ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Διαχείριση της συγκατάθεσης των χρηστών για ρεύματα δεδομένων προσωπικού χαρακτήρα σε διασυνδεδεμένα οχήματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Λαμπρόπουλου Λ. Γεώργιου

Επιβλέπων: Δημήτριος Ασκούνης
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2019

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Διαχείριση της συγκατάθεσης των χρηστών για ρεύματα δεδομένων προσωπικού χαρακτήρα σε διασυνδεδεμένα οχήματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Λαμπρόπουλου Λ. Γεώργιου

Επιβλέπων: Δημήτριος Ασκούνης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τη 18η Νοεμβρίου 2019

.....	.....	.....
Δημήτριος Ασκούνης	Ιωάννης Ψαρράς	Χάρης Δούκας
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2019

.....

Γεώργιος Λαμπρόπουλος

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

Περίληψη

Η διασύνδεση των οχημάτων και η ανταλλαγή δεδομένων μεταξύ τους, με το δίκτυο και άλλες οντότητες επιτρέπουν την ανάπτυξη πλήθους εφαρμογών προστιθέμενης αξίας στους τομείς της ασφάλειας, της βέλτιστης δρομολόγησης, της ψυχαγωγίας κ.α. Ωστόσο, η ανάπτυξη των εφαρμογών αυτών προϋποθέτει την πρόσβαση και χρήση των δεδομένων από τους διαφόρους παρόχους υπηρεσιών.

Η αξιοποίηση των δεδομένων, πρέπει να γίνεται με πλήρη διασφάλιση της ιδιωτικότητας, της προστασίας των προσωπικών δεδομένων και της συγκατάθεσης του χρήστη. Επίσης, η δυνατότητα πρόσβασης και αξιοποίησης των δεδομένων πρέπει να παρέχεται ισότιμα προς τους διάφορους παρόχους, για εξασφάλιση υγιούς ανταγωνισμού και ίσων ευκαιριών μεταξύ των επιχειρήσεων, και να αποφεύγεται προνομιακή αξιοποίησή τους από συγκεκριμένα μέρη (πχ. κατασκευαστές αυτοκινήτων, κατασκευαστές ανταλλακτικών)

Στόχος της παρούσας εργασίας είναι, αφού οριοθετήσει το πεδίο μελέτης των δεδομένων μεγάλης κλίμακας και των διασυνδεδεμένων και αυτοματοποιημένων οχημάτων, να προτείνει μια αρχιτεκτονική και να υλοποιήσει τη λύση για τη διαχείριση της συγκατάθεσης των χρηστών επί ρευμάτων δεδομένων προσωπικού χαρακτήρα που παρέχονται από τα οχήματα σε παρόχους υπηρεσιών.

Η προτεινόμενη αρχιτεκτονική λύση βασίζεται στην υπόθεση εργασίας του Κοινόχρηστου Επεξεργαστή η οποία έχει προταθεί από την ΕΕ ως μια βραχυπρόθεσμη τεχνική λύση, για τη διαχείριση των δεδομένων των οχημάτων και τη διάθεση τους σε παρόχους υπηρεσιών.

Διερευνώντας το περιβάλλον του προβλήματος, διαπιστώθηκε πως η ενδεδειγμένη λύση θα πρέπει να βασίζεται στην αποδοτική διαχείριση της συγκατάθεσης των χρηστών με βάση τα πεδία (attributes) των ρευμάτων δεδομένων που παράγονται από τα οχήματα. Πιο συγκεκριμένα το πρόβλημα που επιλύουμε στην παρούσα εργασία είναι η σε πραγματικό χρόνο διαχείριση της εξουσιοδότησης για πρόσβαση σε ρεύματα δεδομένων με βάση την επιθυμία των χρηστών εκφρασμένη σε επίπεδο χαρακτηριστικού για τα δεδομένα αυτά (Consent management of streaming data by Attribute Based Access Control).

Εξειδικεύοντας την προτεινόμενη αρχιτεκτονική, στο πλαίσιο της διπλωματικής, γίνεται προγραμματιστική υλοποίηση της λύσης σε τεχνολογία Apache Spark. Το προγραμματιστικό περιβάλλον στο οποίο πραγματοποιείται η υλοποίηση και τα εργαλεία που αξιοποιούνται παρέχονται από την πλατφόρμα Microsoft Azure.

Για το σχεδιασμό της προτεινόμενης αρχιτεκτονικής λύσης και υλοποίησης, μελετήθηκαν μεταξύ άλλων, εναλλακτικές υλοποιήσεις, με στόχο την ελαχιστοποίηση του χρόνου επεξεργασίας των ρευμάτων δεδομένων ώστε η διάθεσή τους να μπορεί να συναντήσει τις απαιτήσεις για την παροχή υπηρεσιών σχεδόν πραγματικού χρόνου από τους παρόχους στους τελικούς χρήστες.

Ως συμπέρασμα, προκύπτει ότι η σε πραγματικό χρόνο διαχείριση της συγκατάθεσης της πρόσβασης σε ρεύματα δεδομένων με βάση τις προτιμήσεις των χρηστών για επιλεγμένα χαρακτηριστικά των δεδομένων είναι εφικτή και εξαιρετικά αποδοτική με την αξιοποίηση τεχνολογιών επεξεργασίας ρευμάτων δεδομένων και δεδομένων μεγάλης κλίμακας όπως η τεχνολογία Apache Spark. Παράλληλα μπορούμε να συμπεράνουμε πως η προτεινόμενη αρχιτεκτονική και η σχετική υλοποίηση είναι κατάλληλες για το περιβάλλον της διαχείρισης των ρευμάτων δεδομένων από διασυνδεμένα οχήματα δίνοντας τη δυνατότητα να παρασχεθούν υπηρεσίες σε παρόχους υπηρεσιών και τελικούς χρήστες σε πραγματικό χρόνο.

Λέξεις - Κλειδιά: ρεύματα δεδομένων, διαχείριση συγκατάθεσης σε ρεύματα δεδομένων, ABAC on streaming data, δεδομένα μεγάλης κλίμακας, διασυνδεδεμένα οχήματα, συνεργατικά και ευφυή συστήματα μεταφορών, Apache Spark, συγκατάθεση χρήστη

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

Connectivity features on vehicles and data exchange either within fleet or with other entities, such as the network infrastructure, enable the development of many value-adding, innovative applications and services in the areas of transport safety, optimal routing, entertainment etc. However, key requirement for the development of these applications is the access and use of the vehicle data by the relevant service providers.

Prerequisite for data access from various providers should be the assurance of privacy as well as the protection of personal data and user consent. Also, equality principle on the access and use of data among the various service providers should be ensured. Equal access ensures fair competition and equal opportunities for businesses to develop and provide new products and services.

Focus area of this thesis is the access management of service providers on vehicle data, based on user consent. More specifically, scope of current thesis is to propose an appropriate system architecture and develop a customized software solution, which manages providers' access on vehicle data streams, upon considering the user consent input. Before presenting proposed architecture and implementation, our work provides a literature review on key elements of its focus area, which include big data and connected and automated vehicles.

The proposed system architecture is based on the Shared Server concept, which is the selected EU mid-term technical solution to centrally manage vehicle data and make them available to service providers.

Upon detailed study on the access management problem and technical constraints, we concluded that the appropriate solution should be based on the efficient management of user's consent which will be linked with the attributes of the data streams generated by vehicles. Specifically, the implemented solution performs real-time management of user consent and provides access to data streams based on users' decision, which is expressed at attribute level. (Consent management of streaming data by Attribute Based Access Control)

On this project, the development of proposed solution is based on Apache Spark platform, while the implementation environment and deployment tools are provided within Microsoft Azure platform.

During the design of the proposed solution and implementation, we also examined alternative implementations aiming to minimize the processing time of data streams in order to be available to service providers in a timely manner and enable the provision of near real-time services to end-users.

In conclusion, the real-time access management to data streams, based on users' consent, proves to be feasible and efficient, through utilization of data streaming technologies and big data technologies such as Apache Spark. Also, the Microsoft Azure platform provides the required tools and platforms for the various layers of the implementation. Finally, we assume that the proposed architecture and implementation are suitable for vehicle data streams' access management and allow service providers to provide near real-time services.

Keywords: data stream, consent management on streaming data, ABAC on streaming data, big data, connected vehicles, cooperative intelligent transport systems, Apache Spark, user consent

Η σελίδα αυτή είναι σκόπιμα λευκή.

Ευχαριστίες

Με την παρούσα διπλωματική ολοκληρώνονται οι σπουδές μου στη Σχολή των Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π.. Πριν την παρουσίαση της διπλωματικής εργασίας θα ήθελα να ευχαριστήσω όλους όσους συνεργάστηκαν και έπαιξαν καθοριστικό ρόλο για την ολοκλήρωσή της.

Αρχικά, ευχαριστώ θερμά τον επιβλέποντα της διπλωματικής μου εργασίας, Καθηγητή κύριο Δημήτριο Ασκούνη για την εμπιστοσύνη που μου έδειξε αναθέτοντάς μου αυτή την εργασία και για τη συνεχή παροχή υποστήριξης.

Επιπλέον, θα ήθελα να εκφράσω τις ευχαριστίες μου στον συνεπιβλέποντα κύριο Κωνσταντίνο Τζαμαλούκα για την ουσιαστική καθοδήγηση που μου παρείχε και για την υποστήριξη σε κάθε στάδιο κατά την εκπόνηση της διπλωματικής εργασίας, αφιερώνοντας πολύτιμο χρόνο για εμένα.

Τέλος ευχαριστίες απευθύνω και στα άλλα δύο μέλη της τριμελούς επιτροπής, κύριο Ιωάννη Ψαρρά και κύριο Χάρη Δούκα που δέχτηκαν να συμμετάσχουν στην τριμελή επιτροπή αξιολόγησης της διπλωματικής μου εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

Κατάλογος εικόνων και πινάκων

1. Εισαγωγή

1.1 Αντικείμενο Διπλωματικής

Η εξέλιξη της τεχνολογίας και η αξιοποίησή της στον τομέα της αυτοκίνησης έδωσε τη δυνατότητα για διασύνδεση των οχημάτων με άλλες οντότητες, όπως για παράδειγμα άλλα οχήματα ή στοιχεία της υποδομής των οδικών δικτύων αλλά και τη δυνατότητα αυτοματοποίησης των οχημάτων. Τα οχήματα αποτελούν πηγές παραγωγής τεράστιου όγκου δεδομένων και η διασύνδεσή τους έχει δώσει πλέον τη δυνατότητα για διαχείριση, επεξεργασία και αξιοποίηση τόσο αυτών των δεδομένων όσο και των συμπερασμάτων που προκύπτουν από την ανάλυση τους.

Στην κατεύθυνση αυτή, δημιουργείται η ευκαιρία για ανάπτυξη καινοτόμων εφαρμογών και υπηρεσιών, οι οποίες έχουν να προσφέρουν πολλαπλά οφέλη στους χρήστες σε διάφορους τομείς όπως είναι η ασφάλεια των μετακινήσεων, η βελτιστοποίηση των διαδρομών, η ψυχαγωγία κ.α. Ωστόσο η ευκαιρία ανάπτυξης τέτοιων εφαρμογών, είναι δυνατή μόνο με πρόσβαση στο σύνολο των δεδομένων. Συνεπώς το ποιος θα έχει πρόσβαση στα δεδομένα των οχημάτων είναι ένα ζήτημα που απαιτεί προσεκτικό χειρισμό, πάντα υπό το πρίσμα της προστασίας της ιδιωτικότητας των προσωπικών αυτών δεδομένων, αλλά παράλληλα με στόχο τη μεγιστοποίηση του οφέλους για τον χρήστη, που θα προέλθει από τη βέλτιστη αξιοποίηση των δεδομένων.

Με την παρούσα διπλωματική, θα διατρέξουμε το πεδίο έρευνας, το οποίο συνθέτουν α) τα δεδομένα μεγάλης κλίμακας, β) τα διασυνδεδεμένα και αυτοματοποιημένα οχήματα, γ) η υπόθεση εργασίας του Κοινόχρηστου Επεξεργαστή και δ) οι τεχνολογίες που χρησιμοποιούνται στα προαναφερθέντα στοιχεία του πεδίου έρευνας.

Σημειώνεται ότι η λύση του «Κοινόχρηστου επεξεργαστή» αποτελεί τη βραχυπρόθεσμη λύση που προτάθηκε στο πλαίσιο μελέτης που έγινε για την Ευρωπαϊκή Επιτροπή με στόχο τη διαχείριση της πρόσβασης στα δεδομένα των οχημάτων.

Στην παρούσα διπλωματική προτείνεται αρχιτεκτονική, βασισμένη στην υπόθεση εργασίας του Κοινόχρηστου Επεξεργαστή, με στόχο τη διαχείριση των ρευμάτων δεδομένων των οχημάτων και τη διάθεσή τους σε παρόχους υπηρεσιών βάσει της επιθυμίας των χρηστών.

Επίσης, ως εξειδίκευση της προτεινόμενης αρχιτεκτονικής, περιλαμβάνεται προγραμματιστική υλοποίηση μέσω της οποίας επιτυγχάνεται η σε πραγματικό χρόνο διαχείριση της εξουσιοδότησης για πρόσβαση στα ρεύματα δεδομένων των οχημάτων σύμφωνα με τη συγκατάθεση των χρηστών η οποία είναι εκφρασμένη σε επίπεδο χαρακτηριστικού για τα δεδομένα αυτά και δηλώνει την επιθυμία των χρηστών για παροχή του συνόλου ή επιλεγμένων δεδομένων των οχημάτων τους σε παρόχους υπηρεσιών της προτίμησής τους.

1.2 Στόχοι Διπλωματικής

Στο πλαίσιο του παραπάνω αντικειμένου, επιθυμούμε την επίτευξη των παρακάτω στόχων:

- Περιγραφή και κατανόηση των στοιχείων του πεδίου έρευνας

- Μοντελοποίηση του προβλήματος της διαχείρισης της πρόσβασης πάνω σε ρεύματα δεδομένων των οχημάτων βάσει της συγκατάθεσης των χρηστών
- Σχεδιασμός αντίστοιχου υποσυστήματος διαχείρισης της συγκατάθεσης, ως υποσύστημα του Κοινόχρηστου Επεξεργαστή
- Υλοποίηση του προαναφερθέντος συστήματος για την επίτευξη του ελέγχου της πρόσβασης πάνω στα δεδομένα των οχημάτων και παρουσίαση των χρησιμοποιούμενων τεχνολογιών.

1.3 Οργάνωση Κειμένου

- Στο 1ο Κεφάλαιο γίνεται η εισαγωγή στο αντικείμενο εργασίας και παρουσιάζεται η οργάνωση του κειμένου
- Στο 2ο κεφάλαιο γίνεται εισαγωγή και περιγραφή στις έννοιες του πεδίου έρευνας της διπλωματικής
- Στο 3ο κεφάλαιο αναφερόμαστε στην υπόθεση εργασίας του Κοινόχρηστου Επεξεργαστή και στην τρέχουσα κατάσταση στη διαχείριση των δεδομένων των οχημάτων
- Στο 4ο κεφάλαιο παρουσιάζουμε τον τρόπο διαχείρισης της πρόσβασης στα δεδομένα των οχημάτων βάσει της συγκατάθεσης των χρηστών και την αρχιτεκτονική λύση του αντίστοιχου συστήματος σε επίπεδο υπηρεσιών
- Στο 5ο κεφάλαιο μελετάμε ορισμένους από τους κυριότερους μηχανισμούς ελέγχου πρόσβασης και την αναγωγή τους στο πρόβλημα της διαχείρισης της πρόσβασης στα δεδομένα των οχημάτων
- Στο 6ο κεφάλαιο παρουσιάζουμε τη μοντελοποίηση του προβλήματος και περιγράφουμε τις ροές επεξεργασίας σε υψηλό επίπεδο.
- Στο 7ο κεφάλαιο παρουσιάζουμε την αρχιτεκτονική του προτεινόμενου συστήματος σε τεχνολογικό επίπεδο και περιγράφουμε τα συστατικά στοιχεία της.
- Το 8ο κεφάλαιο περιγράφει συνοπτικά τις υπηρεσίες που παρέχει η πλατφόρμα Azure σχετικά με τα δεδομένα μεγάλης κλίμακας.
- Στα κεφάλαια 9 και 10, γίνεται αναλυτική περιγραφή του υποσυστήματος για την εφαρμογή του ελέγχου πρόσβασης στα ρεύματα δεδομένων των οχημάτων και των τεχνικών μας επιλογών στην υλοποίηση αυτού του συστήματος.
- Στο 11ο κεφάλαιο παρουσιάζεται ένα παράδειγμα λειτουργίας τους συστήματος που περιγράφηκε στα δύο προηγούμενα κεφάλαια
- Το κεφάλαιο 12 αποτελείται από τα συμπεράσματα της εργασίας
- Στα παραρτήματα, περιλαμβάνεται η τεχνοδιαμόρφωση της πλατφόρμας Azure (Παράρτημα 1ο) και ο κώδικας της υλοποίησης (Παράρτημα 2ο).

2 Ορισμοί και πεδίο έρευνας

2.1 Δεδομένα Μεγάλης Κλίμακας

2.1.1 Ορισμός

Ο όρος «Δεδομένα Μεγάλης Κλίμακας», ευρέως γνωστός με την ονομασία Big Data, χρησιμοποιείται τόσο στο ευρύτερο πεδίο της πληροφορικής όσο και στην επιχειρηματική κοινότητα για να περιγράψει τη ραγδαία παραγωγή τεράστιου όγκου δεδομένων από μεγάλο πλήθος πηγών. Η τάση αυτή αποτελεί πραγματικότητα τα τελευταία χρόνια και αποτυπώνεται απτά με τη διακίνηση τεράστιων και συνεχώς αυξανόμενων όγκων δεδομένων πάνω από τις δικτυακές υποδομές αλλά και με τη διαρκή αύξηση των εσόδων στην παγκόσμια αγορά από προσφερόμενες λύσεις πάνω σε Big Data. Το γεγονός αυτό αποτυπώνεται και μέσα από την έκθεση του οργανισμού IDC, στην οποία αναφέρεται ότι ο όγκος των δεδομένων το έτος 2025 προβλέπεται να φτάσει τα 175 zettabytes από τα 33 zettabytes που υπολογιζόταν το έτος 2018.[1] Αντίστοιχα τα έσοδα από την ευρύτερη οικονομία των δεδομένων μεγάλης κλίμακας και των προσφερόμενων λύσεων βασισμένων σε αυτά προβλέπεται να αγγίξουν το έτος 2019 τα 189 δις δολάρια σημειώνοντας αύξηση 12% σε σχέση με το έτος 2018, ενώ η αντίστοιχη εκτίμηση για το έτος 2022 αναφέρει ποσό που ανέρχεται στα 274.3 δις δολάρια.[2]

Παρότι ο όρος Big Data αποτελεί κοινό τόπο, δεν υπάρχει κάποιος αυστηρός ορισμός που να είναι κοινά αποδεκτός. Για την επεξήγηση του όρου μπορούμε να χρησιμοποιήσουμε τον ακόλουθο ορισμό: *Τα δεδομένα μεγάλης κλίμακας είναι σύνολα δεδομένων των οποίων η συλλογή, αποθήκευση, διαχείριση και ανάλυση απαιτεί νέες αρχιτεκτονικές, τεχνικές, αλγορίθμους και εφαρμογές, λόγω αυξημένου όγκου ή σημαντικής πολυπλοκότητας που εμπεριέχουν.* [3]

Εναλλακτικά, το βασικό στοιχείο που προσδίδει το χαρακτηρισμό Big Data σε ένα σύνολο δεδομένων είναι η αδυναμία των παραδοσιακών συστημάτων να τα διαχειριστούν με αποτελεσματικό τρόπο.

2.1.2 Χαρακτηριστικά Δεδομένων Μεγάλης Κλίμακας

Τα σύνολα δεδομένων στα οποία απονέμουμε τον όρο Big Data διέπονται από ένα σύνολο συγκεκριμένων χαρακτηριστικών. Αρχικά τα χαρακτηριστικά αυτά περιορίζονταν στον *όγκο* των δεδομένων (*Volume*), την *ταχύτητα* παραγωγής τους (*Velocity*) και την *ποικιλία* που χαρακτήριζε τον τύπο και τη μορφή τους *(Variety)*. Τα τρία χαρακτηριστικά αυτά δημιούργησαν τον όρο “3 Vs”. Στη συνέχεια όμως στο σύνολο αυτό προστέθηκε και το χαρακτηριστικό της *ποιότητας* των δεδομένων (*Veracity*) προσδίδοντας την έκφραση “4 Vs” για την περιγραφή των χαρακτηριστικών των Big Data. Τελικά στο σύνολο αυτό προστίθεται και η *αξία (Value)* που προκύπτει από την αξιοποίηση των Big Data καταλήγοντας να μιλάμε για “5 Vs”. Στο σύνολό τους περιγράφονται τα χαρακτηριστικά των Big Data ως εξής:

- Όγκος (Volume)

Ο όγκος αφορά την ποσότητα των δεδομένων που παράγονται και αποθηκεύονται. Αποτελεί καθοριστικό παράγοντα για το χαρακτηρισμό ενός συνόλου δεδομένων ως Big Data και για την αξία που μπορεί να προκύψει από την ανάλυση των δεδομένων αυτών.

- Ταχύτητα (Velocity)

Η ταχύτητα αναφέρεται στον ρυθμό με τον οποίο παράγονται και διακινούνται τα δεδομένα, ο οποίος είναι ιδιαίτερα αυξημένος για τα Big Data. Ως παράδειγμα μπορούμε να αναφέρουμε τα δεδομένα που προέρχονται από τα κοινωνικά δίκτυα, τα οποία παράγονται και διακινούνται με πολύ υψηλή ταχύτητα καθώς δισεκατομμύρια χρηστών ενεργούν ταυτόχρονα.

- Ποικιλία (Variety)

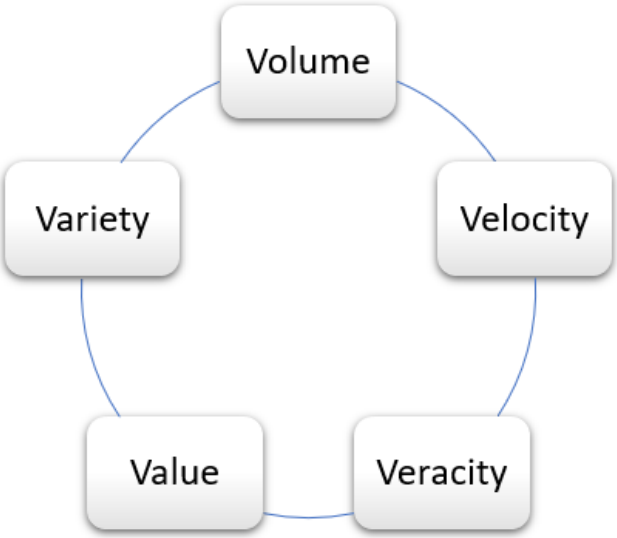
Η ποικιλία στα Big Data δηλώνει την διαφορά που παρουσιάζουν τα δεδομένα ως προς τον τύπο ή / και το βαθμό δόμησης. Για παράδειγμα, μπορεί να είναι δεδομένα που αποτελούνται από κείμενο, από εικόνες, από ήχο κ.α. ενώ μπορεί να είναι δομημένα, αδόμητα ή ημι-δομημένα.

- Ποιότητα (Veracity)

Η ποιότητα των δεδομένων αναφέρεται στην εγκυρότητα και την αξιοπιστία τους. Η ποιότητα των δεδομένων επηρεάζει άμεσα τη δυνατότητα ανάλυσής τους και επομένως την αξία που εξάγεται.

- Αξία (Value)

Η αξία είναι το επιθυμητό αποτέλεσμα της ανάλυσης των δεδομένων και της χρησιμοποίησής τους για την εξαγωγή αποφάσεων ή σημαντικών πληροφοριών και συμπερασμάτων.



Εικόνα 1: Χαρακτηριστικά Δεδομένων Μεγάλης Κλίμακας

2.1.3 Προκλήσεις στη διαχείριση των Δεδομένων Μεγάλης Κλίμακας

Η διαχείριση των δεδομένων μεγάλης κλίμακας αποτελεί πρόκληση τόσο για τα συστήματα και τις εφαρμογές όσο και για τους ανθρώπους που ασχολούνται με αυτήν. Προκλήσεις για τα δεδομένα μεγάλης κλίμακας αποτελούν η συλλογή τους, η αποθήκευση τους, η ανάλυσή τους, η αναζήτηση σε αυτά, η μεταφορά τους, η οπτικοποίησή τους, η ενημέρωσή τους, η διατήρηση της ιδιωτικότητας της πληροφορίας κ.α. [4] Ενδεικτικά περιγράφουμε τρία παραδείγματα:

- **Διαχείριση του όγκου των δεδομένων**: Ο όγκος των δεδομένων συνεχώς αυξάνεται με την ανάγκη για αποθήκευσή τους να δημιουργεί τεράστιες απαιτήσεις σε αποθηκευτικούς πόρους. Μελετώνται συνεπώς τεχνικές έτσι ώστε τα δεδομένα να καταλαμβάνουν τον ελάχιστο δυνατό χώρο. Τέτοια παραδείγματα αποτελούν οι αλγόριθμοι συμπίεσης που χρησιμοποιούνται και η αφαίρεση διπλότυπων εγγραφών.
- **Αναπαράσταση των δεδομένων**: Πολλά σύνολα δεδομένων παρουσιάζουν έντονες διαφοροποιήσεις ως προς το είδος, τη δομή και τη σημασιολογία. Η ανάλυση αυτών των δεδομένων απαιτεί την κατάλληλη αναπαράστασή τους κάτι που δεν είναι τετριμμένο δεδομένης της ετερογένειας που εμφανίζουν.
- **Αναγνώριση της εγκυρότητας των δεδομένων**: Ο τεράστιος όγκος δεδομένων μπορεί παράλληλα να περιλαμβάνει δεδομένα παραπλανητικά έχοντας ως συνέπεια να αλλοιώνεται το αποτέλεσμα της ανάλυσης.

2.2 Επεξεργασία στατικών δεδομένων και ρεύματος δεδομένων

Η επεξεργασία στατικών δεδομένων ή δεδομένων σε δεσμίδες, γνωστή ως *batch processing*, αφορά δεδομένα που βρίσκονται στο σύνολό τους αποθηκευμένα πριν την έναρξη της επεξεργασίας τους.

