σφάλματος.
Stream replay	Η χρήση της τεχνολογίας Apache Kafka ως κύριο μηχανισμό για την επεξεργασία των ροών δεδομένων σε πραγματικό χρόνο, επιτρέπει την διατήρηση των δεδομένων για ορισμένο χρονικό διάστημα, δίνοντας την δυνατότητα επαναχρησιμοποίησης των δεδομένων αυτών ως συνεχή ροή, σε περίπτωση που υπάρξουν αλλαγές στον κώδικα επεξεργασίας του συστήματος οι οποίες καθιστούν outdated τα ήδη επεξεργασμένα δεδομένα.
Χαμηλός χρόνος απόκρισης	Το σύστημα χαρακτηρίζεται από χαμηλό χρόνο απόκρισης λόγω της χρήσης στοιχείων edge computing. Το σύστημα μέσω του Input Layer εκτελεί μερική επεξεργασία των δεδομένων σε τοπικό επίπεδο κοντά στους αισθητήρες με την βοήθεια της αρχιτεκτονικής των Cloudlet. Αυτό έχει ως αποτέλεσμα την αφαίρεση υπολογιστικού φόρτου από το Streaming Layer το οποίο υλοποιείται κυρίως σε cloud επίπεδο μειώνοντας έτσι το μέγεθος και το εύρος των δεδομένων που πρέπει να μεταφερθούν στο cloud για επεξεργασία και παράλληλα τον χρόνο απόκρισης του συστήματος. Ακόμη η χρήση του Load Balancer στο Serving Layer προσφέρει στην ισόποση κατανομή των αιτημάτων από τους χρήστες στους server του συστήματος, μειώνοντας έτσι το χρόνο εξυπηρέτησης.
Remote Sensor Control	Μέσω μιας αντίστροφης ροής μέσα στο σύστημα ο χρήστης ή κάποιος διαχειριστής, μπορεί να προβεί σε αλλαγές στις ρυθμίσεις των αισθητήρων, απομακρυσμένα.

5. Επίλογος

Ο σκοπός της παρούσας διπλωματικής εργασίας ήταν η μελέτη αρχιτεκτονικών επεξεργασίας δεδομένων μεγάλης κλίμακας, οι οποίες σε συνδυασμό με την μελέτη τεχνικών μετάδοσης δεδομένων - πρωτοκόλλων επικοινωνίας μεταξύ των συσκευών, τεχνικών αποθήκευσης δεδομένων μεγάλης κλίμακας, τεχνικών streaming και επεξεργασίας δεδομένων σε πραγματικό χρόνο, τεχνικών διαμοιρασμού πόρων και παράλληλης επεξεργασίας δεδομένων καθώς και τεχνικών ενορχήστρωσης, συντονισμού συστήματος και απομόνωσης εφαρμογών θα οδηγήσουν σε εύρεση βέλτιστων μεθόδων αντιμετώπισης των προβλημάτων διαχείρισης και επεξεργασίας του τεραστίου όγκου δεδομένων. Στην παρούσα διπλωματική εργασία μελετήθηκαν διάφορες αρχιτεκτονικές επεξεργασίας δεδομένων μεγάλης κλίμακας (Big Data), οι οποίες αφορούσαν τόσο την αρχιτεκτονική του Cloud Computing όσο και την αρχιτεκτονική του Edge Computing. Έγινε εκτενής ανάλυση της λειτουργίας, των πλεονεκτημάτων και των μειονεκτημάτων που παρουσιάζει η κάθε μια αρχιτεκτονική, όπως επίσης και τις εφαρμογές τους στον τομέα του Internet of Things. Μέσω της σύγκρισης και της ανάλυσης των δυο αυτών αρχιτεκτονικών αναδείχθηκε η ανάγκη για συνδυασμό τους σε μια καθολική αρχιτεκτονική η οποία έχει να προσφέρει σημαντικά πλεονεκτήματα στα σύγχρονα IoT συστήματα. Πλεονεκτήματα όπως η μείωση των χρόνων απόκρισης, μεταφορά υπολογιστικού φόρτου κοντά στις πηγές παραγωγής και αποσυμφόρηση του cloud, μείωση συμφόρησης στο δίκτυο κατά την μεταφορά των δεδομένων – εξοικονόμηση bandwidth, δυνατότητες οριζόντιας κλιμακωσιμότητας και γεωγραφικής κατανομής του συστήματος όπως επίσης και μείωση του ενεργειακού κόστους και απαιτήσεων των συσκευών.