Η επεξεργασία ρεύματος δεδομένων, γνωστή ως *stream processing*, απαιτεί τη συνεχή εκτέλεση των επιθυμητών εργασιών πάνω σε μία ροή εισόδου δεδομένων σε πραγματικό χρόνο, με τα αποτελέσματα της εργασίας να προκύπτουν επίσης σε πραγματικό *(real-time processing)* ή σχεδόν πραγματικό χρόνο *(near-real-time processing)*. Στην επεξεργασία ρεύματος δεν απαιτείται η αποθήκευση των δεδομένων πριν την επεξεργασία. Η επεξεργασία ρεύματος δεδομένων προορίζεται για ένα πλήθος σεναρίων χρήσης που έχουν απαίτηση για λήψη αποφάσεων σε πραγματικό χρόνο. Τυπικά παραδείγματα αποτελούν η ανίχνευση εισβολής στο δικτυακό σύστημα ενός οργανισμού και η παρακολούθηση της κίνησης στο οδικό δίκτυο.

Μπορούμε να διακρίνουμε δύο τρόπους εκτέλεσης της επεξεργασίας πάνω σε ρεύμα δεδομένων.

- Συνεχόμενη επεξεργασία: Τα δεδομένα που εισέρχονται επεξεργάζονται αμέσως.
- Micro-batch επεξεργασία: Με τη συγκεκριμένη μέθοδος συλλέγεται επαναληπτικά ένα μικρό σύνολο δεδομένων και η επεξεργασία πραγματοποιείται πάνω σε αυτό.

2.3 Συστοιχία υπολογιστικών μηχανών

Με τον όρο computer cluster αναφερόμαστε σε μία συστοιχία υπολογιστικών μηχανών που συνδέονται μεταξύ τους μέσω του δικτύου και χρησιμοποιούνται σαν μία ενιαία λογική μονάδα για την εκτέλεση εργασιών. Η χρήση τους ενδείκνυται για εφαρμογές που έχουν απαιτήσεις για υψηλή διαθεσιμότητα ή για υψηλή απόδοση. [5] Στην παρούσα εργασία θα χρησιμοποιήσουμε το μοντέλο του cluster computing για την κατανομή του φόρτου εργασίας με στόχο την καλύτερη απόδοση.

Η χρήση του cluster computing ενδείκνυται για τις υψηλές απαιτήσεις που έχει η επεξεργασία δεδομένων μεγάλης κλίμακας.

2.4 Διασυνδεδεμένα και αυτοματοποιημένα οχήματα

Με τον όρο διασυνδεδεμένα οχήματα αναφερόμαστε στα οχήματα τα οποία μέσω εφαρμογών, υπηρεσιών και τεχνολογιών έχουν τη δυνατότητα επικοινωνίας με άλλες οντότητες.[6] Ενδεικτικές είναι οι περιπτώσεις οχημάτων με εφαρμογές ψυχαγωγίας (infotainment systems) που συνδέονται με έξυπνες κινητές συσκευές εντός του οχήματος ή οχημάτων με την υπηρεσία eCall η οποία παρέχει τη δυνατότητα για αυτόματη κλήση έκτακτης ανάγκης βάσει του αριθμού 112 σε περίπτωση ατυχήματος. [7]

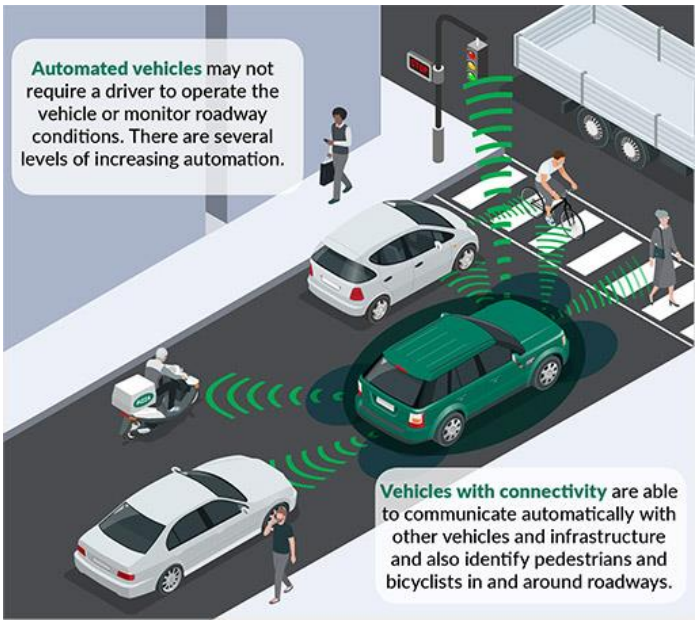
Ωστόσο, αυτή είναι μόνο μία όψη των δυνατοτήτων των διασυνδεδεμένων οχημάτων. Συγκεκριμένα η εξέλιξη των τεχνολογιών στον ευρύτερο κλάδο των υπολογιστικών συστημάτων, των δικτύων επικοινωνιών, του διαδικτύου των αντικειμένων αλλά και η ανάπτυξη κατάλληλων πρωτοκόλλων επικοινωνίας αξιοποιήθηκε από τον κλάδο της αυτοκίνησης και έδωσε τη δυνατότητα στα οχήματα να επικοινωνούν με άλλα οχήματα, με την περιβάλλουσα υποδομή και με άλλες οντότητες, προσδίδοντας έτσι στα οχήματα την έννοια της συνεργατικότητας.

Η αλληλεπίδραση των οχημάτων με άλλες οντότητες εμπίπτει στο πεδίο των Συνεργατικών και Ευφυών Συστημάτων Μεταφορών (C-ITS ,Cooperative Intelligent Transport Systems).

Με τον όρο αυτοματοποιημένο χαρακτηρίζουμε ένα όχημα που αντιλαμβάνεται τις συνθήκες του περιβάλλοντος και μπορεί να κινηθεί ασφαλώς, είτε με ελάχιστη ή καθόλου ανθρώπινη παρέμβαση.[8] Συχνά, η έννοια του αυτοματοποιημένου οχήματος αναφέρεται σε οχήματα που κινούνται χωρίς καμία ανθρώπινη παρέμβαση, ωστόσο δεν υπάρχει κάποιος σαφής και κοινά αποδεκτός ορισμός. Για τον ακριβέστερο προσδιορισμό της έννοιας των αυτοματοποιημένων οχημάτων ορίστηκαν από τον οργανισμό SAE (Society of Automotive Engineers) και χρησιμοποιούνται συχνά τα παρακάτω επίπεδα αυτοματοποίησης [9].

- Επίπεδο 0 - No Automation. Η οδήγηση του οχήματος εκτελείται αποκλειστικά από άνθρωπο.
- Επίπεδο 1 - Driver Assistance. Σε ορισμένες περιπτώσεις, ο έλεγχος της διεύθυνσης του οχήματος ή η πέδηση/επιτάχυνσή του μπορούν να εκτελεστούν από συστήματα υποβοήθησης του οδηγού. Τυπικό παράδειγμα αποτελεί η διατήρηση του οχήματος εντός της λωρίδας κυκλοφορίας.

- Επίπεδο 2 - Partial Automation. Ο έλεγχος της διεύθυνσης του οχήματος και η επιτάχυνσή/ πέδησή του μπορούν να εκτελεστούν από συστήματα υποβοήθησης του οχήματος σε ορισμένες περιστάσεις.
- Επίπεδο 3 - Conditional Automation. Σε ορισμένες περιστάσεις, το αυτοματοποιημένο σύστημα οδήγησης του οχήματος μπορεί να χειριστεί αποκλειστικά το όχημα. Απαραίτητη είναι όμως η επαναφορά του οχήματος στον έλεγχο του οδηγού (ανθρώπου) σε περίπτωση που το σύστημα το απαιτήσει.
- Επίπεδο 4 - High Automation. Το όχημα μπορεί να κινηθεί αποκλειστικά από το αυτοματοποιημένο σύστημα οδήγησης που διαθέτει σε ορισμένες συνθήκες, ακόμα και στην περίπτωση μη συμμετοχής του οδηγού (ανθρώπου) έπειτα από υπόδειξη του συστήματος.
- Επίπεδο 5 – Full Automation. Η οδήγηση του οχήματος μπορεί να εκτελεστεί από το αυτοματοποιημένο σύστημα οδήγησης χωρίς καμία ανθρώπινη παρέμβαση και υπό οποιεσδήποτε περιστάσεις/συνθήκες.



Εικόνα 2: Αυτοματοποιημένα και διασυνδεδεμένα οχήματα – Ενδεικτική απεικόνιση [10]

2.5 Αξιοποίηση δεδομένων από διασυνδεδεμένα οχήματα

Τα συνεργατικά, συνδεδεμένα και αυτοματοποιημένα οχήματα καθιστούν τον τομέα της αυτοκίνησης ως μία από τις μεγαλύτερες πηγές παραγωγής δεδομένων. Είναι εύκολο να αναλογιστεί κανείς τον όγκο των δεδομένων αλλά και το ρυθμό που θα παράγονται με την πάροδο του χρόνου και την αντικατάσταση παλαιότερων οχημάτων με νέα που θα υποστηρίζουν τη διασυνδεσιμότητα.

Εφαρμογές πάνω στα δεδομένα μεγάλης κλίμακας των οχημάτων αυτών επιτρέπουν στους κατασκευαστές, στους προμηθευτές, και σε άλλους σχετικούς παρόχους υπηρεσιών να αξιοποιήσουν την εξαγόμενη πληροφορία για τη δημιουργία σύγχρονων και καινοτόμων υπηρεσιών κινητικότητας**.** Επίσης, στο σύνολο των δεδομένων που αξιοποιούνται δύναται να ενταχθούν δεδομένα από εξωτερικές πηγές όπως οι υποδομές μεταφορών, τα κοινωνικά δίκτυα

και άλλες διαδικτυακές υπηρεσίες επεκτείνοντας έτσι το εύρος των δυνητικά παρεχόμενων υπηρεσιών.

Η πρόσβαση στα δεδομένα των οχημάτων μπορεί να δώσει σε πολλές επιχειρήσεις την ευκαιρία για ανάπτυξη καινοτόμων λύσεων και μέσα από αυτές να αναπτυχθούν και οι ίδιες και να διεκδικήσουν μερίδιο στην αγορά. Παράλληλα ο τελικός χρήστης θα επωμίζεται τα οφέλη της χρήσης των υπηρεσιών αυτών. Ιδιαίτερη μνεία αξίζουν οι εφαρμογές που προορίζονται για την ενίσχυση της ασφάλειας της κυκλοφορίας και τη μείωση των ατυχημάτων. Άλλες υπηρεσίες μπορεί να προορίζονται επίσης και για ψυχαγωγία, για απομακρυσμένο διαγνωστικό έλεγχο, για ανεύρεση θέσης παρκινγκ, για επίτευξη οικονομικής οδήγησης κ.α.

Τα δεδομένα των οχημάτων ανήκουν στην κατηγορία των ευαίσθητων προσωπικών δεδομένων και επομένως η διαχείριση τους απαιτεί πλήρη συμμόρφωση με τον Γενικό Κανονισμό Προστασίας Δεδομένων (GDPR). [11] Ο χαρακτηρισμός αυτός προκύπτει καθώς δεδομένης της πρόσβασης στα δεδομένα αυτά δύναται να προκύψει η ταυτότητα των χρηστών.

3 Μελέτη περίπτωσης: Δεδομένα Διασυνδεδεμένων Οχημάτων

3.1 Υφιστάμενη κατάσταση στη διαχείριση δεδομένων των οχημάτων

Στην παρούσα φάση, οι κατασκευαστές των οχημάτων έχουν ήδη αναπτύξει υπηρεσίες βασισμένες στα δεδομένα των οχημάτων στα οποία έχουν πρόσβαση. Επίσης, οι κατασκευαστές πρωτότυπου εξοπλισμού (OEMs) με τις ψηφιακές πλατφόρμες που διαθέτουν είτε στο όχημα είτε σε απομακρυσμένους εξυπηρετητές έχουν σημαντικό πλεονέκτημα στην παρακολούθηση των νέων διασυνδεδεμένων και αυτοματοποιημένων οχημάτων, γεγονός που τους επιτρέπει τη συλλογή των δεδομένων των οχημάτων και την αξιοποίησή τους για παροχή καινοτόμων υπηρεσιών.

Το γεγονός ότι μεμονωμένες ομάδες έχουν αποκλειστικά πρόσβαση στα δεδομένα των οχημάτων, δημιουργεί στρέβλωση του ανταγωνισμού αφού οι υπόλοιποι δυνητικά ενδιαφερόμενοι φορείς δεν αντιμετωπίζονται με τους ίδιους όρους.

3.1.1 Δράσεις Ευρωπαϊκής Επιτροπής

Η Ευρωπαϊκή Επιτροπή για να διασφαλίσει μία συντονισμένη προσέγγιση στην αντιμετώπιση των προκλήσεων στην αυτοκινητοβιομηχανία στην Ευρώπη σύστησε μία ομάδα (“High-Level Group”), ‘GEAR 2030’ τον Οκτώβριο του έτους 2015.[12] Επίσης άνοιξε διάλογο μεταξύ όλων των ενδιαφερόμενων μέσω της πλατφόρμας για την υλοποίηση των Συνεργατικών και Ευφυών Συστημάτων Μεταφοράς (C-ITS Platform).[13] Η ομάδα Working Group 6 της πλατφόρμας C-ITS εξέτασε τους πιθανούς τρόπους πρόσβασης στα δεδομένα των οχημάτων προς αξιοποίησή τους και παροχή υπηρεσιών στο κοινό.

Η ανάγκη για δίκαιο ανταγωνισμό εκφράστηκε από την Ευρωπαϊκή Επιτροπή και η Ένωση των Κατασκευαστών Οχημάτων στην Ευρώπη (ACEA) πρότεινε την ιδέα του “Extended Vehicle” τον Απρίλιο του 2016 [14] και λίγο αργότερα την λύση του “Neutral Server” [15] για

να ευθυγραμμιστεί με τις υποδείξεις αυτές. Την αντίθεση τους με την πρόταση αυτή εξέφρασαν η Ευρωπαϊκή Ένωση Ανεξάρτητων Διανομέων Ανταλλακτικών και Αντιπροσώπων ("FIGIEFA") και το Ευρωπαϊκό Συμβούλιο για το Εμπόριο και την Επισκευή οχημάτων ("CECRA") καθώς και άλλες ενώσεις σχετικές με τον χώρο, ισχυριζόμενοι ότι η πρόταση της ACEA θέτει σε κίνδυνο τον ελεύθερο ανταγωνισμό, περιορίζει την καινοτομία και τις επιλογές του κοινού. Στην κατεύθυνση αυτή εμπόδισαν τις προσπάθειες των κατασκευαστών να κατοχυρωθεί η πρότασή τους, βάσει της Ευρωπαϊκής νομοθεσίας. [16]

Η Ευρωπαϊκή Επιτροπή το Μάϊο του 2017 ανακοίνωσε μελέτη που έγινε για λογαριασμό της, σχετικά με την πρόσβαση στα δεδομένα και τους πόρους των οχημάτων. Οι κύριοι στόχοι της παρούσας μελέτης ήταν:

- Να παράσχει περαιτέρω καθοδήγηση σχετικά με τις κατάλληλες δράσεις, ιδίως για τον προσδιορισμό και την ποσοτικοποίηση των νομικών ζητημάτων που αφορούν την πρόσβαση στα δεδομένα και τους πόρους επί οχημάτων και την αξιολόγηση των διαφόρων τεχνικών λύσεων που πρότεινε η ομάδα εργασίας 6 (WG 6) της πλατφόρμας C-ITS,
- Να διεξάγει ανάλυση κόστους-οφέλους των άμεσων και έμμεσων οικονομικών, κοινωνικών και περιβαλλοντικών επιπτώσεων και
- Να αναπτύξει και να αναλύσει ένα σύνολο σεναρίων λαμβάνοντας υπόψη την εξέλιξη της αγοράς, τις ισχύουσες κοινοτικές, εθνικές και διεθνείς νομοθεσίες και τις εργασίες της ομάδας WG 6 της πλατφόρμας C-ITS.[17]

Στη μελέτη αυτή, προτείνεται σαν βραχυπρόθεσμη λύση η έννοια του *Κοινόχρηστου Επεξεργαστή,* "Shared Server", καθώς αφενός διατηρεί την ασφάλεια των οχημάτων και δεν επιβαρύνει σημαντικά την αυτοκινητοβιομηχανία και αφετέρου παρέχει με την προσθήκη παρεμβάσεων σε ευρωπαϊκό επίπεδο, χαρακτηριστικά τα οποία είναι ευθυγραμμισμένα με την παροχή δικαιότερου ανταγωνισμού σε σχέση με τη λύση Extended Vehicle / Neutral Server.

Η Ευρωπαϊκή Επιτροπή στις 17 Μαΐου του 2018 τοποθετήθηκε μέσω ανακοίνωσης COM(2018) 283 [18] "On the road to automated mobility: An EU strategy for mobility of the future". Στην ανακοίνωση αυτή γίνεται παραπομπή στην μελέτη "Access to In-vehicle Data and Resources" και αναφέρει ότι προκύπτουν ενδείξεις ότι η κεντρική διάθεση των δεδομένων των οχημάτων μέσω της λύσης "extended vehicle data platform servers" δεν διασφαλίζει από μόνη της ένα περιβάλλον δίκαιου και υγιούς ανταγωνισμού μεταξύ των διάφορων παρόχων υπηρεσιών. Στην κατεύθυνση αυτή θα συνεχίσει να παρακολουθεί τις εξελίξεις και να εντοπίζει όλες τις απαραίτητες ενέργειες που απαιτούνται ώστε να δημιουργηθεί το πλαίσιο που απαιτείται για την κοινή χρήση των δεδομένων των οχημάτων σύμφωνα πάντα με την νομοθεσία και με απόλυτο σεβασμό στην προστασία των προσωπικών δεδομένων.[17]

3.2 Ένα πραγματικό πρόβλημα: Υπόθεση εργασίας του Κοινόχρηστου Επεξεργαστή (Shared Server Concept)

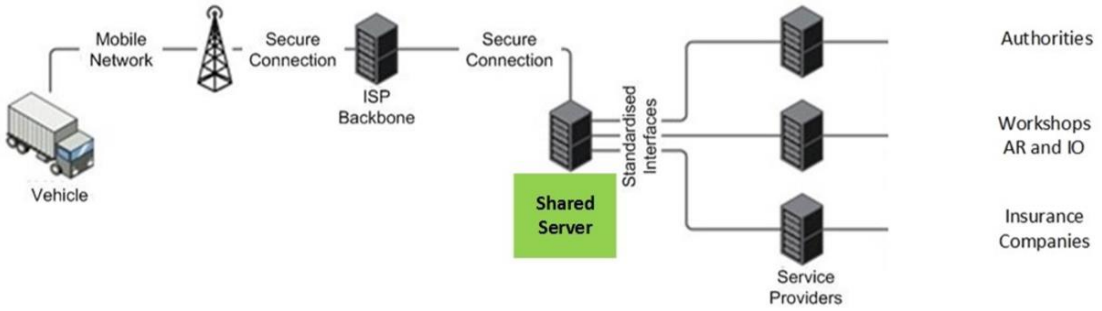
Στη μελέτη της Ευρωπαϊκής Επιτροπής "Access to in-vehicle data and resources" αναλύθηκαν τρείς αρχιτεκτονικές:

- Αρχιτεκτονική “Data Server Platform” : Βασίζεται στην ιδέα της αποστολής των δεδομένων από τα οχήματα σε έναν εξυπηρετητή. Μέσω του εξυπηρετητή, τα δεδομένα θα γίνονται διαθέσιμα προς κατανάλωση για τις διάφορες υπηρεσίες. Σε αυτή την αρχιτεκτονική τόσο τα δεδομένα όσο και οι υπηρεσίες που τα καταναλώνουν βρίσκονται εκτός του οχήματος.
- Αρχιτεκτονική “In-vehicle interface”: Βασίζεται στη διάθεση των δεδομένων μέσω μίας διεπαφής που βρίσκεται στο αυτοκίνητο ενώ οι υπηρεσίες που τα καταναλώνουν παραμένουν εκτός του αυτοκινήτου.
- Αρχιτεκτονική “On-board Application Platform”: Βασίζεται στη χρήση μίας πλατφόρμας στο περιβάλλον του οχήματος η οποία θα επιτρέπει την πρόσβαση στα δεδομένα αλλά και την εκτέλεση των εφαρμογών-υπηρεσιών εντός του περιβάλλοντος του οχήματος.[17]

Ο Κοινόχρηστος Επεξεργαστής (Shared Server), που προτάθηκε ως βραχυπρόθεσμη λύση σε αυτή τη μελέτη, αποτελεί μία εκ των τεχνικών λύσεων που προορίζεται για τη διαχείριση της πρόσβασης στα δεδομένα των οχημάτων, και υπάγεται στην αρχιτεκτονική “Data Server Platform”.

Στη λύση του Κοινόχρηστου Επεξεργαστή τα δεδομένα του οχήματος αποστέλλονται μέσω του δικτύου τηλεπικοινωνιών στον πάροχο υπηρεσιών διαδικτύου (ISP) και ύστερα μέσω ασφαλούς σύνδεσης στον Κοινόχρηστο Επεξεργαστή από τον οποίο μέσω διεπαφής, η οποία θα πληροί συγκεκριμένες απαιτήσεις, γίνονται διαθέσιμα στους ενδιαφερόμενους φορείς και παρόχους υπηρεσιών. Απαραίτητη προϋπόθεση είναι ο Κοινόχρηστος Εξυπηρετητής να διαχειρίζεται από ουδέτερο φορέα, ο οποίος θα είναι και θα παραμείνει υπό την καθοδήγηση της ένωσης των εμπλεκόμενων και ενδιαφερόμενων μερών [17].

Η προϋπόθεση αυτή είναι ουσιαστικά που διαχωρίζει το μοντέλο του Κοινόχρηστου Επεξεργαστή από το μοντέλο του «Εκτεταμένου Οχήματος» που προτείνουν οι OEMs και στο οποίο ο εξυπηρετητής που παρέχει τα δεδομένα θα ήταν υπό τη διαχείρισή τους. Επιπροσθέτως, η λύση του Κοινόχρηστου Επεξεργαστή απαιτεί τα δεδομένα που παρέχονται μέσω της διεπαφής να είναι της ίδιας ποιότητας με τα δεδομένα που λαμβάνει ο εξυπηρετητής των OEMs. Μία υψηλού επιπέδου άποψη της λύσης αυτής φαίνεται στην παρακάτω εικόνα.



Εικόνα 3: Αρχιτεκτονική Shared Server [17]

3.3 Απαιτήσεις περιβάλλοντος σε θεσμικό και τεχνολογικό επίπεδο

Η προτεινόμενη λύση του κοινόχρηστου επεξεργαστή συμμορφώνεται σε θεσμικό και τεχνολογικό επίπεδο με τις σχετικές απαιτήσεις όπως αυτές αποτυπώθηκαν στις

κατευθυντήριες αρχές της μελέτης «Access to in-vehicle data and resources». Παρακάτω περιγράφονται συνοπτικά οι σχετικές κατευθυντήριες.

3.3.1 Συγκατάθεση χρήστη

Ο χρήστης που φέρει την κυριότητα των δεδομένων, είτε λόγω ιδιοκτησίας είτε λόγω χρήσης του οχήματος, αποφασίζει αν και σε ποιον θα παρέχει τα δεδομένα και για ποιο σκοπό. Ο σκοπός χρήσης των δεδομένων αντικατοπτρίζει την επιθυμητή υπηρεσία. Επίσης, πρέπει να έχει τη δυνατότητα μη συμμετοχής στην παροχή των δεδομένων αλλά και τη δυνατότητα να ανακαλέσει τη συγκατάθεσή του οποιαδήποτε στιγμή.

Η ικανοποίηση αυτής της απαίτησης συνδέεται με τη συλλογή της συγκατάθεσης του χρήστη. Ο χρήστης μπορεί να δηλώνει τις επιλογές του μέσα από την διεπαφή χρήστη-μηχανής HMI του οχήματος, που του παρέχει ο κατασκευαστής. Ο μηχανισμός αυτός είναι όμως διαθέσιμος αποκλειστικά για τους κατασκευαστές των οχημάτων, με τους υπόλοιπους ενδιαφερόμενους να μην έχουν τη δυνατότητα αυτή. Για να μπορούν να έχουν και οι υπόλοιποι ενδιαφερόμενοι πρόσβαση στη συγκατάθεση του χρήστη θα πρέπει να χρησιμοποιούν μία εφαρμογή που θα εκτελείται σε διαφορετική συσκευή. Με αυτόν τον τρόπο όλοι οι ενδιαφερόμενοι δύναται να έχουν πρόσβαση στη συγκατάθεση του χρήστη. Ωστόσο η δυνατότητα που έχουν οι κατασκευαστές, δείχνει να είναι φιλικότερη προς το χρήστη και να του παρέχει ευκολία στη δήλωση της συγκατάθεσής του η οποία φέρνει το μηχανισμό αυτόν σε πλεονεκτικότερη θέση έναντι μιας λύσης που θα περιλαμβάνει επιπλέον εφαρμογή. [17]

3.3.2 Δίκαιος ανταγωνισμός

Στην προηγούμενη υπο-ενότητα διαπιστώθηκε ότι η πρόσβαση στη συγκατάθεση των χρηστών δεν γίνεται επί ίσοις όροις για όλα τα ενδιαφερόμενα μέρη. Συνεπώς, θα πρέπει να διασφαλιστεί ότι τα δεδομένα θα παρέχονται σε όλους τους ενδιαφερόμενους με την ίδια καθυστέρηση και θα έχουν την ίδια ποιότητα. Η πραγματοποίηση αυτής της απαίτησης, πιθανώς απαιτεί την κοινή χρήση των δεδομένων ή την αποστολή των δεδομένων μέσω ξεχωριστής σύνδεσης από το όχημα στον Κοινόχρηστο Επεξεργαστή. [17]

3.3.3 Προστασία δεδομένων

Τα δεδομένα των οχημάτων αποτελούν προσωπικά δεδομένα και επομένως πρέπει να προστατεύονται. Προκύπτει η ανάγκη για ελεγκτές οι οποίοι θα πρέπει εκ σχεδιασμού να εξασφαλίζουν την προστασία των δεδομένων. Ο Κοινόχρηστος Επεξεργαστής, ως ένας ελεγκτής, αποφασίζει με βάσει τις οδηγίες του χρήστη για το πώς και για ποιο σκοπό θα επεξεργαστεί τα δεδομένα του.

3.3.4 Ασφάλεια των οχημάτων

Οποιαδήποτε λύση παρέχεται σχετικά με την πρόσβαση στα δεδομένα και τους πόρους των οχημάτων έχει απαραίτητη προϋπόθεση να μη θέτει σε κίνδυνο την ασφαλή λειτουργία του οχήματος, να μην επηρεάζει καμία από τις λειτουργίες του και να μη θέτει σε κίνδυνο την αξιοπιστία του κατασκευαστή.

Στην κατεύθυνση αυτή η λύση του Κοινόχρηστου Επεξεργαστή αποτρέπει την πρόσβαση τρίτων σε δεδομένα και πόρους του οχήματος, παρέχοντας έτσι μια ασφαλή λύση και εφικτή σε σύντομο χρονικό διάστημα. Σημειώνεται, ωστόσο, ότι πιθανόν να απαιτείται αναθεώρηση των κανόνων και της αρχιτεκτονικής των συστημάτων ασφαλείας των οχημάτων για να αντιμετωπιστούν ορισμένες αδυναμίες που συναντώνται. [17]

4 Διαχείριση της συγκατάθεσης των υποκειμένων (χρηστών)

Όπως παρουσιάστηκε στο 3ο κεφάλαιο, η υφιστάμενη προτεινόμενη λύση για διαχείριση των δεδομένων των οχημάτων είναι αυτή του Κοινόχρηστου Επεξεργαστή. Με βάση αυτή την αρχιτεκτονική, στην παρούσα διπλωματική γίνεται προσπάθεια σχεδιασμού και μοντελοποίησης μιας λύσης η οποία θα έχει στόχο να διαχειριστεί τη συγκατάθεση των χρηστών σε πραγματικό χρόνο για επεξεργασία των δεδομένων που παράγουν και αποστέλλουν τα οχήματά τους από τους διαφορετικούς διαθέσιμους παρόχους υπηρεσιών.

4.1 Προτεινόμενη λύση

4.1.1 Αντικείμενο υλοποίησης

Αντικείμενο της υλοποίησης είναι ο σχεδιασμός λογισμικού που θα επιτρέπει την εφαρμογή ελέγχου πρόσβασης πάνω στα ρεύματα εισόδου των δεδομένων που φτάνουν στον Κοινόχρηστο Επεξεργαστή βάσει της συγκατάθεσης του χρήστη.

Συγκεκριμένα, αφού τα δεδομένα που αποστέλλει το όχημα παραληφθούν στον Κοινόχρηστο Επεξεργαστή, αντικείμενο της υλοποίησης είναι η επεξεργασία τους σύμφωνα με τη συγκατάθεση του χρήστη, έτσι ώστε οι πάροχοι δεδομένων να βλέπουν μόνο τα δεδομένα για τα οποία ο χρήστης έχει δώσει συγκατάθεση, και όχι το σύνολο των δεδομένων.

Κατά αυτόν τον τρόπο, και με την εφαρμογή ελέγχου πρόσβασης θα επιτυγχάνεται η προστασία του απορρήτου των δεδομένων.

Καθώς ο απώτερος σκοπός του συνολικού συστήματος είναι η αξιοποίηση των δεδομένων για παροχή υπηρεσιών, ακόμα και με χρόνους απόκρισης στο φάσμα της σχεδόν πραγματικής υπηρεσίας, ο χρόνος επεξεργασίας των δεδομένων αποτελεί κρίσιμο σημείο προς ελαχιστοποίηση.

4.1.2 Παραδοχές υλοποίησης

Στο πλαίσιο σχεδιασμού της λύσης για τη διαχείριση της συγκατάθεσης των χρηστών, γίνεται η παραδοχή ότι θα υπάρχει απαραίτητη διεπαφή, ώστε ο χρήστης να έχει προηγουμένως ορίσει ποια δεδομένα δέχεται να διαθέσει και τις υπηρεσίες που επιτρέπει να διαμοιραστούν αυτά. Οι πληροφορίες αυτές αποτελούν δεδομένο εισόδου για την υλοποίησή μας.

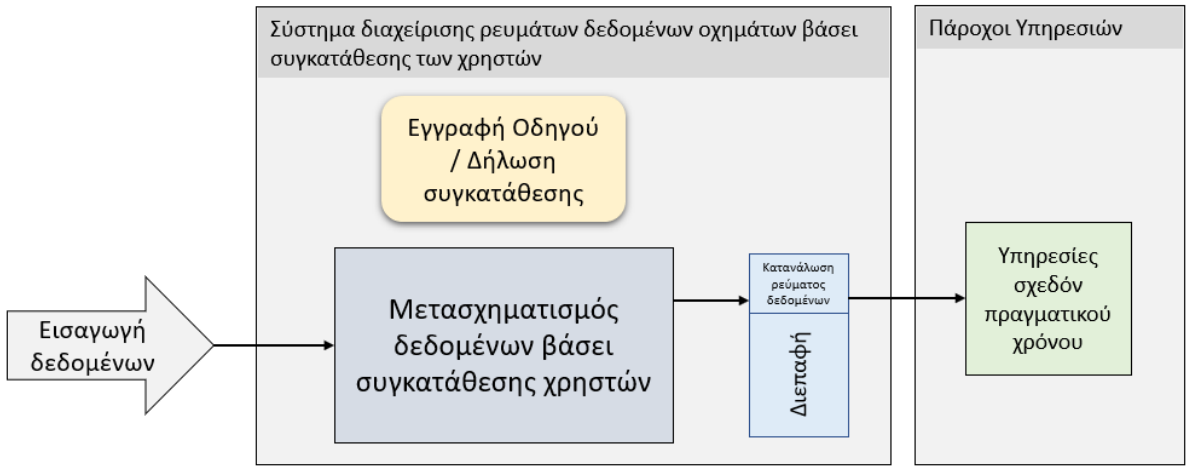
Συγκεκριμένα, γίνεται η παραδοχή ότι οι πάροχοι υπηρεσιών ενημερώνουν το χρήστη για το σύνολο των προσφερόμενων υπηρεσιών καθώς και την εξάρτηση καθεμιάς από αυτές τις υπηρεσίες με τις συγκεκριμένες κατηγορίες δεδομένων που παράγονται από το όχημα.

Στη συνέχεια ο χρήστης επιλέγει ποιες από αυτές θα χρησιμοποιήσει και παράλληλα συναινεί για την παροχή των σχετικών δεδομένων. Επίσης, θα πρέπει να έχει τη δυνατότητα, εφόσον μια υπηρεσία παρέχεται από πολλαπλούς παρόχους, να ορίζει με ποιον / ποιους επιθυμεί να συνεργαστεί.

Κατά αυτόν τον τρόπο θα ορίζεται για κάθε κατηγορία διαθέσιμων δεδομένων η άδεια του χρήστη για επεξεργασία καθώς και οι επιτρεπόμενοι αποδέκτες (πάροχοι).

4.1.3 Αρχιτεκτονική προτεινόμενου συστήματος σε επίπεδο υπηρεσιών

Στην Εικόνα 4 παρουσιάζεται η αρχιτεκτονική του συστήματος σε επίπεδο υπηρεσιών. Στη συνέχεια ακολουθεί επεξήγηση των επιμέρους τμημάτων.



Εικόνα 4: Συστατικά αρχιτεκτονικής προτεινόμενου συστήματος σε επίπεδο υπηρεσιών

Εισαγωγή Δεδομένων

Αρχικά τα δεδομένα που αποστέλλουν τα οχήματα φθάνουν στο σύστημα. Στο σημείο αυτό παρατηρούμε ότι τα δεδομένα από διαφορετικούς κατασκευαστές μπορεί να έχουν διαφορετικό σχήμα και επομένως είναι απαραίτητο να έχει συμφωνηθεί ένα ελάχιστο σύνολο δεδομένων που θα είναι κοινό.

Για τη μετάδοση των δεδομένων απαιτείται επίσης να συμφωνηθεί και ποιος θα είναι ο μορφότυπός τους. Προτείνεται η χρήση του μορφότυπου JSON (JavaScript Object Notation) καθώς είναι ευρέως αποδεκτός, χρησιμοποιείται συχνά για ανταλλαγή δεδομένων μεταξύ δικτυακών συστημάτων και η ανάλυση του γίνεται εύκολα και γρήγορα.

Εγγραφή Οδηγού / Δήλωση συγκατάθεσης

Ο χρήστης-οδηγός εγγράφεται στο σύστημα και με το λογαριασμό του δηλώνει και επεξεργάζεται τις επιλογές του για συγκατάθεση χρήσης των δεδομένων του οχήματός του από τους διαφορετικούς παρόχους υπηρεσιών.

Μετασχηματισμός δεδομένων βάσει συγκατάθεσης

Στο σημείο αυτό γίνεται η επεξεργασία των δεδομένων σύμφωνα με τη συγκατάθεση του οδηγού-χρήστη. Κατά αυτόν τον τρόπο μόνο τα επιλεγμένα δεδομένα αποστέλλονται στους παρόχους της προτίμησης του χρήστη.

Διεπαφή

Η διεπαφή του συστήματος είναι προσβάσιμη στους παρόχους υπηρεσιών και χρησιμοποιείται ώστε οι πάροχοι υπηρεσιών να αποκτούν πρόσβαση στο ρεύμα δεδομένων που τους αντιστοιχεί βάσει του μετασχηματισμού που προηγείται.

5 Μηχανισμοί Ελέγχου Πρόσβασης

Η κεντρική διεργασία του συστήματος που περιγράφηκε στο 4ο κεφάλαιο είναι ο μετασχηματισμός των δεδομένων βάσει της συγκατάθεσης των υποκειμένων (χρηστών). Το πρόβλημα αυτό ανάγεται στην επίτρεψη ή όχι της πρόσβασης των παρόχων υπηρεσιών πάνω στο ρεύμα δεδομένων. Προκύπτει λοιπόν η ανάγκη να εφαρμόσουμε ένα μηχανισμό ελέγχου πρόσβασης. Για το λόγο αυτό θα μελετήσουμε δύο από τους κυριότερους μηχανισμούς ελέγχου πρόσβασης που χρησιμοποιούνται.

5.1.1 RBAC (Role Based Access Control)

RBAC (Role-based access control) είναι το μοντέλο που χρησιμοποιείται για τον περιορισμό της πρόσβασης πάνω στους επιθυμητούς πόρους μόνο στα άτομα που έχουν εξουσιοδότηση η οποία δηλώνεται μέσω των ρόλων τους. Σε έναν οργανισμό στον οποίον η πρόσβαση ελέγχεται μέσω RBAC, τα δικαιώματα (permissions) εκχωρούνται σε συγκεκριμένους ρόλους. Κατά αυτόν τον τρόπο οι χρήστες ενός συστήματος αποκτούν δικαιώματα πρόσβασης μέσω της ανάθεσης ρόλων σε αυτούς. Παρατηρούμε λοιπόν ότι ένας ρόλος αντιστοιχίζεται σε ένα σύνολο δικαιωμάτων πρόσβασης. Σύμφωνα με το πρότυπο NIST/ANSI/INCITS RBAC standard (2004) μπορούμε να αναγνωρίσουμε τρία επίπεδα RBAC ως εξής. [19]

1. Core RBAC

Το core RBAC πληροί τα βασικά στοιχεία του RBAC δηλαδή την αντιστοίχιση χρήστη με ρόλο και την αντιστοίχιση ρόλου με δικαιώματα. Επιπλέον οι παραπάνω αναθέσεις χρήστη-ρόλου και ρόλου-δικαιωμάτων είναι σχέσεις με πληθικότητα πολλά-πολλά, εννοώντας για παράδειγμα ότι ένας ρόλος μπορεί να απονέμεται σε πολλούς χρήστες και ένας χρήστης δύναται να έχει πολλούς ρόλους. Στις απαιτήσεις του core RBAC βρίσκεται επίσης η δυνατότητα για την αναθεώρηση των παραπάνω αναθέσεων. Επιπρόσθετα, πρέπει να υποστηρίζεται η ιδέα των συνόδων (user sessions) κατά την οποία επιλεκτικά ενεργοποιούνται ή απενεργοποιούνται ρόλοι. Τέλος, προϋπόθεση αποτελεί και η δυνατότητα των χρηστών να ασκούν ταυτόχρονα τα δικαιώματα πολλαπλών ρόλων.

2. Hierarchical RBAC

Οι ρόλοι μπορεί να έχουν επικαλυπτόμενα δικαιώματα και μέσα σε έναν οργανισμό είναι σύνηθες να υπάρχει ένας αριθμός από δικαιώματα που απονέμονται σε μεγάλο αριθμό χρηστών. Έτσι, επεκτείνοντας το RBAC να υποστηρίζει την ιεραρχία ρόλων προκύπτει το Hierarchical RBAC. Η ιεραρχία ρόλων είναι μία σχέση μερικής διάταξης (partial order).

3. Constrained RBAC

Το επίπεδο αυτό εισάγει την έννοια του διαχωρισμού των καθηκόντων (SoD - Separation of Duties) έτσι ώστε να μειώνεται το ρίσκο κακόβουλης ενέργειας ή καταστροφής εξαιτίας πιθανού λάθους. Η ιδέα πίσω από το SoD είναι ότι μία κρίσιμη δραστηριότητα απαιτεί την έγκριση περισσοτέρων του ενός ατόμων για να πραγματοποιηθεί.

5.1.2 ABAC

5.1.2.1 Βασικά συστατικά

ABAC (Attribute-based access control) είναι το μοντέλο ελέγχου πρόσβασης σύμφωνα με το οποίο αποφασίζεται αν τα υποκείμενα τα οποία αιτούνται την πρόσβαση θα την λάβουν ή όχι. Για την εξαγωγή αυτής της απόφασης αξιολογούνται σύμφωνα με τους κανόνες της πολιτικής πρόσβασης τα χαρακτηριστικά του υποκειμένου, του αντικειμένου, δηλαδή του πόρου για τον οποίο αιτείται η πρόσβαση και του περιβάλλοντος. Οι οντότητες που συμμετέχουν στο ABAC περιγράφονται ως εξής: [20]

Υποκείμενο: Μπορεί να είναι είτε άνθρωπος είτε μία ψηφιακή οντότητα που δεν έχει να κάνει με άνθρωπο, όπως για παράδειγμα μία ηλεκτρονική συσκευή, και αιτείται την πρόσβαση σε λειτουργίες πάνω στο αντικείμενο (πόρο).

Αντικείμενο: Είναι ο πόρος του συστήματος στον οποίον η πρόσβαση καθορίζεται μέσω ABAC, για παράδειγμα μπορεί να είναι αρχεία, πίνακες, διεργασίες, προγράμματα, δικτυακές υποδομές ή τομείς που περιέχουν ή λαμβάνουν πληροφορίες καθώς και γενικά οτιδήποτε πάνω στο οποίο μπορεί να εφαρμοστεί μία λειτουργία από κάποιο υποκείμενο, συμπεριλαμβανομένων δεδομένων, εφαρμογών, υπηρεσιών, συσκευών και δικτύων.

Λειτουργία: Είναι η εκτέλεση μιας εργασίας κατά την αίτηση του υποκειμένου πάνω σε ένα αντικείμενο. Μπορεί να είναι ανάγνωση, εγγραφή, επεξεργασία, διαγραφή, αντιγραφή, εκτέλεση και αλλαγή.

Πολιτική πρόσβασης: Είναι η αναπαράσταση κανόνων ή σχέσεων που καθιστούν δυνατή την λήψη απόφασης για το αν η πρόσβαση στον πόρο πρέπει να επιτραπεί ή όχι, δεδομένων των χαρακτηριστικών υποκειμένου και αντικειμένου αλλά και των πιθανών συνθηκών του περιβάλλοντος.

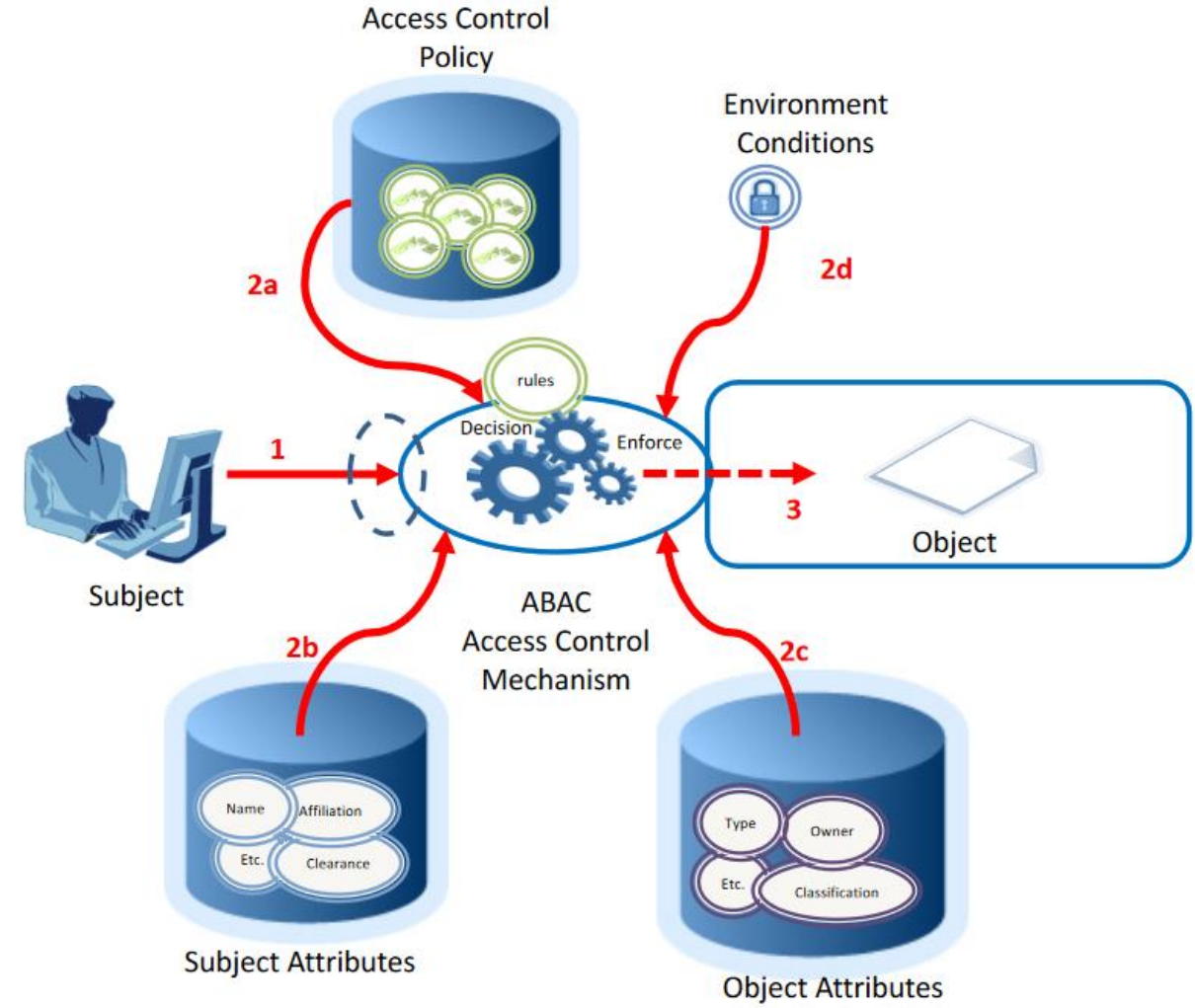
Συνθήκες περιβάλλοντος: Είναι τα ανιχνεύσιμα χαρακτηριστικά του περιβάλλοντος που είναι ανεξάρτητα του υποκειμένου και αντικειμένου.

Χαρακτηριστικά: Είναι τα στοιχεία που ελέγχονται σύμφωνα με την πολιτική πρόσβασης. Μπορούν να ταξινομηθούν στις εξής κατηγορίες.

- Χαρακτηριστικά που αφορούν το υποκείμενο, π.χ. η ηλικία, το επάγγελμα
- Χαρακτηριστικά που αφορούν την επιθυμητή δράση, π.χ. ανάγνωση, επεξεργασία, διαγραφή
- Χαρακτηριστικά που αφορούν το αντικείμενο (πόρο), π.χ. το είδος (ιατρικό έγγραφο, τραπεζικό έγγραφο), τη διαβάθμιση ασφαλείας

- Χαρακτηριστικά που αφορούν τις συνθήκες περιβάλλοντος, π.χ. η ημερομηνία, η ώρα, η τοποθεσία του υποκειμένου, η θερμοκρασία, το τρέχων επίπεδο απειλής.

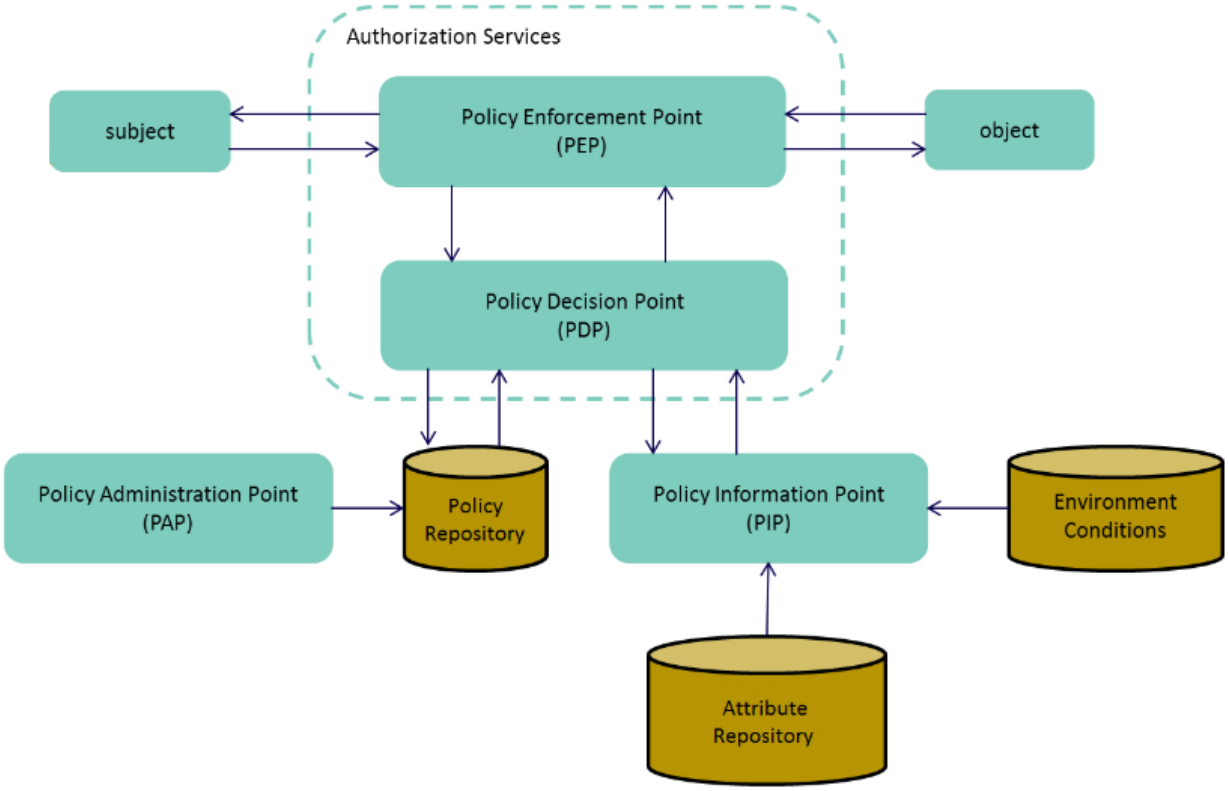
Στη συνέχεια παρατίθεται εικόνα ενδεικτική ενός σεναρίου εφαρμογής ABAC.



Εικόνα 5: Σενάριο εφαρμογής ABAC [20]

5.1.2.2 Εφαρμογή ABAC - Αρχιτεκτονική

Στην Εικόνα 6 παρουσιάζεται η λογική αρχιτεκτονική ενός συστήματος ABAC. Προτού παρουσιάσουμε τα κύρια λειτουργικά συστατικά του μηχανισμού ελέγχου πρόσβασης παραθέτουμε τους παρακάτω δύο χρήσιμους για τη συνέχεια ορισμούς. [20]



Εικόνα 6: Αρχιτεκτονική ABAC [20]

Digital Policy (DP): Το σύνολο των κανόνων ελέγχου πρόσβασης που μεταφράζονται άμεσα σε κώδικα εκτελέσιμο από υπολογιστική μηχανή. Την πολιτική και τα τμήματα δηλώσεων των κανόνων της, απαρτίζουν κυρίως τα χαρακτηριστικά των αντικειμένων, των υποκειμένων και του περιβάλλοντος.

Metapolicy (MP): Η πολιτική για τη διαχείριση των άλλων πολιτικών, όπως για παράδειγμα η ιεραρχία στην εφαρμογή των πολιτικών ή η επίλυση πιθανών συγχύσεων μεταξύ DPs και MPs.

Συστατικά μηχανισμού πρόσβασης:

Policy Decision Point (PDP): Στο στοιχείο αυτό αξιολογούνται οι σχετικές πολιτικές (DPs, MPs) ώστε να αποφασιστεί η έγκριση ή απόρριψη της πρόσβασης.

Policy Enforcement Point (PEP): Εφαρμόζει την απόφαση που εξάγεται από το PDP και είναι σύμφωνη με την πολιτική πρόσβασης δίνοντας έτσι την απάντηση στην αίτηση του υποκειμένου για πρόσβαση.

Policy Information Point (PIP): Λειτουργεί ως πηγή ανάκτησης των χαρακτηριστικών που είναι απαραίτητα ώστε να μπορέσει να γίνει η αξιολόγηση της πολιτικής πρόσβασης στο PDP και να εξαχθεί η απόφαση.

Policy Administration Point (PAP): Αποτελεί μία διεπαφή για τον χρήστη ώστε να μπορεί να δημιουργεί, να διαχειρίζεται, να δοκιμάζει τις πολιτικές (DPs, MPs) και να τις αποθηκεύει στον κατάλληλο χώρο.

5.1.3 Σύγκριση RBAC-ABAC στο σενάριο της διαχείρισης της συγκατάθεσης των υποκειμένων

Στην υλοποίηση καλούμαστε να εφαρμόσουμε μηχανισμό ελέγχου πρόσβασης πάνω στα ρεύματα των δεδομένων. Κατανοούμε ότι η απόφαση για πρόσβαση απαιτεί αξιολόγηση της συγκατάθεσης του χρήστη, και επομένως πρέπει να διαχειριζόμαστε την πρόσβαση πάνω στα ρεύματα των δεδομένων. Το μοντέλο που συμπεριλαμβάνει τη δυνατότητα αυτή είναι το ABAC. Στο RBAC οι ρόλοι είναι προκαθορισμένοι και δεν μας δίνουν τη δυνατότητα για έλεγχο πρόσβασης σε επίπεδο χαρακτηριστικού (συγκατάθεση). Συνεπώς, οι απαιτήσεις του μηχανισμού ελέγχου πρόσβασης πάνω στο ρεύμα δεδομένων είναι παρόμοιες με αυτές μιας ABAC λύσης.