Ακόμη έγινε μελέτη διάφορων τεχνικών και τεχνολογιών για την υλοποίηση των επιμέρους επιπέδων μιας αρχιτεκτονικής επεξεργασίας δεδομένων μεγάλης κλίμακας η οποία μπορεί να εφαρμοστεί σε IoT συστήματα, παρουσιάζοντας την λειτουργία, καθώς και τα πλεονεκτήματα και μειονεκτήματα της κάθε μιας.

Τέλος μέσω της υλοποίησης μια μικρής κλίμακας αρχιτεκτονικής επεξεργασίας μεγάλου όγκου δεδομένων, συνδυάστηκαν μερικά από τα είδη αρχιτεκτονικών, τεχνικών και τεχνολογιών τα οποία παρουσιάστηκαν και αναλύθηκαν στις προηγούμενες ενότητες, προσφέροντας έτσι ένα μικρό παράδειγμα δημιουργίας μιας σύγχρονης, κλιμακώσιμης και λειτουργικής αρχιτεκτονικής, η οποία είναι ικανή να εξυπηρετήσει τις ανάγκες ενός σύγχρονου IoT συστήματος.

Η παρούσα διπλωματική εργασία προσέφερε στον αναγνώστη την δυνατότητα να μελετήσει διάφορες σύγχρονες τεχνολογίες και αρχιτεκτονικές οι οποίες αφορούν την επεξεργασία μεγάλου όγκου δεδομένων. Έτσι με βάση τα πλεονεκτήματα και μειονεκτήματα που παρουσιάζει η κάθε μια τεχνολογία και αρχιτεκτονική, ο αναγνώστης έχει την δυνατότητα να συνδυάσει διάφορες από τις λύσεις τις οποίες έχουν μελετηθεί, με σκοπό την δημιουργία σύγχρονων και κλιμακώσιμων αρχιτεκτονικών, οι οποίες θα είναι ικανές να εξυπηρετήσουν ποικίλα σενάρια χρήσης και τις απαιτήσεις των σύγχρονων IoT συστημάτων. Το γεγονός αυτό θα συνεισφέρει στην αποδοτική αντιμετώπιση των διαφόρων προβλημάτων και περιορισμών που παρουσιάζουν τα σημερινά IoT συστήματα, λόγω της ανάγκης για αποδοτική επεξεργασία των δεδομένων μεγάλης κλίμακας (Big Data).

Βιβλιογραφία

[1] Ranger, S. (2020, February 3). *What is the IoT? Everything you need to know about the Internet of Things right now.* Ανάκτηση από ZDNet: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

[2] Microsoft. (2018, December 2). *Big data architectures*. Ανάκτηση από Microsoft: https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/

[3] Iman Samizadeh, P. (2018, March 15). *A brief introduction to two data processing architectures — Lambda and Kappa for Big Data*. Ανάκτηση από Medium: https://towardsdatascience.com/a-brief-introduction-to-two-data-processing-architectures-lambda-and-kappa-for-big-data-4f35c28005bb

[4] *Lambda Architecture*. (n.d.). Ανάκτηση από Databricks: https://databricks.com/glossary/lambda-architecture

[5] Verrilli, M. (2017, August 28). *talend*. Ανάκτηση από From Lambda to Kappa: A Guide on Real-time Big Data Architectures: https://www.talend.com/blog/2017/08/28/lambda-kappa-real-time-big-data-architectures/

[6] Kreps, J. (2014, July 2). *Questioning the Lambda Architecture*. Ανάκτηση από oreilly: https://www.oreilly.com/radar/questioning-the-lambda-architecture/

[7] Gezer, V., Um, J., & Ruskowski, M. (2017, November). An Extensible Edge Computing Architecture: Definition, Requirements and Enablers.

[8] Ai, Y., Peng, M., & Zhang, K. (2018). Edge computing technologies for Internet of Things: a primer.