5.1.4 Περιορισμοί στην εφαρμογή του ABAC

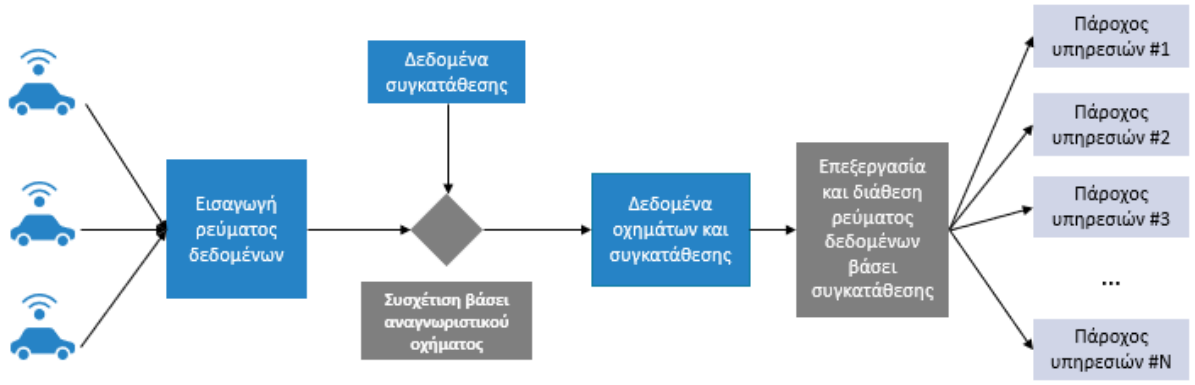
Η αποδοτική διαχείριση της εξουσιοδότησης πρόσβασης πάνω στο ρεύμα δεδομένων των οχημάτων καθιστά προτιμότερη την εφαρμογή της λογικής του μηχανισμού ABAC μέσω επεξεργασίας των δεδομένων με εργαλεία κατάλληλα για επεξεργασία ρευμάτων δεδομένων και δεδομένων μεγάλης κλίμακας .Επίσης, στο περιβάλλον της πλατφόρμας Azure, στην οποία θα στηριχθεί η υλοποίησή μας στην παρούσα διπλωματική, η υποστήριξη για μηχανισμούς ελέγχου πρόσβασης περιορίζεται στο RBAC. Συνεπώς θα εφαρμόσουμε τη λογική ιδέα του ABAC μέσω της επεξεργασίας των δεδομένων που καταφθάνουν στον Κοινόχρηστο Επεξεργαστή και θα διαχωρίσουμε το ρεύμα δεδομένων κατάλληλα σε εξόδους, ώστε σε κάθε πάροχο να αντιστοιχεί μία έξοδο. Στη συνέχεια, δύναται να απονεμηθεί για κάθε πάροχο ένας ρόλος ο οποίος θα του επιτρέπει πρόσβαση στην έξοδο που του αντιστοιχεί βάσει της πρωτύτερης ABAC επεξεργασίας.

6 Περιγραφή ροών επεξεργασίας και αλγορίθμου

6.1 Ροές επεξεργασίας

Το ρεύμα των δεδομένων που εισέρχεται στο σύστημα πρέπει να αναλυθεί συντακτικά και από αυτό να εξαχθεί το σύνολο των δεδομένων βάσει ενός προσυμφωνημένου σχήματος μεταξύ των διαφορετικών κατασκευαστών.

Στη συνέχεια, τα δεδομένα πρέπει να συσχετιστούν με την αντίστοιχη συγκατάθεση των χρηστών ώστε να πραγματοποιηθεί η επεξεργασία τους και να εφαρμοστεί ο έλεγχος πρόσβασης, αποτέλεσμα του οποίου είναι ο διαχωρισμός του εισερχόμενου ρεύματος στις διαφορετικές εξόδους, μία για κάθε πάροχο υπηρεσιών. Σχηματικά μπορούμε να αναπαραστήσουμε την ανωτέρω ροή επεξεργασίας με την Εικόνα 7.



Εικόνα 7: Ροή επεξεργασίας δεδομένων

6.2 Μοντελοποίηση προβλήματος

Για την εφαρμογή ελέγχου πρόσβασης πάνω στο ρεύμα δεδομένων χρησιμοποιείται η συγκατάθεση του κάθε χρήστη που βρίσκεται αποθηκευμένη στο σύστημά μας.

Η μοντελοποίηση του προβλήματος στηρίζεται σε ένα διαθέσιμο σετ δεδομένων τα οποία έχουν ένα σύνολο πεδίων που απεικονίζονται στον παρακάτω πίνακα μαζί με μια ενδεικτική εγγραφή (Εικόνα 8).

Time	VehicleID	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density_y
0	3	53	0	600867	5676073	95079	53	1

Εικόνα 8: Ενδεικτική εγγραφή δεδομένων

Περισσότερες πληροφορίες σχετικά με τα δεδομένα παρουσιάζονται στην ενότητα 10.1.

Τη συγκατάθεση των χρηστών ανά κατηγορία δεδομένων τη μοντελοποιούμε ως ένα διάνυσμα από bits. Κάθε bit αντιστοιχεί σε έναν πάροχο υπηρεσιών.

Βασιζόμενοι στην ανωτέρω δομή παρατηρούμε ότι προφανώς το ρόλο ενός μοναδικού αναγνωριστικού για τα διαθέσιμα δεδομένα των οχημάτων παίζει το αναγνωριστικό του οχήματος (VehicleID). Έτσι και σαν μοναδικό αναγνωριστικό της συγκατάθεσης κάθε χρήστη-

οδηγού θα χρησιμοποιηθεί το πεδίο VehicleID. Με τη χρήση του πεδίου αυτού και στα δύο σύνολα δεδομένων επιτυγχάνεται η επιθυμητή συσχέτιση του ρεύματος δεδομένων των οχημάτων με τα δεδομένα συγκατάθεσης μέσω πράξης (stream-static) join.

Επόμενη ανάγκη που προκύπτει είναι η μοντελοποίηση των επιλογών του χρήστη-οδηγού.

Υπενθυμίζεται εδώ ότι ένας χρήστης-οδηγός δύναται να επιλέξει από ένα πλήθος υπηρεσιών και από ένα πλήθος παρόχων για κάθε μία από αυτές τις υπηρεσίες. Στο σενάριο αυτό μια υπηρεσία έχει εξάρτηση από ορισμένες κατηγορίες-στήλες δεδομένων και έτσι για κάθε στήλη προκύπτει η επιλογή του χρήστη ανάμεσα στους παρόχους υπηρεσιών.

Αυτή η επιλογή ορίζει στην ουσία σε ποιους παρόχους πρέπει να επιτραπεί η πρόσβαση και σε ποιους όχι. Αυτή η λογική μας οδηγεί να δώσουμε για κάθε πάροχο υπηρεσιών από ένα bit, του οποίου η λογική τιμή "1" φανερώνει επίτρεψη πρόσβασης και η λογική τιμή "0" απόρριψη πρόσβασης.

Για επεξήγηση της μοντελοποίησης της συγκατάθεσης του χρήστη παρατίθεται ένα συνοπτικό παράδειγμα.

Time Consent	VehicleID	Speed Consent	Dist_front Consent	xCoord Consent	yCoord Consent	SectionID Consent	Avg_Sect_Speed Consent	Sect_Density_y Consent
(0,0,1)	3	(1,0,1)	(0,0,0)	(1,1,0)	(1,1,0)	(0,1,1)	(1,0,1)	(0,0,1)

Εικόνα 9: Παράδειγμα μοντελοποίησης συγκατάθεσης χρήστη (Σενάριο με 3 παρόχους)

Στο συγκεκριμένο παράδειγμα κάνουμε την παραδοχή ότι έχουμε τρεις παρόχους υπηρεσιών. Τα τρία bits της επιλογής του χρήστη για κάθε κατηγορία δεδομένων αντιστοιχίζονται με τους τρεις παρόχους υπηρεσιών ξεκινώντας από δεξιά. Συνεπώς προκύπτουν οι συσχετίσεις:

- Το πρώτο bit από δεξιά αντιστοιχεί στον πρώτο πάροχο
- Το δεύτερο bit από δεξιά αντιστοιχεί στον δεύτερο πάροχο
- Το τρίτο bit από δεξιά αντιστοιχεί στον τρίτο πάροχο

Στο παράδειγμά μας, το γεγονός ότι ισχύει "*Time Consent = (0,0,**1**)"* μεταφράζεται σε απόκτηση του πεδίου "*Time"* των δεδομένων μόνο από τον πρώτο πάροχο. Αντίστοιχα εξάγονται τα αποτελέσματα και για τους τρεις παρόχους και παρουσιάζονται στους παρακάτω πίνακες.

Πάροχος 1: Ληφθέντα δεδομένα

Time	VehicleID	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density_y
0	3	53	-	-	-	95079	53	1

Πάροχος 2: Ληφθέντα δεδομένα

Time	VehicleID	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density_y
-	3	-	-	600867	5676073	95079	-	-

Πάροχος 3: Ληφθέντα δεδομένα

Time	VehicleID	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density_y
-	3	5	-	600867	5676073	-	53	-

7 Περιγραφή τεχνικής αρχιτεκτονικής

Στο κεφάλαιο 4 (4.1.3), παρουσιάστηκε η αρχιτεκτονική λύση σε επίπεδο υπηρεσιών για το σύστημα διαχείρισης της πρόσβασης πάνω στα ρεύματα δεδομένα των οχημάτων. Για την υλοποίηση του συστήματος σε τεχνικό επίπεδο προτείνεται η χρήση της πλατφόρμας υπολογιστικού νέφους Azure.

7.1 Πλατφόρμα Azure

Η πλατφόρμα Azure της Microsoft ανακοινώθηκε τον Οκτώβριο του έτους 2008 ενώ η διάθεση της για εμπορικούς σκοπούς ξεκίνησε τον Φεβρουάριο του έτους 2010.[21]

Είναι πλατφόρμα υπολογιστικού νέφους που χρησιμοποιείται για την ανάπτυξη, τον έλεγχο, τη διαχείριση και την υλοποίηση εφαρμογών και υπηρεσιών μέσα από απομακρυσμένα κέντρα δεδομένων υπό τη διαχείριση της Microsoft.

Εικόνα 10: Datacenters Microsoft Azure [22]

Οι υπηρεσίες που παρέχονται στην πλατφόρμα Azure είναι πάνω από 600 με τον αριθμό διαρκώς να αυξάνεται, καλύπτοντας ένα ευρύ φάσμα τεχνολογιών. Το εύρος των λύσεων περιλαμβάνει τόσο υπηρεσίες της Microsoft όσο και υπηρεσίες ανοικτού κώδικα και τρίτων κατασκευαστών λογισμικού. Επίσης, μπορεί κανείς να συναντήσει υπηρεσίες και στα τρία μοντέλα υπηρεσιών του υπολογιστικού νέφους όπως περιγράφονται στη συνέχεια: [23]

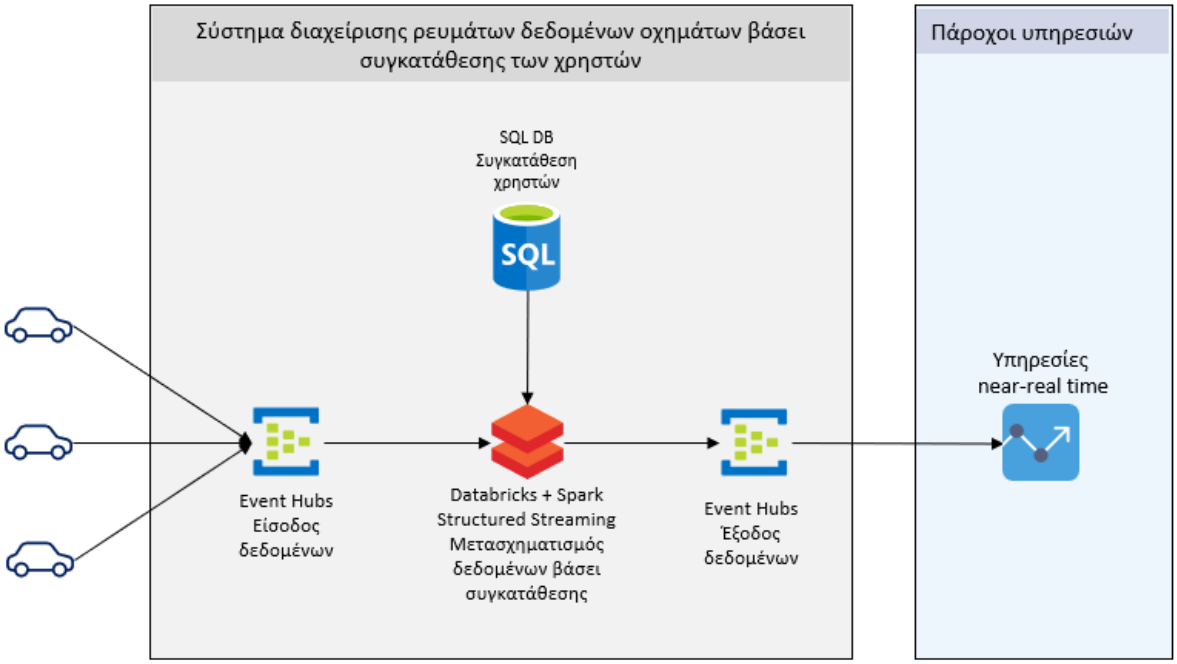
1. Λογισμικό ως Υπηρεσία (Software as a Service-SaaS): Είναι το μοντέλο παροχής λογισμικού κατά το οποίο το λογισμικό και τα σχετικά δεδομένα φιλοξενούνται στο “νέφος” και παρέχονται στο χρήστη μέσω συνδρομής. Οι χρήστες συνήθως έχουν πρόσβαση στις SaaS υπηρεσίες απομακρυσμένα μέσω ενός περιηγητή ιστού.
2. Πλατφόρμα ως Υπηρεσία (Platform as a Service-PaaS): Στο χρήστη παρέχεται σαν υπηρεσία μία πλατφόρμα πάνω στην οποία μπορεί να αναπτύσσει, να εκτελεί και να

διαχειρίζεται τις εφαρμογές του, χωρίς να απαιτείται η διαχείριση της αντίστοιχης υποδομής που είναι απαραίτητη για τη φιλοξενία της πλατφόρμας και την υποστήριξη των εφαρμογών.

3. Υποδομή ως υπηρεσία (Infrastructure as a Service-IaaS): Σύμφωνα με το μοντέλο αυτό, αξιοποιούνται απομακρυσμένοι υπολογιστικοί πόροι, συνήθως από οργανισμούς, ώστε να εξυπηρετούνται οι ανάγκες τους. Η υποδομή αυτή μπορεί να απαρτίζεται από υπολογιστικές μηχανές, αποθηκευτικό χώρο, δικτυακές συσκευές, διακομιστές, εξυπηρετητές και λογισμικό κατά απαίτηση του οργανισμού.

7.2 Προτεινόμενη αρχιτεκτονική συστήματος

Η αρχιτεκτονική της προτεινόμενης λύσης σε επίπεδο συστατικών της πλατφόρμας Azure παρουσιάζεται στην Εικόνα 11, και στη συνέχεια περιγράφονται τα δομικά μέρη της.



Εικόνα 11: Προτεινόμενη λύση σε πλατφόρμα MS Azure

7.3 Συστατικά μέρη αρχιτεκτονικής λύσης

7.3.1 Event Hubs

7.3.1.1 Περιγραφή

Το Event Hubs προορίζεται για εργασίες που αφορούν ρεύματα δεδομένων μεγάλης κλίμακας και χρησιμοποιείται σαν υπηρεσία εισαγωγής συμβάντων. Ανήκει στο μοντέλο PaaS του υπολογιστικού νέφους και μας δίνει τη δυνατότητα να λαμβάνουμε και να επεξεργαζόμαστε εκατομμύρια γεγονότων ανά δευτερόλεπτο.

7.3.1.2 Δυνατότητες [24]

- Υποστήριξη για επεξεργασία ρευμάτων δεδομένων σε πραγματικό χρόνο (real-time processing), αλλά και σε παρτίδες (batch processing)

- Αποθήκευση γεγονότων σε αποθηκευτικό χώρο, Azure Blob Storage ή Azure Data Lake Storage μέσω της ενσωματωμένης λειτουργίας Capture
- Κλιμακοσιμότητα. Με την επιλογή auto-inflate, οι μονάδες διεκπεραίωσης (Throughput Units), το πλήθος των οποίων καθορίζει τον όγκο των δεδομένων που μπορούν να εισαχθούν και να εξαχθούν από ένα Event Hub, αυξάνονται αυτόματα για να συναντούν τις απαιτήσεις του φορτίου της εφαρμογής
- Είσοδος δεδομένων από οποιαδήποτε εφαρμογή με χρήση των πρωτοκόλλων AMQP, HTTPS και Apache Kafka
- Υποστήριξη και διασύνδεση με Apache Kafka οικοσύστημα, χωρίς την ανάγκη για δημιουργία και παραμετροποίηση Kafka Clusters
- Κατανάλωση των γεγονότων εύκολα από πολλές και διαφορετικές εφαρμογές παράλληλα με κατάλληλη διαμέριση των γεγονότων

7.3.1.3 Αρχιτεκτονική Event Hubs

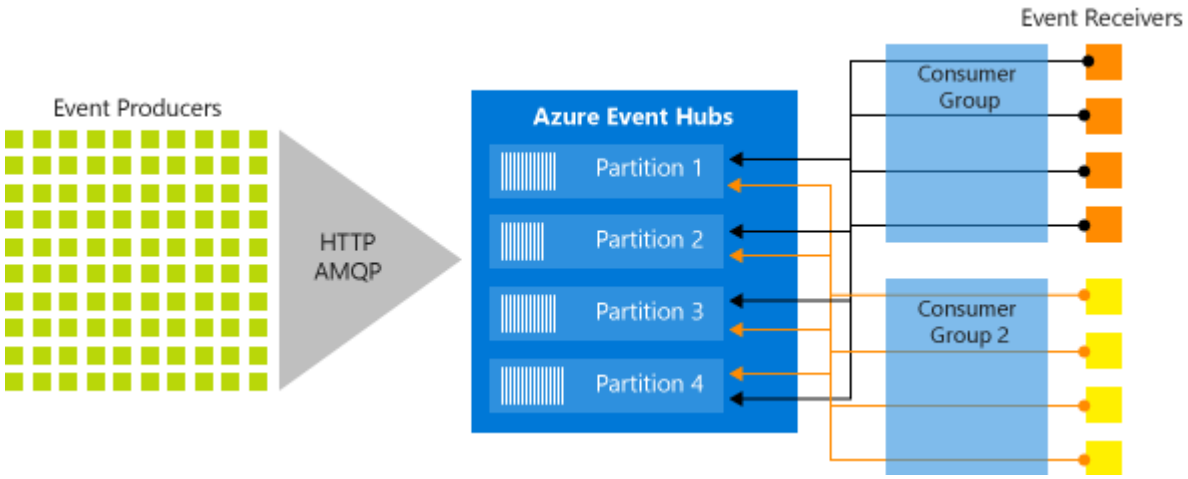
Τα κύρια στοιχεία της αρχιτεκτονικής του Event Hubs είναι τα εξής [24]

- Παραγωγοί (Event producers): Είναι οποιαδήποτε οντότητα στέλνει δεδομένα στο Event Hub χρησιμοποιώντας ένα εκ των πρωτοκόλλων HTTP, AMQP, Apache Kafka
- Διαμερίσεις (Partitions): Αποτελεί ένα υποσύνολο των δεδομένων που εισέρχονται στο Event Hub.
- Ομάδες καταναλωτών (Consumer Groups): Είναι μια ξεχωριστή όψη του Event Hub. Διαφορετικές ομάδες καταναλωτών δίνουν τη δυνατότητα σε εφαρμογές να καταναλώνουν ανεξάρτητα τα δεδομένα που εισέρχονται στο Event Hubs με το δικό τους ρυθμό.
- Αποδέκτες (Event Receivers): Είναι οποιαδήποτε οντότητα η οποία διαβάζει δεδομένα από το Event Hub.

7.3.1.4 Ρυθμός εισόδου και εξόδου δεδομένων

Ο ρυθμός εισόδου και εξόδου δεδομένων καθορίζεται από τον αριθμό των TUs που χρησιμοποιεί το Event Hubs. Επίσης τα TUs καθορίζουν τον όγκο των δεδομένων που μπορούν να διατηρηθούν στο Event Hub έως ότου παρέλθει η χρονική διάρκεια διατήρησης. Συγκεκριμένα τα στατιστικά όταν διαθέτουμε ένα TU είναι: [25]

- Είσοδος 1MB ή 1000 γεγονότων του 1KB/δευτερόλεπτο
- Έξοδος 2MB ή 4096 γεγονότων του 1KB/δευτερόλεπτο
- Αποθήκευση έως 84GB δεδομένων



Εικόνα 12: Azure Event Hubs [24]

7.3.1.5 Εφαρμογή στην προτεινόμενη λύση

Στην προτεινόμενη αρχιτεκτονική το Event Hubs χρησιμοποιείται τόσο για την εισαγωγή των δεδομένων στο σύστημα, όπου και επεξεργάζονται, όσο και για την έξοδο αυτών σε διαφορετικά ρεύματα, με το καθένα από αυτά να προορίζεται για τον κατάλληλο πάροχο υπηρεσιών σύμφωνα με την επεξεργασία που προηγήθηκε.

7.3.2 SQL Database

Η βάση δεδομένων SQL είναι μία γενικού σκοπού σχεσιακή βάση δεδομένων που παρέχεται από την πλατφόρμα Azure ως υπηρεσία (database-as-a-service). Στην προτεινόμενη λύση, χρησιμοποιείται για την αποθήκευση της συγκατάθεσης των χρηστών και των στοιχείων των λογαριασμών των χρηστών.

7.3.3 Spark Structured Streaming

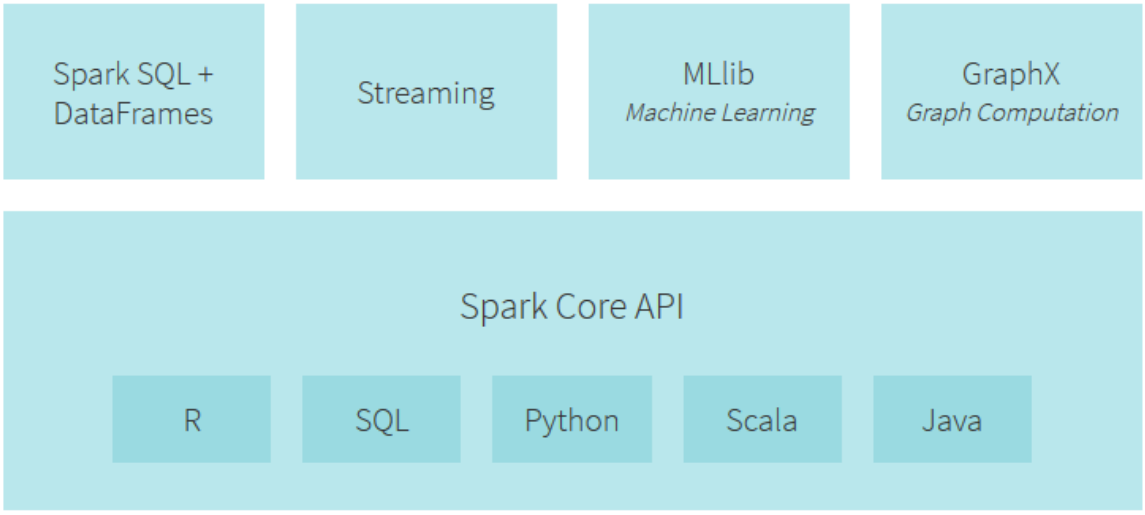
Προτού περιγράψουμε το στοιχείο Spark Structured Streaming, το οποίο στην προτεινόμενη αρχιτεκτονική χρησιμοποιείται για την επεξεργασία του ρεύματος δεδομένων, θα ασχοληθούμε με τη μηχανή Spark στην οποία βασίζεται και το εν λόγω στοιχείο αλλά και όλοι οι μετασχηματισμοί, η επεξεργασία και η ανάλυση που πραγματοποιούνται στα δεδομένα στο προτεινόμενο σύστημα.

7.3.3.1 Apache Spark

Το Apache Spark αποτελεί μια ισχυρή μηχανή ανοικτού κώδικα κατάλληλη για την επεξεργασία δεδομένων μεγάλης κλίμακας και για μηχανική μάθηση με την αρχική δημιουργία της να χρονολογείται το 2009 στο UC Berkeley και έπειτα να μεταβιβάζεται στον οργανισμό λογισμικού Apache. Έχει αξιοποιηθεί από μεγάλο πλήθος οργανισμών, με ιδιαίτερα υψηλές απαιτήσεις στην επεξεργασία δεδομένων μεγάλης κλίμακας, που αγγίζουν δεδομένα τάξης μεγέθους των petabytes και κόμβους εργασίας περισσοτέρων των 8000, όπως οι εταιρείες Netflix, Yahoo, eBay. [26][27]

Εικόνα 13: Apache Spark logo [26]

7.3.3.2 Οικοσύστημα Spark [27]



Εικόνα 14: Οικοσύστημα Apache Spark [27]

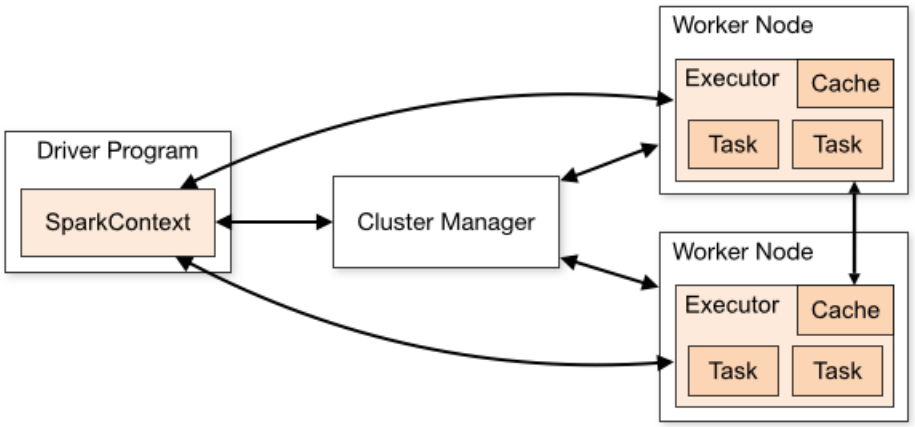
Το οικοσύστημα Spark αποτελείται από το Spark Core API που περιλαμβάνει όλες τις βασικές λειτουργίες που συνθέτουν τον πυρήνα μιας μηχανής επεξεργασίας Spark και τις καθιστά διαθέσιμες μέσω προγραμματιστικών διεπαφών για τις γλώσσες προγραμματισμού (R,Python,Scala,Java) και από προγραμματιστικές διεπαφές υψηλότερου επιπέδου οι οποίες έχουν χτιστεί πάνω στο Spark Core API. Οι διεπαφές που χτίζονται πάνω στο Spark Core API διακρίνονται ως εξής:

- Spark SQL: Είναι η λύση που παρέχεται για δομημένα δεδομένα στα οποία επιθυμούμε να κάνουμε SQL ερωτήματα. Στην Spark SQL μηχανή βασίζεται και το Spark Structured Streaming που ενδείκνυται για επεξεργασία ρεύματος δομημένων δεδομένων.
- Streaming API: Αποτελεί λύση για επεξεργασία ρευμάτων δεδομένων
- MLlib: Είναι η βιβλιοθήκη που παρέχεται για εφαρμογή μοντέλων μηχανικής μάθησης.
- GraphX: Συνιστά λύση για την επεξεργασία και το χειρισμό γράφων.

7.3.3.3 Spark Architecture [28]

Ο τρόπος εκτέλεσης μίας εφαρμογής Spark βασίζεται στην αρχιτεκτονική του μοντέλου master/slave. Στην Εικόνα 15 βλέπουμε την αρχιτεκτονική που ένα σύστημα ακολουθεί όταν εκτελεί μία εφαρμογή Spark σε cluster mode, με το στοιχείο “Driver Program” σε ρόλο master

και τα στοιχεία “Worker Node” σε ρόλο slave. Στη συνέχεια ακολουθεί συνοπτική παρουσίαση των συστατικών της αρχιτεκτονικής.



Εικόνα 15: Εκτέλεση εφαρμογής σε Spark cluster mode [28]

7.3.3.3.1 Spark Driver Program

Το πρόγραμμα οδήγησης (Spark Driver Program) είναι υπεύθυνο για την εκτέλεση της εφαρμογής του χρήστη και την επιστροφή των αποτελεσμάτων σε αυτόν. Δημιουργεί το Spark Context που είναι το κύριο συστατικό μιας εφαρμογής Spark και είναι αυτό που δίνει την πρόσβαση στις λειτουργίες του Spark. Συνεπώς το πρόγραμμα οδήγησης αποφασίζει για τον τρόπο εκτέλεσης της εφαρμογής, την κατάτμησή της σε διεργασίες τις οποίες θα αναθέσει στους κόμβους εργασίας και θα παρακολουθεί την εκτέλεσή τους.

7.3.3.3.2 Cluster Manager

Το Spark Context κατά την εκτέλεση μιας εφαρμογής θα επικοινωνήσει με το διαχειριστή του cluster έτσι ώστε αυτός να δεσμεύσει πόρους συστήματος και να δημιουργήσει τους κόμβους εργασίας, με σκοπό να τους επιστρέψει στην εφαρμογή.

7.3.3.3.3 Worker Nodes

Στους κόμβους εργασίας, γίνεται η εκτέλεση ενός τμήματος του φόρτου εργασίας της εφαρμογής σύμφωνα με την ανάθεση του προγράμματος οδήγησης. Έτσι ο Executor του κάθε κόμβου εργασίας εκτελεί το έργο που έχει αναλάβει ο κόμβος να διεκπεραιώσει.

7.3.3.4 Στοιχεία της αρχιτεκτονικής του Spark

7.3.3.4.1 Οντότητες δεδομένων στο Spark

Κοιτώντας την προγραμματιστική διεπαφή που μας παρέχει η μηχανή Spark παρατηρούμε ότι έχουμε τη δυνατότητα να εκμεταλλευτούμε τρεις οντότητες δεδομένων. Συγκεκριμένα, μπορούμε να κάνουμε χρήση των DataFrame, Dataset και RDD. Αρχικά πρέπει να αναφέρουμε ότι το DataFrame και το Dataset αποτελούν οντότητες δεδομένων που εσωτερικά μετατρέπονται σε RDD και βρίσκονται σε υψηλότερο αφαιρετικό επίπεδο. Για το λόγο αυτό θα μελετήσουμε περεταίρω τα RDDs. [29]

7.3.3.4.2 Resilient Distributed Datasets (RDDs) [30],[31]

Resilient Distributed Dataset (RDD) είναι μία συλλογή από δεδομένα (Dataset) τα οποία κατανέμονται σε μία σειρά από κόμβους ενός συνεργατικού δικτύου κόμβων διεργασίας (Distributed) γεγονός που επιτρέπει και την κατανεμημένη εκτέλεση μιας εργασίας. Επίσης, βασικό χαρακτηριστικό ενός RDD είναι η δυνατότητα ανακατασκευής και επανυπολογισμού της εκάστοτε χαμένης διαμέρισής του (Resilient) σε περίπτωση αστοχίας ενός κόμβου, χαρακτηριστικό ιδιαίτερα σημαντικό αν αναλογιστούμε ότι η εκτέλεση μιας εργασίας γίνεται σε ένα συνεργατικό δίκτυο από πιθανώς μεγάλο πλήθος κόμβων.

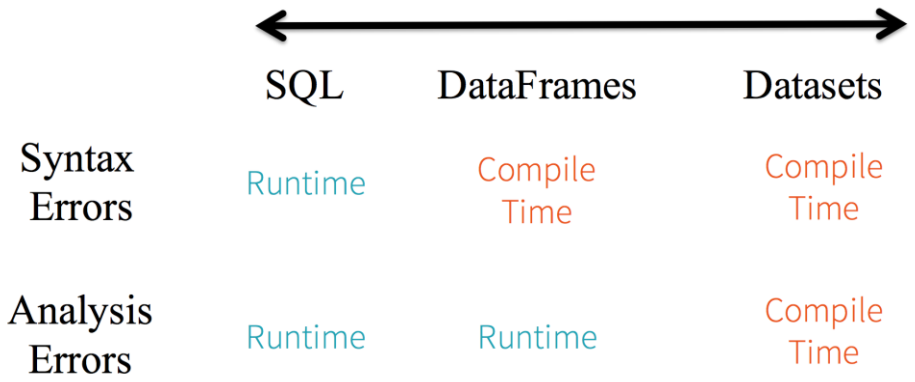
Οι πράξεις που μπορούμε να εκτελέσουμε πάνω σε ένα RDD χωρίζονται στις δύο ακόλουθες κατηγορίες:

- Μετασχηματισμοί (Transformations) : Είναι οι πράξεις που έχουν σαν αποτέλεσμα τη δημιουργία ενός νέου RDD από κάποιο υπάρχον. Αυτό συμβαίνει διότι τα RDDs είναι στοιχεία που δεν μπορούν να μεταβληθούν (immutable) και έτσι η μεταβολή τους πραγματοποιείται με τη δημιουργία νέων.
- Actions: Είναι οι πράξεις που έχουν σαν αποτέλεσμα την επιστροφή τιμών στο πρόγραμμα οδήγησης, σαν επακόλουθο υπολογισμών πάνω σε RDDs.

7.3.3.4.3 DataFrames και Datasets [29]

DataFrame είναι η συλλογή δεδομένων που είναι οργανωμένη σε ονοματισμένες στήλες. Σε επίπεδο λογικής αντιστοιχίζεται με πίνακα στο σχεσιακό μοντέλο. Το συγκεκριμένο API έχει εισαχθεί από την έκδοση Spark 1.3 και εκμεταλλεύεται τις βελτιστοποιήσεις που κάνει η μηχανή Spark SQL. Σφάλματα στη σύνταξη ελέγχονται κατά τη μεταγλώττιση.

Dataset είναι η συλλογή κατανεμημένων δεδομένων. Η διεπαφή Dataset του Spark που υπάρχει από την έκδοση Spark 1.6 συνδυάζει τα χαρακτηριστικά των RDDs με τις δυνατότητες που παρέχει η μηχανή Spark SQL για βελτιστοποίηση εκτέλεσης των πράξεων επί των δεδομένων. Με το συγκεκριμένο API σφάλματα στη σύνταξη, όπως για παράδειγμα ένα τυπογραφικό στο όνομα μιας μεθόδου, και στην ανάλυση, όπως είναι ένας λάθος τύπος μίας μεταβλητής, εντοπίζονται στο στάδιο της μεταγλώττισης. Η δυνατότητα αυτή προκύπτει διότι ένα Dataset αποτελείται από αντικείμενα JVM με προκαθορισμένο τύπο και επομένως υπάρχει δυνατότητα για έλεγχο στους τύπους κατά τη μεταγλώττιση.

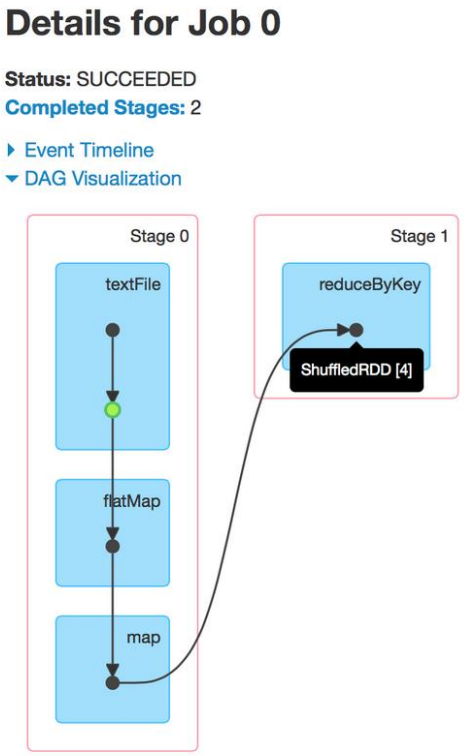


Εικόνα 16: Συντακτικά σφάλματα και σφάλματα ανάλυσης [32]

7.3.3.4.4 Κατευθυνόμενοι Ακυκλικοί Γράφοι στον πυρήνα του Spark

Όταν γίνεται μία εργασία πάνω σε RDDs μπορεί να κατατμηθεί σε επιμέρους ξεχωριστές διεργασίες, οι οποίες θα αποτελέσουν τους κόμβους ενός κατευθυνόμενου ακυκλικού γράφου ενώ οι ακμές του φανερώνουν την αλληλουχία αυτών. Χαρακτηριστικό στοιχείο μιας Spark εφαρμογής είναι και ο DAGScheduler ο οποίος αναλαμβάνει τη μετατροπή του λογικού πλάνου εκτέλεσης σε φυσικό. Η διαδικασία αυτή έχει σαν αποτέλεσμα να ελέγχονται οι μετασχηματισμοί που λαμβάνουν χώρα και να τμηματοποιείται ο γράφος σε στάδια ανάλογα με το είδος αυτών. [33]

Συγκεκριμένα, διακρίνουμε δύο είδη μετασχηματισμών, τους μετασχηματισμούς υπό τη στενή έννοια (narrow-transformations) και τους μετασχηματισμούς υπό την ευρεία έννοια (wide-transformations). Το στοιχείο που τους διαφοροποιεί είναι οι εξαρτήσεις των διαμερίσεων των νεοδημιουργηθέντων RDDs από τα αρχικά RDDs στα οποία εφαρμόστηκε η πράξη. Στην επόμενη υπο-ενότητα αναλύονται οι εξαρτήσεις αυτές. Τελικά, ο γράφος σπάει σε δύο ή περισσότερα στάδια αν έχουμε μετασχηματισμούς υπό την ευρεία έννοια ενώ στην αντίθετη περίπτωση οι διεργασίες εκτελούνται στο ίδιο στάδιο.



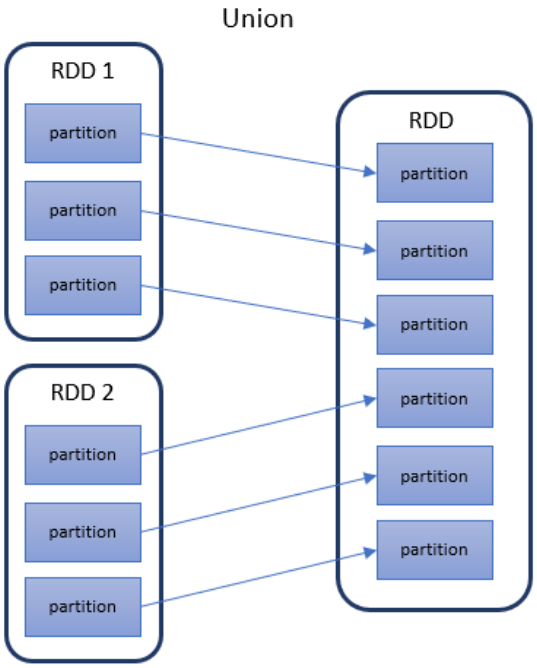
Εικόνα 17: Παράδειγμα γράφου 2 σταδίων [33]

Στο παράδειγμα της εικόνας βλέπουμε τρεις διεργασίες (textFile operation, flatMap, map) να εκτελούνται στο ίδιο στάδιο ενώ η διεργασία reduceByKey να εκτελείται σε διαφορετικό υποδηλώνοντας την ανάγκη για μεταφορά δεδομένων ανάμεσα σε διαμερίσεις διαφορετικών κόμβων.

7.3.3.5 Εξαρτήσεις υπό τη στενή και την ευρεία έννοια [31]

Εξαρτήσεις υπό τη στενή έννοια (Narrow Dependencies): Προκύπτουν, όταν κάθε μία από τις διαμερίσεις του RDD που δημιουργείται εξαρτάται το πολύ από μία διαμέριση του αρχικού RDD.

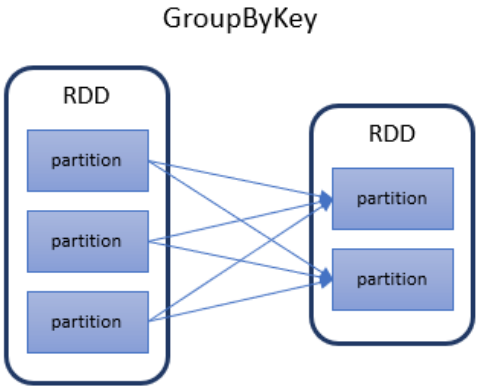
Στην εικόνα που ακολουθεί παρουσιάζεται ένα παράδειγμα της πράξης της συνένωσης (union) πάνω σε δύο RDDs. Παρατηρούμε ότι η κάθε διαμέριση του νέου RDD εξαρτάται ακριβώς από μία διαμέριση στο αντίστοιχο γονεϊκό RDD και επομένως πληρείται η απαίτηση για το πολύ μία εξάρτηση.



Εικόνα 18: Παράδειγμα narrow-dependencies

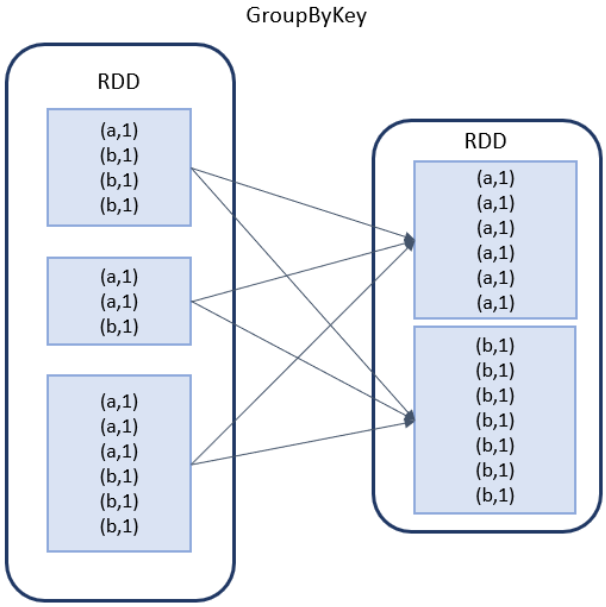
Εξαρτήσεις υπό την ευρεία έννοια (Wide Dependencies): Προκύπτουν, όταν μία από τις διαμερίσεις του RDD που δημιουργείται, εξαρτάται από περισσότερες της μίας διαμερίσεις του αρχικού RDD. Στην εικόνα που ακολουθεί παρουσιάζεται ένα παράδειγμα της πράξης της ομαδοποίησης βάσει κάποιου γνωρίσματος (groupByKey) πάνω σε δύο RDDs.

Παρατηρούμε ότι η κάθε διαμέριση του νέου RDD εξαρτάται από περισσότερες της μίας διαμερίσεις των αρχικών RDDs. Αυτή η εξάρτηση έχει σαν αποτέλεσμα την ανακατανομή των δεδομένων μεταξύ των διαμερίσεων (Shuffling) και σε περίπτωση που οι διαμερίσεις δεν βρίσκονται στον ίδιο κόμβο εκτέλεσης (Executor) αυτό έχει ως αποτέλεσμα η μεταφορά των δεδομένων αυτή να προκαλεί κίνηση στο δίκτυο με αρνητική επίδραση στην επίδοση.



Εικόνα 19: Παράδειγμα wide-dependencies

Για την καλύτερη κατανόηση του φαινομένου Shuffling παρουσιάζουμε ένα ακόμα παράδειγμα που μας δείχνει τη μεταφορά των δεδομένων.



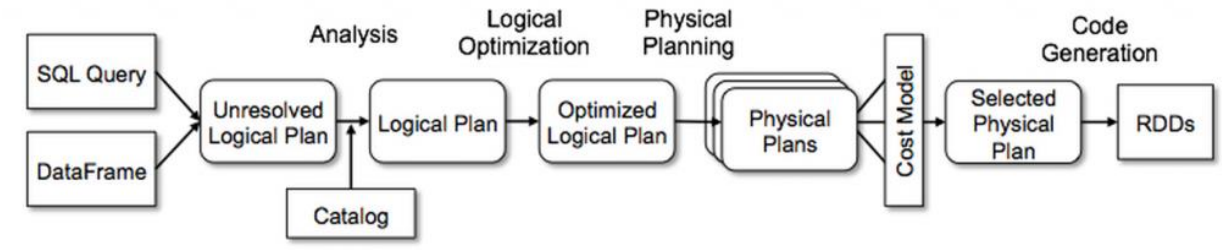
Εικόνα 20: Wide-dependencies groupByKey

7.3.3.6 Catalyst Optimizer [34]

Στον πυρήνα της Spark SQL μηχανής βρίσκεται το στοιχείο Catalyst Optimizer, η χρήση του οποίου συνεισφέρει στο να εκτελούνται οι πράξεις και τα ερωτήματα πάνω στα δεδομένα με το βέλτιστο τρόπο.

Ο κυρίαρχος τύπος δεδομένων στο Catalyst είναι τα δέντρα. Για το χειρισμό των δέντρων που προκύπτουν χρησιμοποιούνται κανόνες, που είναι συναρτήσεις που μετασχηματίζουν τα δέντρα που δέχονται ως είσοδο. Οι κανόνες αυτοί συνήθως κάνουν χρήση τεχνικών που συναντάμε στις γλώσσες συναρτησιακού προγραμματισμού όπως είναι η αντιπαραβολή προτύπων (pattern matching).

Το πλάνο εκτέλεσης του Catalyst Optimizer φαίνεται στην ακόλουθη εικόνα και διακρίνεται στα εξής στάδια.



Εικόνα 21: Πλάνο εκτέλεσης Catalyst Optimiser [34]

1. Ανάλυση (Analysis): Αρχικά, συναντάμε το στάδιο της ανάλυσης, στο οποίο καθορίζονται οι αναφορές και οι σχέσεις που συμμετέχουν στις πράξεις, αποτέλεσμα του οποίου είναι το λογικό πλάνο εκτέλεσης. Για παράδειγμα, όταν επιλέγουμε μία στήλη με συγκεκριμένο όνομα, θα πρέπει να ελεγχθεί ο πίνακας για να διαπιστωθεί ότι πρόκειται για έγκυρο όνομα αλλά και για να γνωρίζουμε τον τύπο δεδομένων της στήλης.
2. Βελτιστοποίηση λογικού πλάνου (Logical Optimization): Στο στάδιο αυτό οι βελτιστοποιήσεις πραγματοποιούνται βάσει ενός συνόλου κανόνων.
3. Σχεδιασμός φυσικού πλάνου εκτέλεσης: Στο στάδιο αυτό το λογικό πλάνο μετατρέπεται στο πλάνο εκτέλεσης για τη μηχανή Spark. Μπορεί να προκύψουν περισσότερα του ενός πλάνα εκτέλεσης, στην περίπτωση αυτή επιλέγεται το τελικό μέσω ενός μοντέλου κόστους. Χαρακτηριστικό παράδειγμα αποτελεί ο τρόπος εκτέλεσης της πράξης join που μελετάται στην παράγραφο 10.7.1.
4. Δημιουργία κώδικα: Τελευταίο στάδιο αποτελεί η δημιουργία του Java bytecode ώστε να γίνει η εκτέλεση από τους κόμβους επεξεργασίας. Για τη δημιουργία του κώδικα αυτού γίνεται χρήση του χαρακτηριστικού Quasiquotes της γλώσσας προγραμματισμού Scala. Τα Quasiquotes επιτρέπουν τη δημιουργία αφηρημένων συντακτικών δέντρων (Abstract Syntax Trees-ASTs) σε Scala, τα οποία ύστερα χρησιμοποιεί ο μεταγλωττιστής (compiler) της Scala για να δημιουργήσει bytecode. Έτσι για παράδειγμα μια έκφραση σε SQL αναπαρίσταται σε ένα AST σε Scala και έπειτα μεταγλωττίζεται και μπορεί να εκτελεστεί.

7.3.3.7 Πλεονεκτήματα Spark

Όσον αφορά τα πλεονεκτήματα του Apache Spark μπορούμε να διακρίνουμε δύο βασικά χαρακτηριστικά όπως περιγράφονται στη συνέχεια.

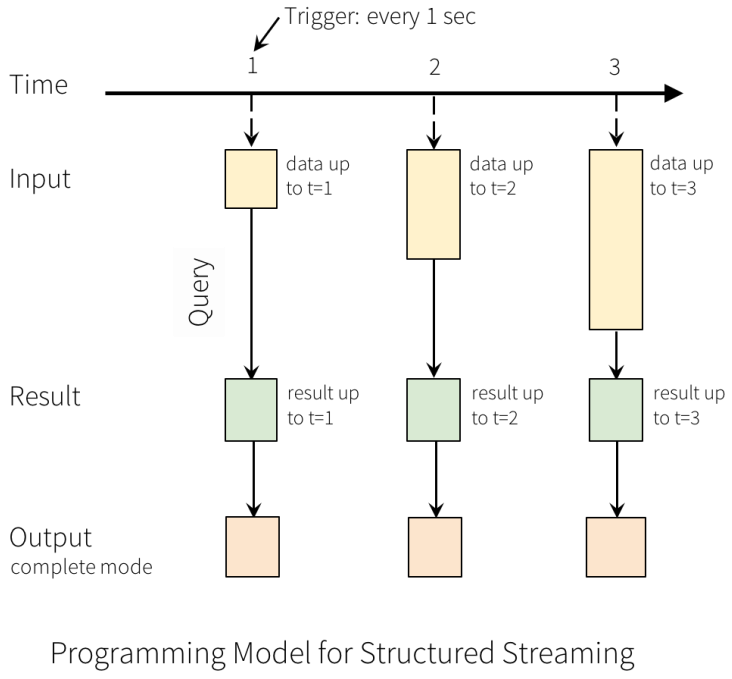
- Ταχύτητα: Το ανταγωνιστικότερο πλεονέκτημά του είναι η ταχύτητα που επιτυγχάνει και μπορεί να είναι έως και εκατό φορές καλύτερη, έναντι της μεθόδου MapReduce που χρησιμοποιεί το Apache Hadoop [27] και αποτελεί επίσης μία λύση για την κατανεμημένη επεξεργασία δεδομένων μεγάλης κλίμακας. Η δυνατότητα για την επίτευξη υψηλών ταχυτήτων οφείλεται στο γεγονός ότι οι υπολογισμοί γίνονται στη μνήμη του εκάστοτε κόμβου εργασίας, περιορίζοντας τις προσβάσεις στους δίσκους για είσοδο-έξοδο δεδομένων.
- Ευκολία στη χρήση: Το Apache Spark παρέχει ένα σύνολο από APIs υψηλού επιπέδου, παρέχοντας στο χρήστη τη δυνατότητα για επεξεργασία δεδομένων με συναρτήσεις και πράξεις που γίνονται εύκολα κατανοητές, αποκρύπτοντας την πολυπλοκότητα της

υλοποίησής τους σε χαμηλό επίπεδο. Το γεγονός αυτό φαίνεται με τη δημιουργία των APIs που χειρίζονται DataFrames και Datasets στις νεότερες εκδόσεις έναντι των RDDs που υπήρχαν από την αρχική έκδοση. Επίσης, παρατηρούμε ότι το σύνολο πολλών διαφορετικών λειτουργιών βρίσκεται σε ένα ενιαίο πλαίσιο στη μηχανή Apache Spark.

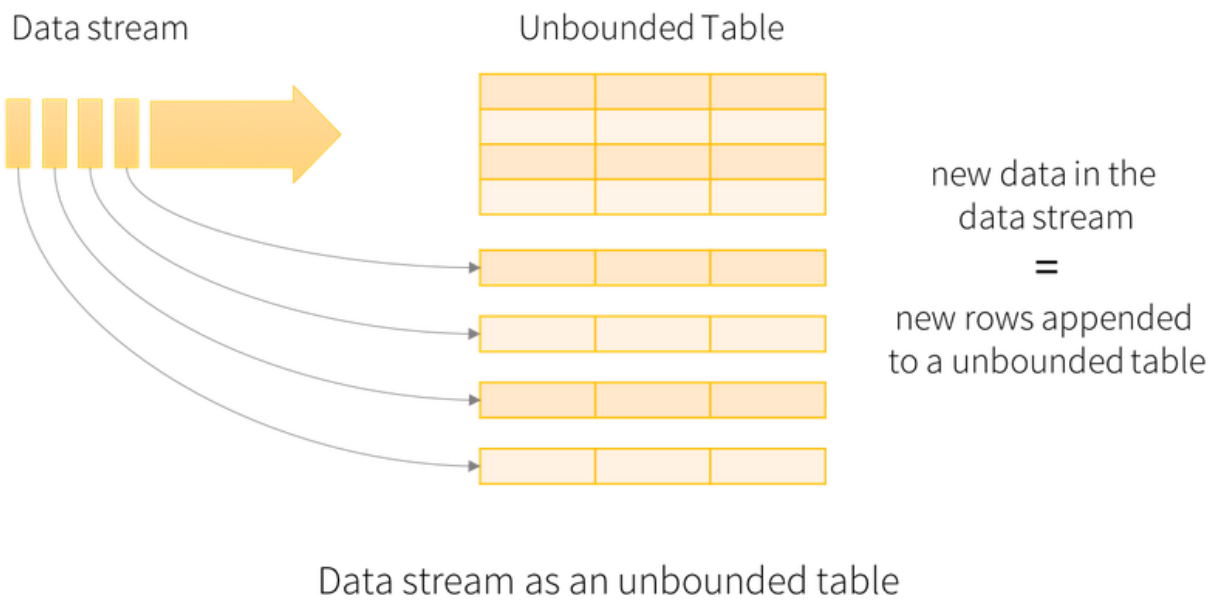
7.3.3.8 Spark Structured Streaming [35]

Το Spark Structured Streaming είναι μία κλιμακούμενη και ανθεκτική στα σφάλματα μηχανή επεξεργασίας ρευμάτων δεδομένων που βασίζεται στη μηχανή Spark SQL. Μας δίνει τη δυνατότητα να εκφράζουμε τα ερωτήματα επί των δεδομένων με την ίδια λογική που θα το κάναμε σε στατικά δεδομένα εξασφαλίζοντάς μας τη συνεχόμενη εκτέλεση και τη διαρκή ενημέρωση των αποτελεσμάτων καθώς φτάνουν στην είσοδο νέα δεδομένα. Στην πραγματικότητα, εσωτερικά ο μηχανισμός επεξεργάζεται τα δεδομένα με τη λογική του micro-batch processing, δηλαδή η είσοδος παίζει το ρόλο μίας σειράς μικρών ξεχωριστών εργασιών με την ίδια λογική της αρχικής ενιαίας εργασίας για το αντίστοιχο μικρό τμήμα των δεδομένων που καταφθάνουν. Το γεγονός αυτό δίνει τη δυνατότητα για επεξεργασία ρεύματος δεδομένων με καθυστέρηση που μπορεί να ξεκινά από τα 100 ms [36]. Επιπλέον παρέχονται εγγυήσεις για ανοχή σε σφάλματα, καθώς και για επεξεργασία ακριβώς μία φορά των δεδομένων.