[9] Sheth, V. (2019, February 14). *Fog / Edge-computing and Cloudlets*. Ανάκτηση από medium: https://medium.com/@virral/fog-edge-computing-and-cloudlets-c4db6cc9b15a

[10] *Cloudlet-based Edge Computing*. (n.d.). Ανάκτηση από elijah: http://elijah.cs.cmu.edu/

[11] Borcoci, E., & Obreja , S. (2018). Edge Computing Architectures – A Survey on Convergence of Solutions.

[12] Kunal, S., Saha, A., & Amin, R. (2019). An overview of cloud-fog computing: Architectures, applications with security challenges.

[13] Naha1, R. K., Garg, S., & Chan , A. (2018). Fog Computing Architecture: Survey and Challenges.

[14] Antonini, M., Vecchio, M., & Antonelli, F. (2019). Fog Computing Architectures: a Reference for Practitioners.

[15] Rahman, G., & Wen, C. C. (2018). Fog Computing, Applications , Security and Challenges, Review. *International Journal of Engineering & Technology*.

[16] *Multi-access Edge Computing (MEC)*. (n.d.). Ανάκτηση από ETSI: https://www.etsi.org/technologies/multi-access-edge-computing

[17] Mach, P., & Becvar, Z. (2017). Mobile Edge Computing: A Survey on Architecture and Computation Offloading.

[18] Adib, D. (n.d.). *Mobile edge computing (MEC): What is mobile edge computing?* Ανάκτηση από stlpartners: https://stlpartners.com/edge-computing/mobile-edge-computing/

[19] Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2017). Mobile Edge Computing: A Survey.

[20] Jaikar, S. P., & Iyer, D. K. (2018). A Survey of Messaging Protocols for IoT Systems.

[21] Ali, A. A. (2018). Constrained Application Protocol (CoAP) for the IoT.

[22] Gour, R. (2019, December 5). *4 Major IoT Protocols — MQTT, CoAP, AMQP, DDS*. Ανάκτηση από Medium: https://medium.com/@rinu.gour123/4-major-iot-protocols-mqtt-coap-amqp-dds-46016897c3e9

[23] DIZDAREVIĆ, J., CARPIO, F., & JUKAN, A. (2019). A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration.

[24] Borthakur, D. (2019, August 22). *HDFS Architecture Guide*. Ανάκτηση από Hadoop: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[25] AWS. (2020). *Introduction to Amazon S3*. Ανάκτηση από Amazon AWS: https://docs.aws.amazon.com/AmazonS3/latest/dev/Introduction.html#overview

[26] Kayay, C. (2018, June 8). *How do you pick the right database for Big Data Architecture?* Ανάκτηση από Medium: https://medium.com/@ckayay/how-to-pick-the-right-database-c2539efe2589

[27] Prabhu. (2019, August 11). *Architecture of Elastic Search & Installation — Part 2*. Ανάκτηση από Medium: https://medium.com/everythingatonce/architecture-of-elastic-search-installation-part-2-ec56a5a3c192

[28] *Introduction*. (n.d.). Ανάκτηση από Apache Kafka: https://kafka.apache.org/intro.html

[29] *Tutorial*. (n.d.). Ανάκτηση από Apache Storm: https://storm.apache.org/releases/current/Tutorial.html

[30] Foundation, A. S. (n.d.). *What is Apache Flink? — Architecture*. Ανάκτηση από Apache Flink: https://flink.apache.org/flink-architecture.html

[31] Garg, A. (2020, January 30). *Apache Spark Architecture*. Ανάκτηση από IntelliPaat: https://intellipaat.com/blog/tutorial/spark-tutorial/spark-architecture/

[32] DATAFLAIR. (2019, February 28). *Top Advantages and Disadvantages of Hadoop 3*. Ανάκτηση από data-flair: https://data-flair.training/blogs/advantages-and-disadvantages-of-hadoop/

[33] *Docker overview*. (n.d.). Ανάκτηση από Docker: https://docs.docker.com/get-started/overview/

[34] Authors, K. (2020, May 30). *Kubernetes Components*. Ανάκτηση από Kubernetes: https://kubernetes.io/docs/concepts/overview/components/

[35] Chand, M. (2018, January 30). *Introduction to Apache Mesos*. Ανάκτηση από DZone: https://dzone.com/articles/introduction-to-apache-mesos