Ένα ερώτημα πάνω στο ρεύμα εισόδου έχει σαν αποτέλεσμα τη δημιουργία ενός πίνακα αποτελεσμάτων. Αν ορίσουμε ένα διάστημα χρόνου (trigger interval) και κοιτάμε την είσοδο κάθε φορά που συμπληρώνεται αυτό το διάστημα παρατηρούμε ότι νέα δεδομένα φτάνουν στην είσοδο και ως επακόλουθο θα πρέπει να επεξεργαστούν και να ενημερωθεί ο πίνακας αποτελεσμάτων.



Εικόνα 22: Μοντέλο λειτουργίας structured streaming [37]



Εικόνα 23: Μοντελοποίηση ρεύματος δεδομένων [37]

Παρατηρούμε ότι το ρεύμα των δεδομένων μοντελοποιείται ως ένας μη οριοθετημένος ως προς τις γραμμές πίνακας, με τα νέα δεδομένα που φτάνουν να παίζουν το ρόλο νέων γραμμών που προστίθενται στον πίνακα αυτό. Με αυτή τη λογική διακρίνονται τρεις τρόποι λειτουργίας για δημιουργία της εξόδου [38]:

- Complete Mode: Ολόκληρος ο πίνακας εξόδου γράφεται σαν αποτέλεσμα
- Append Mode: Μόνο οι νέες γραμμές που προκύπτουν από το τελευταίο έναυσμα για επεξεργασία προστίθενται στον πίνακα εξόδου
- Update Mode: Μόνο οι γραμμές που αλλάζουν από το τελευταίο έναυσμα για επεξεργασία ενημερώνονται στον πίνακα αποτελεσμάτων

7.3.4 Azure Databricks

Το Databricks σχεδιάστηκε το 2013 από τους δημιουργούς που ξεκίνησαν το ερευνητικό έργο για το Apache Spark στο UC Berkeley και παρέχει μία δικτυακή πλατφόρμα για την ανάπτυξη εφαρμογών κάνοντας χρήση της τεχνολογίας Apache Spark, παρέχοντας ευκολία στη διαχείριση του Spark cluster. Στην παρούσα λύση χρησιμοποιείται το Azure Databricks που αποτελείται από την πλατφόρμα Databricks βελτιστοποιημένη και προσανατολισμένη στην πλατφόρμα Azure.[39]

8 Πλατφόρμα Azure για μεγάλα δεδομένα

Στην προηγούμενη ενότητα παρουσιάστηκαν τα συστατικά της πλατφόρμας Azure που προτείνονται για την υλοποίηση της προτεινόμενης λύσης. Στο σημείο αυτό, για λόγους πληρότητας, παρατίθεται συνοπτικά το σύνολο των παρεχόμενων υπηρεσιών από την πλατφόρμα Azure για την επεξεργασία και την ανάλυση μεγάλων δεδομένων. Συνδυασμός της προτεινόμενης λύσης με τις παρακάτω υπηρεσίες, χρησιμεύει δυνητικά στην ανάπτυξη πρόσθετων εφαρμογών στο πεδίο έρευνας.

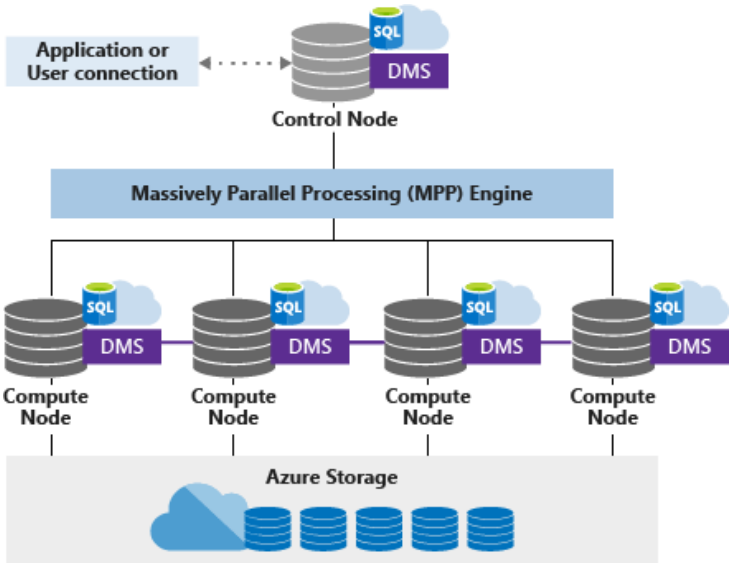
8.1 Azure HDInsight

Το Azure HDInsight παρέχει τη δυνατότητα για την ανάλυση και επεξεργασία δεδομένων μέσα από πληθώρα λογισμικών ανοικτού κώδικα που προορίζονται για αυτόν τον σκοπό. Ενδεικτικά αναφέρουμε το Hadoop, το Apache Kafka, το Apache Spark και το Apache Storm. Η λύση αυτή προσφερόταν και για την προτεινόμενη αρχιτεκτονική ως αντικαταστάτης του Azure Databricks καθώς και το HDInsight παρέχει τη δυνατότητα διασύνδεσης με όλα τα υπόλοιπα συστατικά της λύσης αλλά και την υποστήριξη του Apache Spark. Όμως, όντας το Azure Databricks προσανατολισμένο αποκλειστικά στο Apache Spark, που ήταν το βασικό συστατικό της προτεινόμενης αρχιτεκτονικής, αποτέλεσε προτιμότερη λύση. [40]

8.2 SQL Data Warehouse

Αποτελεί λύση για αποθήκη δεδομένων (data warehouse) βασισμένη στο cloud. Ενδείκνυται όταν το σενάριο εμπεριέχει ερωτήματα πάνω σε δεδομένα της τάξης των petabytes (1 PB = 10^{15} bytes). Το πρώτο στάδιο είναι αυτό της εισαγωγής των δεδομένων στην αποθήκη και πραγματοποιείται με PolyBase T-SQL ερωτήματα. Στο δεύτερο στάδιο πραγματοποιούνται τα κυρίως ερωτήματα πάνω στα δεδομένα που βρίσκονται στην αποθήκη ώστε να γίνει η επιθυμητή ανάλυση των δεδομένων. Για την επίτευξη υψηλών επιδόσεων των ερωτημάτων αυτών χρησιμοποιείται αρχιτεκτονική παράλληλης επεξεργασίας (MPP-Massively Parallel Processing). Το υπολογιστικό στάδιο και τα αποθηκευμένα δεδομένα βρίσκονται σε ξεχωριστά επίπεδα με αποτέλεσμα ο βαθμός κλιμάκωσης των υπολογισμών να είναι ανεξάρτητος των δεδομένων. Για τη διακίνηση των δεδομένων στους κόμβους επεξεργασίας χρησιμοποιείται ενσωματωμένος μηχανισμός. (Data Management Service).

Κάνοντας χρήση της SQL DW, η εκάστοτε εφαρμογή υποβάλλεται στον κόμβο ελέγχου (Control Node) για εκτέλεση και στη συνέχεια στη μηχανή παράλληλης επεξεργασίας (MPP engine) η οποία εφαρμόζει βελτιστοποιήσεις και αναθέτει τα ερωτήματα στους κόμβους επεξεργασίας (Compute Node) για την παράλληλη εκτέλεσή τους. [41][42]



Εικόνα 24: SQL Data Warehouse [42]

8.3 Azure Data Factory

Με το Azure Data Factory επιτυγχάνεται η διασύνδεση και η ολοκλήρωση συστημάτων που έχουν να κάνουν με εργασίες ροών δεδομένων. Συνεισφέρει στην υλοποίηση διαδικασιών ETL (Extract-Transform-Load). Στην ουσία, δίνει τη δυνατότητα για τη δημιουργία, ενορχήστρωση και αυτοματοποίηση ροών δεδομένων καθώς και των μετασχηματισμών πάνω σε αυτά.[43]

8.4 Azure Stream Analytics

Το Azure Stream Analytics είναι υπηρεσία για εφαρμογή ερωτημάτων σε ρεύματα δεδομένων με χρήση γλώσσας SQL για ανάλυση και μετασχηματισμούς δεδομένων σε πραγματικό χρόνο.

8.5 Data Lake Analytics

Το Azure Data Lake Analytics αποτελεί μία υπηρεσία για ανάλυση δεδομένων με χρήση της γλώσσας U-SQL που συνδυάζει τη λογική ερωτημάτων SQL με την ικανότητα έκφρασης επιπλέον μέσω C#. Χαρακτηρίζεται από ευκολία χρήσης καθώς ο χρήστης γράφει τα ερωτήματα που επιθυμεί πάνω στα δεδομένα και δεν απαιτείται να ασχολείται με τη διαμόρφωση του υπολογιστικού περιβάλλοντος.[44]

8.6 Machine Learning

Το Azure Machine Learning δίνει τη δυνατότητα για τη δημιουργία, την εκπαίδευση, τη διαχείριση και την αυτοματοποίηση μοντέλων μηχανικής μάθησης. Παρέχει στο χρήστη την ευκολία να διαχειρίζεται τα μοντέλα μέσω γραφικής διεπαφής και φυσικά την ανάπτυξη κώδικα μέσω Jupyter Notebooks.[45]

8.7 Azure Data Explorer

Το Azure Data Explorer αποτελεί και αυτό μία υπηρεσία για πραγματικού χρόνου ανάλυση δεδομένων αλλά και σταθερών δεδομένων. Για την πραγματοποίηση ερωτημάτων πάνω σε δεδομένα μπορεί να χρησιμοποιηθεί η γλώσσα SQL ή η γλώσσα ερωτημάτων “Kusto Query Language”, στην οποία τα ερωτήματα είναι μόνο ανάγνωσης, και χαρακτηρίζονται από ευκολία στον τρόπο σύνταξής τους.[46]

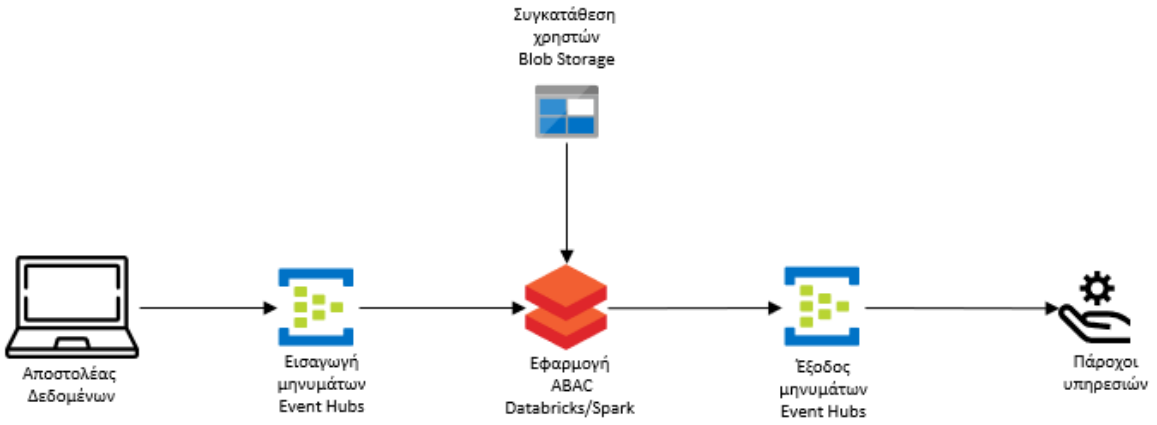
9 Αρχιτεκτονική υποσυστήματος για την εφαρμογή του ABAC

9.1 Αρχιτεκτονική λύση

Σε προηγούμενο κεφάλαιο (Κεφάλαιο 7ο) παρουσιάστηκε η προτεινόμενη τεχνική αρχιτεκτονική λύση για τη διαχείριση των δεδομένων των οχημάτων και των δεδομένων της συγκατάθεσης των χρηστών.

Αντικείμενο της προγραμματιστικής υλοποίησης στην παρούσα εργασία είναι το υποσύστημα που θα χρησιμοποιηθεί για τη διαχείριση της πρόσβασης στα ρεύματα δεδομένων των οχημάτων βάσει της συγκατάθεσης των χρηστών.

Την αρχιτεκτονική του υποσυστήματος αυτού παρουσιάζουμε στην Εικόνα 25.



Εικόνα 25: Αρχιτεκτονική υποσυστήματος προγραμματιστικής υλοποίησης

9.2 Συστατικά αρχιτεκτονικής λύσης υποσυστήματος

9.2.1 Αποστολέας δεδομένων

Για την προσομοίωση των οχημάτων υλοποιήθηκε εφαρμογή η οποία παίρνει ως είσοδο ένα αρχείο με στατικά δεδομένα τα οποία προέρχονται από οχήματα, το διατρέχει και αποστέλλει τα δεδομένα αυτά σε μορφότυπο JSON στο σύστημά μας, στην πλατφόρμα Azure.

9.2.2 Event Hubs

Το Event hubs είναι υπεύθυνο για τη λήψη των μηνυμάτων και αποτελεί το σημείο εισαγωγής των δεδομένων στο σύστημά μας ώστε αυτά να διαβαστούν και να επεξεργαστούν στο Spark Cluster. Επιπλέον χρησιμοποιείται για έξοδο των μηνυμάτων σε διαφορετικούς παρόχους υπηρεσιών σύμφωνα με την επεξεργασία ABAC.

9.2.3 Blob Storage

Το Blob Storage χρησιμοποιείται για αποθήκευση των δεδομένων που αφορούν τη συγκατάθεση των χρηστών και επίσης δύναται να χρησιμοποιηθεί για αποθήκευση των μηνυμάτων που εισέρχονται στο σύστημα για μακροπρόθεσμη διατήρηση.

9.2.4 Spark Cluster

Μέσω της πλατφόρμας Databricks δημιουργείται το Spark Cluster έτσι ώστε να υλοποιηθεί ο μηχανισμός ελέγχου πρόσβασης που βασίζεται στη λογική του μηχανισμού ABAC, κάνοντας χρήση της τεχνολογίας Spark Structured Streaming. Επιπλέον, σε δεύτερο χρόνο χρησιμοποιείται η τεχνολογία Spark για τη μέτρηση της καθυστέρησης της επεξεργασίας.

10 Τεχνικά στοιχεία υλοποίησης

10.1 Τα δεδομένα του προβλήματος

Για τις ανάγκες του εγχειρήματος χρησιμοποιήθηκαν δεδομένα που αφορούν την κίνηση οχημάτων στην Αμβέρσα του Βελγίου, και προήλθαν από το Harvard Dataverse [47] το οποίο παρέχει ένα αποθετήριο δεδομένων που προέρχεται από ερευνητικούς σκοπούς και προορίζεται για ερευνητικούς σκοπούς. Στα δεδομένα αυτά διακρίνουμε τις εξής κατηγορίες όπως φαίνεται στον πίνακα που ακολουθεί:

Πεδίο	Τύπος Δεδομένων	Περιγραφή
Time	integer	Ο χρόνος παίρνει τιμές πολλαπλάσιες των 60 δευτερολέπτων καθώς οι πομποί των αυτοκινήτων είχαν προγραμματιστεί να αποστέλλουν τα δεδομένα κάθε 60 δευτερόλεπτα. Σημειώνεται ότι σαν time δεν αναφέρεται η απόλυτη χρονική στιγμή αλλά η χρονική στιγμή σε σχέση με την έναρξη της παρατήρησης που έχει μηδενική τιμή
VehicleID	integer	Ένα μοναδικό αναγνωριστικό για κάθε όχημα
Speed	integer	Η ταχύτητα του οχήματος την τρέχουσα στιγμή
Dist_front	integer	Η απόσταση από το προπορευόμενο όχημα σε μέτρα
xCoord	integer	Το γεωγραφικό μήκος της τοποθεσίας του οχήματος
yCoord	integer	Το γεωγραφικό πλάτος της τοποθεσίας του οχήματος
SectionID	integer	Μοναδικό αναγνωριστικό για κάθε τμήμα του οδικού δικτύου
Avg_Sect_Speed	integer	Μέση τρέχουσα ταχύτητα για το σύνολο των οχημάτων που βρίσκονται στο τμήμα του δικτύου με το ίδιο SectionID
Sect_Density	integer	Αριθμός οχημάτων στο ίδιο τμήμα του οδικού δικτύου

Πίνακας 1: Περιγραφή δεδομένων εισόδου

Για την εφαρμογή του μηχανισμού ελέγχου πρόσβασης, θα χρησιμοποιήσουμε στην υλοποίησή μας το πεδίο VehicleID για την αντιστοίχιση του οχήματος με τη συγκατάθεση που υπάρχει στο σύστημα δηλωμένη για το συγκεκριμένο όχημα.

Τα υπόλοιπα πεδία χρησιμοποιούνται για παρατήρηση των αποτελεσμάτων της επεξεργασίας.

Τα δεδομένα που αφορούν τη συγκατάθεση των χρηστών δημιουργήθηκαν στο πλαίσιο της υλοποίησης.

Στο Κεφάλαιο 6 περιγράφηκε η μοντελοποίηση του προβλήματος και της ροής των δεδομένων.

Βάσει αυτής της μοντελοποίησης προκύπτει η ανάγκη να δημιουργηθεί για κάθε όχημα η συγκατάθεσή του.

Για να επιτευχθεί το παραπάνω, αναπτύχθηκε κώδικας που παράγει τυχαίους συνδυασμούς συγκατάθεσης και οι παραγόμενες εγγραφές φορτώνονται ως αρχείο στο σύστημα. Για το σκοπό αυτόν ακολουθήθηκε η παρακάτω διαδικασία:

1. Ανέβασμα αρχείου δεδομένων οχημάτων στο Blob Storage του Storage Account.
2. Δημιουργία Python Notebook στο περιβάλλον εργασίας του Databricks
3. Σύνδεση του Databricks με το αρχείο δεδομένων
4. Ανάγνωση των δεδομένων μέσω του Spark Python API
5. Καταμέτρηση μόνο των διαφορετικών VehicleID
6. Δημιουργία συγκατάθεσης για αριθμό οχημάτων ίσο με το πλήθος των διαφορετικών VehicleIDs
7. Εγγραφή του νέου πίνακα δεδομένων συγκατάθεσης σε αρχείο στο Blob Storage.

Ο αντίστοιχος κώδικας παρατίθεται στο παράρτημα 2.

10.2 Αποστολή δεδομένων οχημάτων

Για την αποστολή των δεδομένων μελετήθηκε η εξής περίπτωση:

- Εισαγωγή δεδομένων στο σύστημα από εφαρμογή παραγωγού που τρέχει στον τοπικό υπολογιστή και αποστέλλει τα δεδομένα πάνω από TCP/SSL σύνδεση χρησιμοποιώντας το πρωτόκολλο εφαρμογής HTTP.

Ο παραγωγός δεδομένων υλοποιήθηκε σε γλώσσα Java. Συγκεκριμένα χρειάστηκε να διαβαστεί το αρχείο .csv με τα δεδομένα των οχημάτων γραμμή προς γραμμή και για κάθε μία από αυτές να δημιουργηθεί ένα αντικείμενο JSON, το οποίο στη συνέχεια σειριοποιείται ώστε να αποσταλεί στο σύστημά μας.

Στη συνέχεια δημιουργήθηκε η σύνδεση με το Event Hub που θα υποδεχτεί τα δεδομένα. Για την πιστοποίηση της σύνδεσης χρησιμοποιήθηκε κοινόχρηστο διακριτικό υπογραφής πρόσβασης (Shared Access Signature token).

Τέλος χρησιμοποιήθηκε η προκαθορισμένη επιλογή για την κατανομή των μηνυμάτων στις διαμερίσεις του Event Hubs που υπάγεται στον αλγόριθμο Round Robin.

10.3 Είσοδος και έξοδος δεδομένων

10.3.1 Τεχνικές επιλογές στα Event Hubs

Στο στάδιο της εισαγωγής δεδομένων συναντάμε το Azure Event Hubs όπως περιγράφηκε και στην παράγραφο 7.3.1. Η παραμετροποίηση του Azure Event Hubs βασίζεται στον εκτιμώμενο ρυθμό εισαγωγής μηνυμάτων στο σύστημά μας.

Για την παραμετροποίηση του Event Hubs, απαιτείται, μεταξύ άλλων, ο ορισμός των παρακάτω παραμέτρων:

- **Throughput Units (TUs):** Όπως αναφέρθηκε τα TUs καθορίζουν τη χωρητικότητα της σύνδεσης του Event Hubs δηλαδή τον όγκο των μηνυμάτων που μπορεί να εξυπηρετήσει ανά δευτερόλεπτο. Η επιλογή εξαρτάται από τον εκτιμώμενο όγκο αλλά στην προκειμένη περίπτωση χρησιμοποιήσαμε τη δυνατότητα για αυτόματη δέσμευση επιπλέον TUs όταν το φορτίο το απαιτεί.
- **Αριθμός διαμερίσεων (Partitions):** Σε αντίθεση με τα TUs ο αριθμός των διαμερίσεων καθορίζεται κατά τη δημιουργία του εκάστοτε Hub και δεν μεταβάλλεται δυναμικά.

  Για την επιλογή μπορούμε να ακολουθήσουμε μερικούς κανόνες για καλύτερη εκμετάλλευση των πόρων.

  Αρχικά ο αριθμός των διαμερίσεων εξυπηρετεί την παράλληλη ανάγνωση των δεδομένων από την εφαρμογή, συνεπώς ο επιθυμητός βαθμός παραλληλίας αποτελεί έναν παράγοντα καθορισμού του αριθμού των διαμερίσεων.

  Επιπλέον, για μία διαμέριση η είσοδος και έξοδος δεδομένων έχει άνω φράγμα με επιδόσεις αντίστοιχες του 1 TU. Άρα για να εκμεταλλευτούμε στο έπακρο τους πόρους πρέπει οι διαμερίσεις να είναι τουλάχιστον ίσες με τον αριθμό των TUs.

Κατά αντιστοιχία, στο στάδιο εξόδου των δεδομένων από το σύστημα, συναντάμε ξανά το Event Hubs με τη διαφορά ότι δημιουργούμε πολλαπλά Hubs ένα για κάθε πάροχο υπηρεσιών.

10.3.2 Διάβασμα ρεύματος δεδομένων από Event Hub εισόδου

Για το διάβασμα της εισόδου από το Event Hub έχουμε τις ακόλουθες επιλογές:

1. **Διάβασμα βάσει μετατόπισης (offset):** Έχουμε τη δυνατότητα να ορίσουμε ένα απόλυτο διάστημα μεταξύ δύο σημείων και να διαβάσουμε τα δεδομένα που ανήκουν σε αυτό. Στην προκειμένη περίπτωση όμως μας ενδιαφέρουν κάθε φορά τα νέα δεδομένα που προστίθεται στο Event Hub και συνθέτουν το ρεύμα εισόδου. Επομένως σαν τιμή εκκίνησης της μετατόπισης ορίζεται το τέλος του ρεύματος δεδομένων που υπάρχει ήδη στο Event Hub χωρίς να περιορίζουμε το σημείο τερματισμού ώστε να λαμβάνονται όλα τα δεδομένα που έρχονται σε πραγματικό χρόνο.
2. **Διάβασμα βάσει αριθμού ακολουθίας (sequence number):** Μπορούμε να ορίσουμε τον αριθμό ακολουθίας των μηνυμάτων που επιθυμούμε να διαβάσουμε.
3. **Διάβασμα βάσει χρονοσφραγίδας (enqueuedTime):** Έχουμε τη δυνατότητα να ορίσουμε μία χρονική στιγμή ή και ένα χρονικό διάστημα και να διαβάσουμε τα δεδομένα βάσει αυτών των χρονικών στιγμών.

Σημειώνεται πως οι επιλογές αυτές γίνεται να εφαρμοστούν τόσο συνολικά για το Event Hub όσο και σε επίπεδο διαμέρισης του Event Hub.

Για την ανάγκη της παρούσας υλοποίησης χρησιμοποιείται η $1^{η}$ επιλογή.

10.4 Πρόσβαση στις ροές δεδομένων

Η πρόσβαση στο ρεύμα δεδομένων συνεπάγεται πρόσβαση σε πόρους του Azure Event Hubs. Μία αίτηση για κατανάλωση ή για εγγραφή δεδομένων στο Event Hubs θα πρέπει να εξουσιοδοτηθεί έτσι ώστε η υπηρεσία να γνωρίζει ότι ο αιτών πληροί τα δικαιώματα πρόσβασης.

Η εξουσιοδότηση πρόσβασης στο Event Hubs μπορεί να γίνει είτε με χρήση του Azure Active Directory είτε με χρήση υπογραφής κοινόχρηστης πρόσβασης (SAS: Shared Access Signature).

10.4.1 Εξουσιοδότηση πρόσβασης στο Azure Event Hubs με Active Directory

Κάνοντας χρήση του Active Directory ουσιαστικά δίνουμε πρόσβαση μέσω RBAC στο χρήστη ή στην εφαρμογή που αιτείται την πρόσβαση. Συγκεκριμένα μέσω του AD ορίζουμε και απονέμουμε ρόλους στην οντότητα που επιθυμεί πρόσβαση. Οι ρόλοι που απονέμουμε συνδέονται με δικαιώματα τα οποία καθορίζουν το επίπεδο πρόσβασης πάνω στου πόρους του Event Hubs.

Το επίπεδο πρόσβασης δύναται να περιοριστεί στις κατηγορίες πόρων του Event Hubs. Οι κατηγορίες πόρων παρατίθενται παρακάτω, ξεκινώντας από το χαμηλότερο και καταλήγοντας στο υψηλότερο επίπεδο πρόσβασης:

- **Consumer Group:** Η πρόσβαση περιορίζεται στην ομάδα καταναλωτών
- **Event Hub:** Η πρόσβαση περιορίζεται στην οντότητα του «Διακομιστή Γεγονότων» και λόγω ιεραρχίας και σε όλες τις ομάδες καταναλωτών που περιλαμβάνει
- **Namespace**: Η πρόσβαση περιορίζεται σε όλη την τοπολογία του Event Hubs. Για παράδειγμα, δίνεται πρόσβαση σε πολλαπλά Hubs του ίδιου ονοματοχώρου.
- **Resource group**: Η πρόσβαση περιορίζεται σε όλους του υπάρχοντες πόρους του Azure Event Hubs αρκεί να βρίσκονται στην ίδια ομάδα πόρων.
- **Subscription**: Αποτελεί το υψηλότερο επίπεδο πρόσβασης. Δίνεται πρόσβαση σε όλους του πόρους του Azure Event Hubs που περιλαμβάνει η εκάστοτε συνδρομή.

Για να αποκτήσει πρόσβαση στο Event Hubs μία οντότητα ακολουθούνται δύο βήματα.

1. Αρχικά, η οντότητα πιστοποιείται μέσω του Active Directory και σαν απάντηση στην αίτηση πρόσβασης δέχεται ένα τεκμήριο (token) σύμφωνο με το πρωτόκολλο OAuth 2.0, που αποτελεί πρότυπο για την εξουσιοδότηση πρόσβασης.
2. Σε δεύτερη φάση, το τεκμήριο πρόσβασης συμπεριλαμβάνεται στην αίτηση έτσι ώστε να λάβει την εξουσιοδότηση εφόσον είναι έγκυρο.[48]

10.4.2 Εξουσιοδότηση πρόσβασης στο Azure Event Hubs με SAS

Η εξουσιοδότηση βάσει κοινόχρηστης υπογραφής πρόσβασης μπορεί να χρησιμοποιηθεί για να επιτρέψουμε πρόσβαση στους πόρους του Event Hubs. Συγκεκριμένα, απαιτείται ο ορισμός μιας πολιτικής πρόσβασης η οποία έχει μοναδική ονομασία, συνδέεται με ένα σύνολο

δικαιωμάτων και ένα ζεύγος από κρυπτογραφικά κλειδιά. Το σύνολο δικαιωμάτων στην περίπτωση του Event Hubs αφορούν τις εξής ενέργειες:

1. **Διαχείριση**: Πλήρη δικαιώματα για αλλαγές στην τοπολογία του Event Hubs όπως δημιουργία και διαγραφή, αλλά και δικαιώματα ανάγνωσης και εγγραφής
2. **Ανάγνωση**: Δυνατότητα λήψης μηνυμάτων
3. **Εγγραφή**: Δυνατότητα για υποβολή μηνυμάτων

Στη συνέχεια στη μεριά του χρήστη χρησιμοποιείται το όνομα της πολιτικής και το κρυπτογραφικό κλειδί και με τη χρήση βιβλιοθήκης του Event Hubs δημιουργείται το SAS token, το οποίο χρησιμοποιείται στην αίτηση για πρόσβαση.

Το SAS token περιλαμβάνει ένα σύνολο από παραμέτρους που αφορούν την αίτηση, αλλά και μία παράμετρο που έχει δημιουργηθεί υπογράφοντας ένα υποσύνολο των παραμέτρων με το κρυπτογραφικό κλειδί. Η παράμετρος αυτή αποτελεί την υπογραφή της αίτησης και η εγκυρότητα της πιστοποιείται από την υπηρεσία ώστε να δοθεί η πρόσβαση. [49]

10.5 Πρόσβαση στα στατικά δεδομένα

Η πρόσβαση στα στατικά δεδομένα, δηλαδή στο σύνολο των δεδομένων που είναι αποθηκευμένα στο Blob Storage γίνεται μέσω του Azure Active Directory. Η διαδικασία που ακολουθείται απαρτίζεται από τα δύο βήματα που περιγράφηκαν στην παράγραφο 10.4.1.

10.6 Συσχέτιση των δεδομένων με τη συγκατάθεση

Όταν τα δεδομένα εισάγονται στο σύστημα, προκύπτει η ανάγκη να συσχετιστούν με την αντίστοιχη συγκατάθεση σύμφωνα με τη μοντελοποίηση της παραγράφου 6.1.

Για τη συσχέτιση αυτή χρησιμοποιείται η πράξη join μεταξύ του στατικού πίνακα με τη συγκατάθεση των χρηστών και του πίνακα που περιλαμβάνει τα εισαγόμενα δεδομένα ρεύματος.

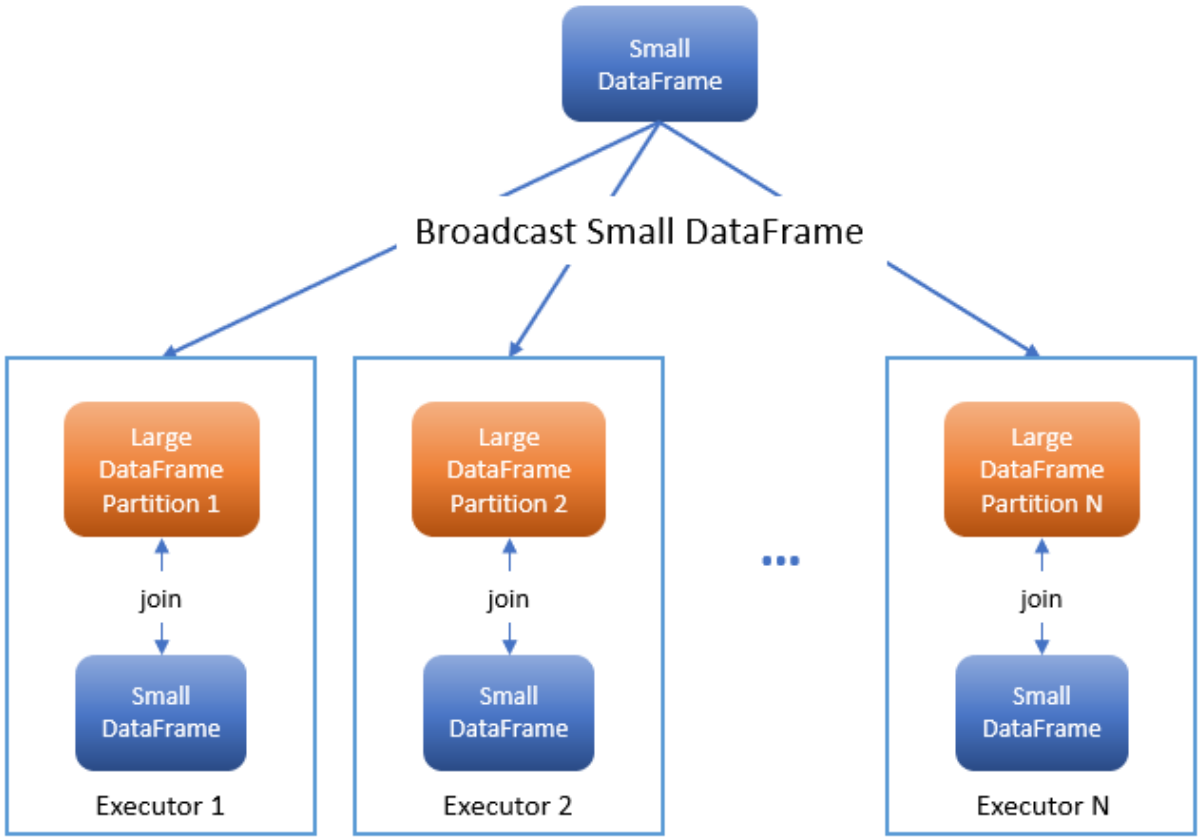
Υπάρχουν διάφοροι τρόποι εκτέλεσης της πράξης join όπως περιγράφονται στη συνέχεια [50]:

- **Broadcast Hash Join:** Ο συγκεκριμένος τρόπος εκτέλεσης join ενδείκνυται στις περιπτώσεις που ένα εκ των δύο μερών της πράξης έχει σχετικά μικρό μέγεθος.

  Στην περίπτωση αυτή ο πίνακας αυτός αποστέλλεται σε όλους τους κόμβους επεξεργασίας, και εκτελείται η πράξη hash join σε κάθε κόμβο ξεχωριστά. Σαν αποτέλεσμα αποφεύγεται η ανακατανομή στις διαμερίσεις βάσει του κλειδιού της πράξης join και επομένως και το φαινόμενο shuffling.

  Το μέγεθος για να θεωρηθεί ένας πίνακας σχετικά μικρός είναι προεπιλεγμένο στα 10MB, μπορεί όμως να μεταβληθεί με εντολή του χρήστη αλλά θα πρέπει στην απόφαση αυτή να ληφθούν υπόψιν οι πόροι των κόμβων επεξεργασίας αλλά και το γεγονός ότι η αποστολή προς όλους του κόμβους προκαλεί κίνηση στο δίκτυο που για μεγάλες τιμές δεδομένων μπορεί να αποτελέσει αποτρεπτικό παράγοντα.

Στην παρακάτω εικόνα παρουσιάζεται ένα παράδειγμα στο οποίο εκτελείται η πράξη join μεταξύ δύο DataFrames.



Εικόνα 26: Broadcast Hash join

- **Shuffle Hash Join:** Στο συγκεκριμένο τρόπο εκτέλεσης, τα δεδομένα θα ανακατανεμηθούν στις διαμερίσεις με τέτοιο τρόπο έτσι ώστε τα στοιχεία των DataFrames που έχουν το ίδιο κλειδί πάνω στο οποίο γίνεται η πράξη hash join να βρεθούν στην ίδια διαμέριση. Αυτό έχει σαν αποτέλεσμα τη μεταφορά δεδομένων ανάμεσα στις διαμερίσεις ή ακόμα και ανάμεσα στους διάφορους κόμβους εργασίας όταν αυτές δεν βρίσκονται στον ίδιο κόμβο. Επίσης, η μέθοδος αυτή εξαιτίας της μεθόδου hash join που εκτελείται, απαιτεί η μία πλευρά της πράξης να φορτώνεται στη μνήμη, όχι η ίδια αλλά ο πίνακας κατακερματισμού της (hash table), και επομένως αποδίδει σε περιπτώσεις που το μέγεθός της, της επιτρέπει να φορτωθεί στη μνήμη.
- **Sort Merge Join:** Σύμφωνα με τη συγκεκριμένη μέθοδο, προηγείται η ταξινόμηση των δεδομένων και επομένως προκαλείται ανακατανομή στις διαμερίσεις βάσει του κλειδιού. Έπειτα δεν εκτελείται πράξη join αλλά μία πράξη συνένωσης με κατάλληλο αλγόριθμο. Αποτελεί τον προεπιλεγμένο τρόπο για την πράξη join.

Κάνοντας χρήση του DataFrame API, τα ερωτήματα πάνω στα δεδομένα, συμπεριλαμβανομένου του παραπάνω join, αναλύονται και με τη βοήθεια του Catalyst Optimizer αποφασίζεται ο βέλτιστος τρόπος εκτέλεσης.

Έτσι και στην προκειμένη περίπτωση προτιμάται η μέθοδος join που ταιριάζει καλύτερα στα δεδομένα του προβλήματος, εφόσον δεν έχει δηλωθεί ρητά ποια μέθοδος να χρησιμοποιηθεί.

10.7 Επεξεργασία των δεδομένων

Στην υλοποίηση προτιμήθηκε η χρήση του DataFrame API καθώς τα δεδομένα ήταν δομημένα. Η επεξεργασία των δεδομένων όπως περιγράφηκε και στην ενότητα 6.1, μπορεί να χωριστεί σε δύο στάδια.

Αρχικά, στο στάδιο της συσχέτισης των εισαγόμενων δεδομένων με τη συγκατάθεση των χρηστών και στη συνέχεια στο στάδιο με το μετασχηματισμό βάσει της συγκατάθεσης αυτής.

Για την επεξεργασία του ρεύματος δεδομένων διακρίνουμε τις εξής περιπτώσεις [51].

- **Προεπιλογή**: Η επεξεργασία εκτελείται σε micro-batches, τα οποία προκύπτουν αμέσως μόλις ολοκληρωθεί η επεξεργασία του τελευταίου micro-batch
- **Micro-batches σε σταθερά διαστήματα**: Ο χρήστης ορίζει ένα χρονικό διάστημα, στο οποίο επεξεργάζονται τα δεδομένα που εισέρχονται ως ένα micro-batch. Στην περίπτωση που η επεξεργασία ολοκληρωθεί πρωτύτερα της παρόδου του διαστήματος προκύπτει αναμονή μέχρι την αρχή του επόμενου διαστήματος. Στην περίπτωση που δεν ολοκληρωθεί η επεξεργασία προκύπτει καθυστέρηση του επόμενου micro-batch έως ότου ολοκληρωθεί το τρέχων. Τέλος στην περίπτωση που δεν υπάρχουν δεδομένα το micro-batch δεν εκτελείται.
- **Micro-batch μιας εκτέλεσης**: Δυνατότητα για επεξεργασία όλων των διαθέσιμων δεδομένων και έπειτα λήξης της επεξεργασίας.
- Συνεχόμενη επεξεργασία του ρεύματος δεδομένων.

Για τις ανάγκες της υλοποίησης χρησιμοποιήθηκε η micro-batch επεξεργασία.

10.7.1 Στάδιο συσχέτισης

Στο στάδιο της συσχέτισης συναντάμε την πράξη της σύνδεσης "join". Εξαιτίας του γεγονότος, ότι η πράξη θα γίνεται για κάθε micro-batch δεδομένων που εισάγονται προτείνεται και υλοποιείται η αποθήκευση των δεδομένων της συγκατάθεσης των χρηστών στην cache εφόσον το μέγεθός τους δεν δημιουργεί πρόβλημα.

Στην περίπτωση αυτή επισημαίνεται ότι σε πραγματικό σύστημα θα πρέπει να υλοποιηθεί και μηχανισμός συνέπειας, ώστε όταν αλλάζουν τα δεδομένα της συγκατάθεσης να ενημερώνεται και η cache. Η απαίτηση αυτή μπορεί να ικανοποιηθεί με τη χρήση μίας υπηρεσίας cache, όπως είναι η υπηρεσία Redis Cache που παρέχει αποθήκευση και προσπέλαση δεδομένων σε γρήγορη μνήμη.

Στη συνέχεια η πράξη "join" θα πραγματοποιηθεί σύμφωνα με το πλάνο εκτέλεσης της Spark SQL μηχανής και σαν αποτέλεσμα θα προκύψει ένα DataFrame που αποτελείται όπως φαίνεται στο παράδειγμα από την παρακάτω εγγραφή:

Time	VehicleID	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density_y	Συνέχεια εγγραφής στον Πίνακας 3
0	3	53	100	600867	5676073	95079	53	1	

Πίνακας 2: Παράδειγμα εγγραφής δεδομένων και διανυσμάτων συγκατάθεσης (1/2)

Συνέχεια εγγραφής από Πίνακας 2	Time Consent	Speed Consent	Dist_front Consent	xCoord Consent	yCoord Consent	SectionID Consent	Avg_Sect_Speed Consent	Sect_Density_y Consent
	(0,0,1)	(1,0,1)	(0,0,0)	(1,1,0)	(1,1,0)	(0,1,1)	(1,0,1)	(0,0,1)

Πίνακας 3: Παράδειγμα εγγραφής δεδομένων και διανυσμάτων συγκατάθεσης (2/2)

10.7.2 Στάδιο μετασχηματισμού

Η διαδικασία του μετασχηματισμού έχει ως αποτέλεσμα τη δημιουργία ενός νέου DataFrame, καθώς τα DataFrames/Datasets/RDDs είναι δομές που δεν μπορούν να μεταβληθούν (immutable). Η διαδικασία μετασχηματισμού θα γίνει τόσες φορές όσο και το πλήθος των παρόχων υπηρεσιών.

Η μέθοδος που ακολουθείται δημιουργεί ένα νέο DataFrame με ίδιες ακριβώς στήλες με αυτές των δεδομένων ρεύματος, μόνο που το περιεχόμενο των στηλών λαμβάνει τιμή βάσει της αντίστοιχης συγκατάθεσης χρήστη και του παρόχου για τον οποίο πραγματοποιείται ο μετασχηματισμός.

Συγκεκριμένα, η δημιουργία του νέου DataFrame γίνεται με την προσθήκη νέας στήλης για κάθε κατηγορία δεδομένων του οχήματος η οποία αντικαθιστά την αρχική, αξιοποιώντας την παράλληλη στήλη η οποία φέρει τη συγκατάθεση δεδομένων για αυτήν την κατηγορία.

Για την επεξεργασία του ρεύματος των δεδομένων ανά πάροχο υπηρεσιών απονέμεται στον εκάστοτε πάροχο ένα δυαδικό διάνυσμα που αποτελεί δύναμη του 2. $(2^n, n = 0,1,2..)$ το οποίο χρησιμοποιείται για την επιλογή του επιθυμητού bit του δυαδικού διανύσματος συγκατάθεσης ανά κατηγορία δεδομένων και εφαρμόζεται μέσω της πράξης σε επίπεδο bit AND μεταξύ των δυαδικών διανυσμάτων συγκατάθεσης και παρόχου. Σε περίπτωση που το αποτέλεσμα της πράξης ταυτίζεται με το δυαδικό διάνυσμα του παρόχου υπηρεσιών τότε το πεδίο δεδομένων διατηρείται στη νέα στήλη που δημιουργείται, διαφορετικά καταργείται και αντί αυτού προστίθεται η τιμή "-1" δηλώνοντας μη εξουσιοδότηση πρόσβασης.

Στη συνέχεια ακολουθεί ενδεικτικό παράδειγμα για μία κατηγορία δεδομένων (Speed) και τρεις παρόχους υπηρεσιών.

		Speed Consent bitwise And					
Speed	Speed Consent	Πάροχος 1 ($2^0 = 001_2$)	Πάροχος 2 ($2^1 = 010_2$)	Πάροχος 3 ($2^2 = 100_2$)	Speed (πάροχος 1)	Speed (πάροχος 2)	Speed (πάροχος 3)
54	101	00**1**	0**0**0	**1**00	54	-1	54
14	001	00**1**	0**0**0	**0**00	14	-1	-1
65	011	00**1**	0**1**0	**0**00	65	65	-1
29	010	00**0**	0**1**0	**0**00	-1	29	-1
40	110	00**0**	0**1**0	**1**00	-1	40	40
87	101	00**1**	0**0**0	**1**00	87	-1	87

Εικόνα 27: Ενδεικτικό παράδειγμα μετασχηματισμού ρεύματος δεδομένων

10.8 Μέτρηση καθυστέρησης

Το γεγονός ότι τα δεδομένα καταφθάνουν ως ρεύμα στο σύστημά μας αλλά και επεξεργάζονται συνεχόμενα ως micro-batches, καθιστά απαραίτητο το μετασχηματισμό των δεδομένων ώστε να γίνει η μέτρηση της καθυστέρησης του συστήματος. Για να το πετύχουμε αυτό, θα εκμεταλλευτούμε τη δομή των μηνυμάτων στο Event Hubs.

Μόλις ένα μήνυμα εισέρχεται στο Event Hub τότε αποκτά ένα πεδίο με την τρέχουσα χρονική στιγμή της μορφής "2019-10-18T16:30:25.599+0000Z".

Στη συνέχεια, ακολουθεί η επεξεργασία και στο σημείο αυτό δημιουργείται μία νέα στήλη η οποία φέρει τις χρονοσφραγίδες εισόδου για κάθε μήνυμα. Κατόπιν της επεξεργασίας, τα δεδομένα θα εγγραφούν στα Event Hubs της εξόδου και εκεί θα αποκτήσουν τις χρονοσφραγίδες εξόδου.

Τελικά, για να μετρήσουμε την καθυστέρηση του κάθε μηνύματος στο σύστημα θα πρέπει να αφαιρέσουμε από τις χρονοσφραγίδες εξόδου τις χρονοσφραγίδες εισόδου.

Για να πραγματοποιηθεί αυτό, αρχικά διαβάζονται τα δεδομένα από τα Event Hubs εξόδου, προστίθεται σε αυτά μία στήλη με τις χρονοσφραγίδες εξόδου και έπειτα δημιουργείται μία νέα στήλη με το αποτέλεσμα της αφαίρεσης. Για να αφαιρέσουμε τις χρονοσφραγίδες τις μετατρέπουμε σε milliseconds. Ως τελικό αποτέλεσμα, παίρνουμε το μέσο όρο των αποτελεσμάτων και έτσι προκύπτει η μέση καθυστέρηση για ένα μήνυμα.

10.9 Περιορισμοί υλοποίησης

Στη παρούσα υλοποίηση το ρεύμα των δεδομένων επεξεργάζεται σε micro-batches και αυτό επιβάλλει στην καθυστέρηση της επεξεργασίας ένα κάτω φράγμα των 100 ms.

Πριν την ολοκλήρωση της υλοποίησης, θεωρήθηκε προτιμητέα η χρήση της συνεχόμενης επεξεργασίας γνωρίζοντας ότι το κάτω φράγμα μπορεί να κατέβει στο 1ms. Ωστόσο, κατόπιν ανάλυσης αποδείχθηκε μη υλοποιήσιμη επιλογή.

Η υλοποίηση συνεχόμενης επεξεργασίας θα απαιτούσε τη χρήση του πρωτοκόλλου Kafka για αποστολή των δεδομένων. Η συγκεκριμένη απαίτηση ικανοποιείται μέσω της δημιουργίας ενός Kafka Enabled Event Hub. Ωστόσο το σημείο που θέτει περιορισμό σε αυτή την προσέγγιση, είναι ότι στα υποστηριζόμενα ερωτήματα πάνω στα ρεύματα δεδομένων δεν βρίσκεται η πράξη join που χρησιμοποιείται στην υλοποίηση. Συνεπώς δεν μπορεί να υλοποιηθεί η προτεινόμενη λύση.

Σημειώνεται ότι κάτι τέτοιο δεν αναφέρεται ρητά στην τεκμηρίωση του Continuous Processing Mode [36] ,και ο παραπάνω περιορισμός εντοπίστηκε στην προσπάθεια υλοποίησής του.

11 Παράδειγμα λειτουργίας και αποτελέσματα

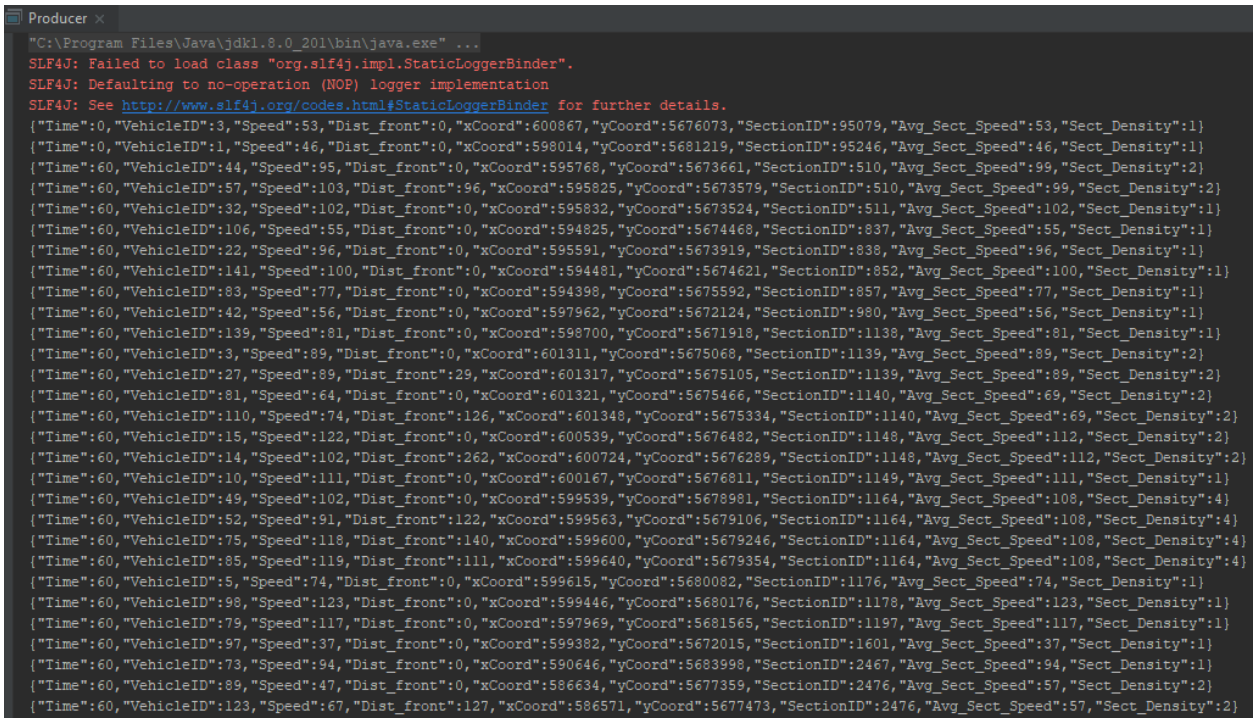
11.1 Σενάριο χρήσης

Για την επίδειξη λειτουργίας του συστήματος παρουσιάζεται ένα παράδειγμα λειτουργίας του συστήματος, στο οποίο γίνονται τα παρακάτω βήματα - στάδια:

1. Αποστέλλουμε τα δεδομένα από τον τοπικό υπολογιστή στο Event Hub εισόδου του συστήματος στην πλατφόρμα Azure
2. Διαβάζουμε τα δεδομένα που καταφθάνουν και τα εμφανίζουμε στην οθόνη, αποκλειστικά για την επίδειξη της λειτουργίας
3. Παρουσιάζονται επίσης τα δεδομένα έπειτα από την ένωση με τα στοιχεία συγκατάθεσης
4. Εγγράφονται τα δεδομένα ύστερα από την επεξεργασία σε τρία διαφορετικά Event Hubs ένα για κάθε πάροχο από τους τρεις στο εν λόγω παράδειγμα
5. Διαβάζονται και εμφανίζονται τα δεδομένα από τα Event Hubs των παρόχων υπηρεσιών και υπολογίζεται η μέση τιμή της καθυστέρησης των μηνυμάτων.
6. Διαβάζονται και εμφανίζονται τα δεδομένα που αποθηκεύονται στο Blob Storage για μακροπρόθεσμη διατήρηση στο σύστημα

11.2 Αποστολή δεδομένων από τον τοπικό υπολογιστή

Στην εικόνα που ακολουθεί φαίνεται ένα απόσπασμα των δεδομένων που αποστέλλεται από τον τοπικό υπολογιστή, στο οποίο βλέπουμε τη JSON μορφή των δεδομένων. Τα δεδομένα που φαίνονται αφορούν το JSON αντικείμενο που κατασκευάζεται διαβάζοντας την κάθε γραμμή του αρχείου δεδομένων των οχημάτων. Στην πραγματικότητα τα δεδομένα αποστέλλονται αφού πρώτα σειριοποιηθούν, και όχι όπως φαίνονται παρακάτω.

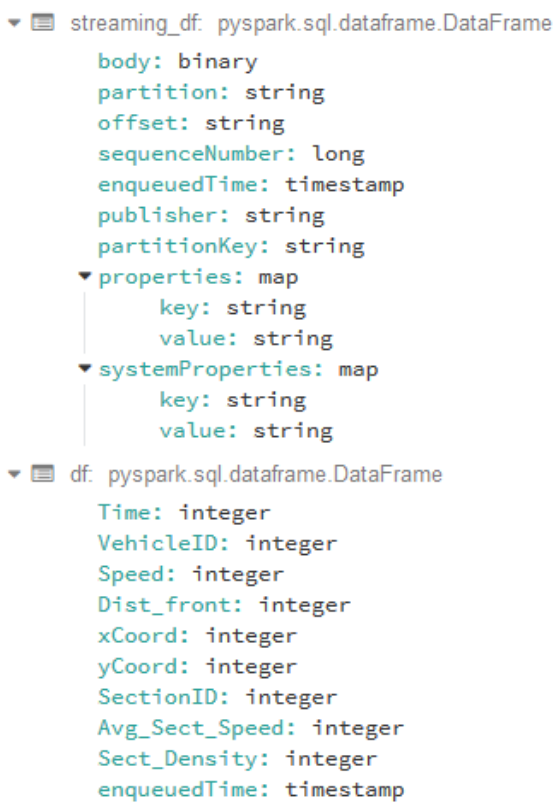


Εικόνα 28: Αποστολή δεδομένων από τοπικό υπολογιστή

11.3 Εμφάνιση ρεύματος δεδομένων

Αρχικά τα δεδομένα εισέρχονται στο Event Hub και επομένως κάθε μήνυμα που θα διαβαστεί θα ακολουθεί την τυπική δομή του Event Hubs. Στην εφαρμογή, μας ενδιαφέρουν τα δεδομένα που αποστέλλονται τα οποία βρίσκονται στο πεδίο “body”.

Πρώτα πρέπει να μετατραπούν από τη δυαδική μορφή που βρίσκονται σε JSON αντικείμενο και στη συνέχεια να αναλυθούν συντακτικά ώστε να προκύψουν οι επιθυμητές στήλες του DataFrame. Επίσης, συμπεριλαμβάνεται η χρονοσφραγίδα εισόδου που προσθέτει το Event Hub σε κάθε μήνυμα-γεγονός ως “enqueuedTime” για να χρησιμοποιηθεί στη μέτρηση της καθυστέρησης όπως περιγράφηκε**.** Στη συνέχεια παρατίθεται η εικόνα που δείχνει το σχήμα των δεδομένων εισόδου ως “streaming_df” και το σχήμα των δεδομένων μετά τον παραπάνω μετασχηματισμό ως “df”.



Εικόνα 29: Σχήμα δεδομένων εισόδου (Event Hubs) προ / μετά μετασχηματισμού

Έπειτα, εμφανίζουμε τα δεδομένα που εισέρχονται στο Event Hub στην οθόνη και ακολουθεί ενδεικτική εικόνα με τμήμα των εισηγμένων δεδομένων.

Cancel

(1) Spark Jobs

display_query_2 (id: 8ce934b0-56c0-4b28-bf46-a0d271d8b77f) *Last updated: 5 seconds ago*

Time	VehicleID	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density	enqueuedTime
0	3	53	0	600867	5676073	95079	53	1	2019-10-19T17:34:15.209+0000
0	1	46	0	598014	5681219	95246	46	1	2019-10-19T17:34:15.319+0000
60	44	95	0	595768	5673661	510	99	2	2019-10-19T17:34:15.428+0000
60	57	103	96	595825	5673579	510	99	2	2019-10-19T17:34:15.522+0000
60	32	102	0	595832	5673524	511	102	1	2019-10-19T17:34:15.615+0000
60	106	55	0	594825	5674468	837	55	1	2019-10-19T17:34:15.725+0000
60	22	96	0	595591	5673919	838	96	1	2019-10-19T17:34:15.819+0000
60	141	100	0	594481	5674621	852	100	1	2019-10-19T17:34:15.912+0000
60	83	77	0	594398	5675592	857	77	1	2019-10-19T17:34:16.006+0000
60	42	56	0	597962	5672124	980	56	1	2019-10-19T17:34:16.100+0000
60	139	81	0	598700	5671918	1138	81	1	2019-10-19T17:34:16.194+0000
60	3	89	0	601311	5675068	1139	89	2	2019-10-19T17:34:16.303+0000

Εικόνα 30: Δεδομένα εισόδου (Event Hubs)

11.4 Εμφάνιση ρεύματος δεδομένων και συγκατάθεσης χρηστών

Προτού παρουσιάσουμε το αποτέλεσμα της πράξης join μεταξύ του ρεύματος εισόδου και των στατικών δεδομένων της συγκατάθεσης, παραθέτουμε ένα υποσύνολο των δεδομένων της συγκατάθεσης στην παρακάτω εικόνα.

Σημειώνεται ότι στις στήλες με όνομα «XXX_consent» βρίσκονται οι επιλογές του χρήστη ως ακέραιοι αριθμοί. Η δυαδική αναπαράσταση των αριθμών αυτών συμβολίζει το δυαδικό διάνυσμα επίτρεψης που περιγράφηκε στη μοντελοποίηση του προβλήματος.

▸ (1) Spark Jobs

Time_consent	Speed_consent	Dist_front_consent	xCoord_consent	yCoord_consent	SectionID_consent	Avg_Sect_Speed_consent	Sect_Density_consent	VehicleID
39	49	44	48	3	23	19	32	31
36	0	47	28	13	58	61	2	85
24	27	50	7	22	23	1	23	137
56	45	9	52	59	56	41	46	65
44	46	59	50	60	43	13	10	53
42	53	26	49	35	7	5	12	133
15	34	16	33	5	3	45	8	78
45	29	24	12	31	36	63	18	108
29	45	6	13	13	24	1	40	34
10	41	44	2	62	63	21	11	126
13	40	7	43	32	39	20	63	115
50	48	2	17	1	32	19	5	101
14	56	35	15	27	24	28	62	81
3	4	19	50	33	45	31	8	28
7	36	0	52	28	23	6	6	76
11	37	25	47	49	46	6	61	27
57	20	7	63	19	17	2	35	26
9	31	40	30	42	11	16	31	44
28	58	24	60	3	53	40	49	103
13	33	16	30	47	21	37	29	12

only showing top 20 rows

Εικόνα 31: Δεδομένα συγκατάθεσης

Στη συνέχεια, παρατίθεται το αποτέλεσμα της πράξης join, που περιλαμβάνει τα δεδομένα και τη σχετική συγκατάθεση. Δίνεται σε 2 εικόνες, που η δεύτερη αποτελεί τη συνέχεια της πρώτης.

Cancel

▾ (1) Spark Jobs

▸ Job 321 View (2 stages)

▸ display_query_4 (id: 3b2decc0-800a-4c9f-85e4-5c7fa7c54719) *Last updated: 5 seconds ago*

VehicleID	Time	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density	enqueuedTime	Time_consent
3	0	53	0	600867	5676073	95079	53	1	2019-10-19T18:01:44.489+0000	10
1	0	46	0	598014	5681219	95246	46	1	2019-10-19T18:01:44.598+0000	45
44	60	95	0	595768	5673661	510	99	2	2019-10-19T18:01:44.692+0000	9
57	60	103	96	595825	5673579	510	99	2	2019-10-19T18:01:44.785+0000	16
32	60	102	0	595832	5673524	511	102	1	2019-10-19T18:01:44.879+0000	41
106	60	55	0	594825	5674468	837	55	1	2019-10-19T18:01:44.973+0000	40
22	60	96	0	595591	5673919	838	96	1	2019-10-19T18:01:45.067+0000	4
141	60	100	0	594481	5674621	852	100	1	2019-10-19T18:01:45.145+0000	60
83	60	77	0	594398	5675592	857	77	1	2019-10-19T18:01:45.239+0000	29
42	60	56	0	597962	5672124	980	56	1	2019-10-19T18:01:45.332+0000	5

Εικόνα 32: Αποτέλεσμα πράξης join (1/2)

Time_consent	Speed_consent	Dist_front_consent	xCoord_consent	yCoord_consent	SectionID_consent	Avg_Sect_Speed_consent	Sect_Density_consent
10	48	17	60	40	40	46	14
45	28	29	50	28	15	28	51
9	31	40	30	42	11	16	31
16	52	6	5	12	13	14	43
41	19	5	22	4	52	2	11
40	62	20	24	36	6	38	44
4	45	8	39	62	34	59	0
60	14	3	23	12	5	24	1
29	4	59	24	29	4	63	25
5	59	25	20	16	51	9	14

Εικόνα 33: Αποτέλεσμα πράξης join (2/2)

Τέλος, παρατίθεται και το σχήμα των δεδομένων όπως φαίνεται παρακάτω:

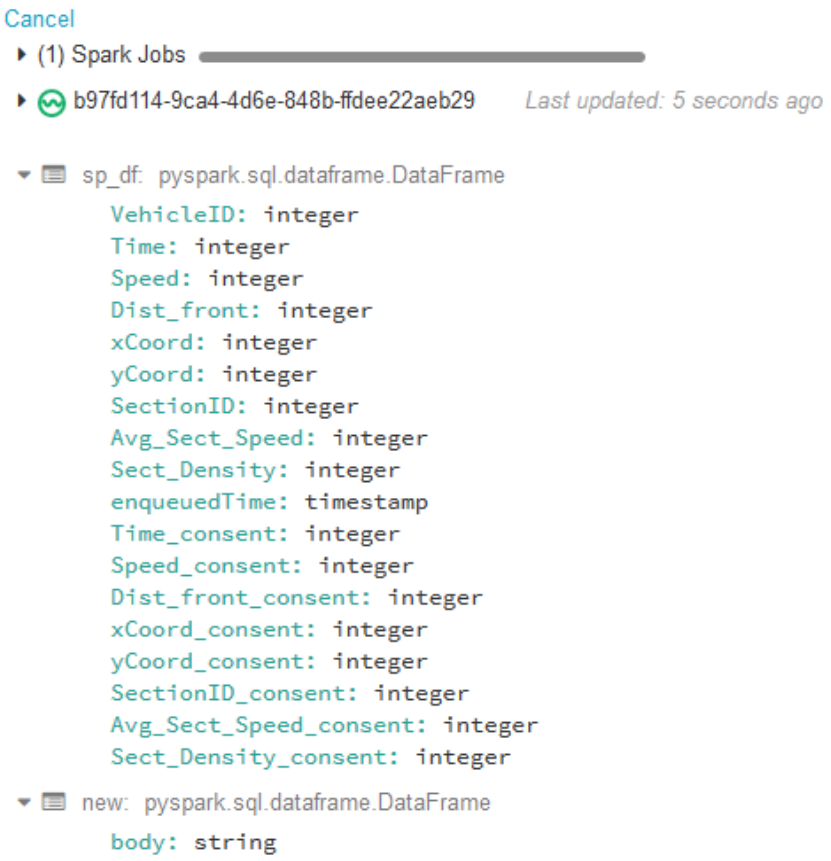
```
joinStream: pyspark.sql.dataframe.DataFrame
    VehicleID: integer
    Time: integer
    Speed: integer
    Dist_front: integer
    xCoord: integer
    yCoord: integer
    SectionID: integer
    Avg_Sect_Speed: integer
    Sect_Density: integer
    enqueuedTime: timestamp
    Time_consent: integer
    Speed_consent: integer
    Dist_front_consent: integer
    xCoord_consent: integer
    yCoord_consent: integer
    SectionID_consent: integer
    Avg_Sect_Speed_consent: integer
    Sect_Density_consent: integer
```

Εικόνα 34: Σχήμα δεδομένων πράξης join

11.5 Αποστολή δεδομένων κατόπιν επεξεργασίας

Κατόπιν επεξεργασίας, τα δεδομένα γράφονται στα Event Hubs εξόδου που αντιστοιχούν στους παρόχους υπηρεσιών.

Στην εικόνα που ακολουθεί παρουσιάζεται η εγγραφή της εξόδου για τον έναν από τους τρεις παρόχους. Παρατηρούμε ότι τα δεδομένα διατηρούν το σχήμα τους (“sp_df”) και στη συνέχεια μετασχηματίζονται σε μία συμβολοσειρά JSON (“new”) ώστε να αποσταλούν στο κατάλληλο Event Hub εξόδου.



Εικόνα 35: Σχήμα δεδομένων εξόδου

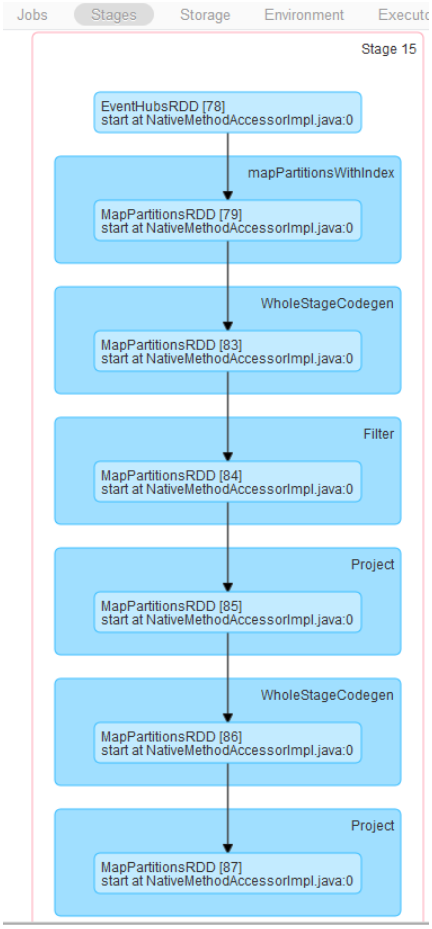
Αναλύοντας το φυσικό πλάνο εκτέλεσης, παρατηρείται ότι επιλέγεται η μέθοδος broadcast hash join ως η βέλτιστη λύση για την πράξη join.

```
1 ds.explain()
```

```
== Physical Plan ==
Project [structstojson(named_struct(VehicleID, VehicleID#534, Time, CASE WHEN ((Time_consent#562 & 1)
= 1) THEN Time#533 ELSE 0 END, Speed, CASE WHEN ((Speed_consent#563 & 1) = 1) THEN Speed#535 ELSE 0 E
ND, Dist_front, CASE WHEN ((Dist_front_consent#564 & 1) = 1) THEN Dist_front#536 ELSE 0 END, xCoord,
CASE WHEN ((xCoord_consent#565 & 1) = 1) THEN xCoord#537 ELSE 0 END, yCoord, CASE WHEN ((yCoord_conse
nt#566 & 1) = 1) THEN yCoord#538 ELSE 0 END, SectionID, CASE WHEN ((SectionID_consent#567 & 1) = 1) T
HEN SectionID#539 ELSE 0 END, Avg_Sect_Speed, CASE WHEN ((Avg_Sect_Speed_consent#568 & 1) = 1) THEN A
vg_Sect_Speed#540 ELSE 0 END, Sect_Density, CASE WHEN ((Sect_Density_consent#569 & 1) = 1) THEN Sect_
Density#541 ELSE 0 END, enqueuedTime, enqueuedTime#871, Time_consent, Time_consent#562, Speed_consen
t, Speed_consent#563, ... 12 more fields), Some(Etc/UTC)) AS body#863]
+- *(3) BroadcastHashJoin [VehicleID#534], [VehicleID#570], Inner, BuildRight
   :- Project [jsontostructs(StructField(Time,IntegerType,true), StructField(VehicleID,IntegerType,tr
ue), StructField(Speed,IntegerType,true), StructField(Dist_front,IntegerType,true), StructField(xCoor
d,IntegerType,true), StructField(yCoord,IntegerType,true), StructField(SectionID,IntegerType,true), S
tructField(Avg_Sect_Speed,IntegerType,true), StructField(Sect_Density,IntegerType,true), cast(body#86
7 as string), Some(Etc/UTC)).Time AS Time#533, jsontostructs(StructField(Time,IntegerType,true), Stru
```

Εικόνα 36: Πλάνο εκτέλεσης επεξεργασίας

Επιπλέον, όπως βλέπουμε στην παρακάτω εικόνα, όλοι οι μετασχηματισμοί συμβαίνουν στο ίδιο στάδιο, και επομένως το join με τις πιθανές εξαρτήσεις υπό την ευρεία έννοια δεν δημιουργεί μεταφορά των δεδομένων μεταξύ των διαμερίσεων. Αυτό συμβαίνει διότι ακολουθείται η μέθοδος broadcast hash join.



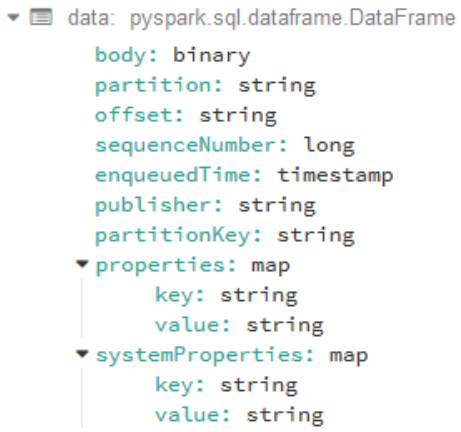
Εικόνα 37: Μετασχηματισμοί δεδομένων

11.6 Εμφάνιση αποτελεσμάτων επεξεργασίας ανά πάροχο υπηρεσιών

Για να παρουσιάσουμε τα αποτελέσματα, θα πρέπει να διαβάσουμε τα αποτελέσματα που γράφονται στα τρία διαφορετικά Event hubs για το εν λόγω παράδειγμα.

Για να το πετύχουμε, θα διαβάσουμε τα δεδομένα σαν ένα σύνολο στατικών δεδομένων, καθορίζοντας το εύρος των δεδομένων που διαβάζουμε βάσει των μεθόδων που περιγράφηκε στην ενότητα 10.3.2.

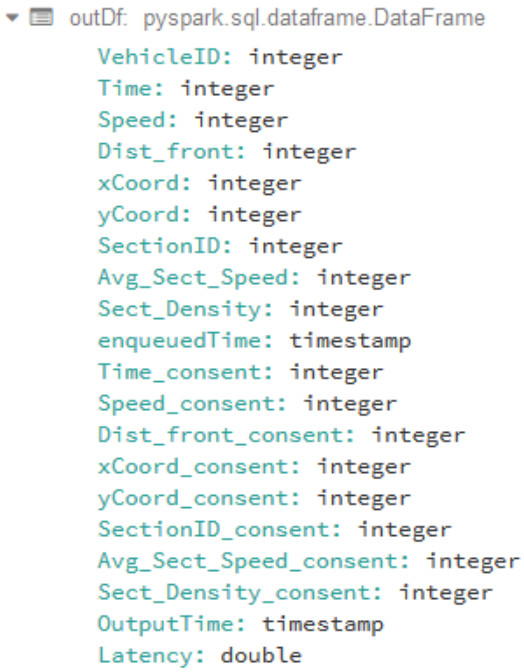
Αρχικά παρουσιάζουμε παρακάτω το σχήμα των δεδομένων αφού διαβαστούν από το Event hub εξόδου:



Εικόνα 38: Σχήμα δεδομένων Event Hub εξόδου

Όπως και στην περίπτωση της εισόδου τα επεξεργαζόμαστε ώστε να προκύψει το επιθυμητό DataFrame. Στην παρούσα περίπτωση το πεδίο "enqueuedTime" αποτελεί τη χρονοσφραγίδα εισόδου στο Event Hub της εξόδου και επομένως τη χρονοσφραγίδα εξόδου για το σύστημά μας "OutputTime".

Το νέο σχήμα των δεδομένων φαίνεται στην εικόνα που ακολουθεί:



Εικόνα 39: Σχήμα δεδομένων εξόδου

Στη συνέχεια φαίνεται ένα υποσύνολο των δεδομένων που λαμβάνει ο κάθε πάροχος.

Εξαιτίας της ανάγκης παράθεσης του σε πολλαπλές εικόνες, επαναλαμβάνεται μεταξύ των εικόνων η στήλη "Time_consent" για βοήθεια στον αναγνώστη.

1ος Πάροχος

VehicleID	Time	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density	enqueuedTime	Time_consent
10	60	111	-1	-1	5676811	1149	111	-1	2019-10-20T23:00:35.891+0000	19
52	-1	-1	-1	-1	5679106	1164	-1	4	2019-10-20T23:00:36.079+0000	40
85	-1	-1	111	-1	5679354	-1	108	-1	2019-10-20T23:00:36.282+0000	36
98	60	-1	0	-1	-1	1178	-1	1	2019-10-20T23:00:36.469+0000	29
97	60	-1	0	599382	-1	-1	-1	-1	2019-10-20T23:00:36.657+0000	23
89	60	47	-1	-1	5677359	2476	57	2	2019-10-20T23:00:36.844+0000	9
24	-1	-1	0	-1	5677517	4812	56	1	2019-10-20T23:00:37.032+0000	52
130	60	106	0	598247	-1	-1	-1	-1	2019-10-20T23:00:37.219+0000	29
138	60	-1	-1	-1	5684550	-1	-1	1	2019-10-20T23:00:37.422+0000	25
13	-1	111	0	599518	-1	41456	-1	1	2019-10-20T23:00:37.610+0000	28
12	60	93	-1	-1	5672816	48187	93	1	2019-10-20T23:00:37.797+0000	13
126	-1	56	-1	-1	-1	63060	56	1	2019-10-20T23:00:38.001+0000	10
82	60	82	-1	598008	-1	72104	-1	1	2019-10-20T23:00:38.188+0000	55

Εικόνα 40: Πάροχος 1 - Δεδομένα εξόδου (1/2)

Time_consent	Speed_consent	Dist_front_consent	xCoord_consent	yCoord_consent	SectionID_consent	Avg_Sect_Speed_consent	Sect_Density_consent	OutputTime	Latency
19	37	54	34	47	11	9	2	2019-10-20T23:00:36.836+0000	0.94499993324279
40	20	58	38	39	59	14	13	2019-10-20T23:00:36.852+0000	0.773000001907341
36	0	47	28	13	58	61	2	2019-10-20T23:00:37.696+0000	1.41400003433227
29	42	25	58	22	15	36	55	2019-10-20T23:00:37.711+0000	1.24199986457824
23	60	17	11	36	8	54	10	2019-10-20T23:00:37.711+0000	1.05399990081787
9	33	46	54	17	51	51	53	2019-10-20T23:00:37.711+0000	0.86699986457824
52	46	35	62	25	61	53	3	2019-10-20T23:00:37.711+0000	0.67899990081787
29	43	3	63	38	28	42	38	2019-10-20T23:00:38.534+0000	1.31499981880187
25	50	60	10	55	52	14	21	2019-10-20T23:00:38.550+0000	1.12800002098083
28	47	35	53	54	9	52	11	2019-10-20T23:00:38.550+0000	0.940000057220459
13	33	16	30	47	21	37	29	2019-10-20T23:00:38.550+0000	0.753000020980835
10	41	44	2	62	63	21	11	2019-10-20T23:00:39.394+0000	1.39300012588500
55	17	18	43	40	49	10	51	2019-10-20T23:00:39.409+0000	1.22099996137809

Εικόνα 41: Πάροχος 1 - Δεδομένα εξόδου (2/2)

Επεξήγηση δεδομένων

Στην πρώτη γραμμή του πίνακα βλέπουμε ότι το πεδίο “Dist_front” έχει πάρει την τιμή “-1” κάτι που φανερώνει ότι για τον συγκεκριμένο πάροχο δεν υπήρχε επίτρεψη αποστολής από το χρήστη.

Αυτό το επιβεβαιώνουμε αν ανατρέξουμε στη στήλη “Dist_front_consent”, στην οποία βλέπουμε τον αριθμό “54” ο οποίος αντιστοιχεί στον 0011011**0**” σε δυαδική μορφή με το πρώτο bit που αφορά τον πρώτο πάροχο να είναι 0 και επομένως να φανερώνει αποκλεισμό του παρόχου αυτού.

Σε αντίθεση στην ίδια γραμμή βλέπουμε ότι το πεδίο “Speed” έχει την τιμή “111”. Ανατρέχοντας στην στήλη “Speed_consent” βλέπουμε την τιμή “37” η οποία στο δυαδικό είναι η “0010010**1**” με το πρώτο bit να υποδεικνύει την επίτρεψη.

Σημειώνεται ότι οι στήλες με τη συγκατάθεση δε διαγράφηκαν για λόγους παρουσίασης και επαλήθευσης των δεδομένων χωρίς να έχουν κάποια αξία για τον πάροχο υπηρεσιών.

Αντίστοιχα παρουσιάζονται και τα αποτελέσματα της επεξεργασίας για τους άλλους δύο παρόχους του εν λόγω παραδείγματος.

2ος Πάροχος

VehicleID	Time	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density	enqueuedTime	Time_consent
71	-1	80	-1	-1	-1	-1	88	3	2019-10-20T22:59:48.377+0000	33
54	-1	54	0	-1	-1	-1	56	4	2019-10-20T22:59:48.564+0000	21
50	60	-1	-1	595202	5674315	94916	56	-1	2019-10-20T22:59:48.752+0000	18
59	60	46	0	595071	5674386	-1	-1	-1	2019-10-20T22:59:48.940+0000	47
84	60	-1	141	-1	-1	94917	48	3	2019-10-20T22:59:49.127+0000	11
78	60	84	-1	-1	-1	94982	-1	-1	2019-10-20T22:59:49.330+0000	15
35	60	-1	97	-1	5672144	-1	-1	-1	2019-10-20T22:59:49.518+0000	35
115	-1	-1	79	597189	-1	94991	-1	2	2019-10-20T22:59:49.799+0000	13
77	-1	104	-1	597620	5672322	94988	104	1	2019-10-20T22:59:49.612+0000	4
58	-1	-1	86	601301	5675408	95039	103	2	2019-10-20T22:59:50.080+0000	20
28	60	-1	28	600841	-1	-1	96	-1	2019-10-20T22:59:50.268+0000	3
100	60	85	0	-1	5675900	95042	-1	-1	2019-10-20T22:59:50.456+0000	22

Εικόνα 42: Πάροχος 2 - Δεδομένα εξόδου (1/2)

Time_consent	Speed_consent	Dist_front_consent	xCoord_consent	yCoord_consent	SectionID_consent	Avg_Sect_Speed_consent	Sect_Density_consent	OutputTime	Latency
33	11	12	4	45	41	35	39	2019-10-20T22:59:49.822+0000	1.44499993324279
21	27	2	12	0	13	22	54	2019-10-20T22:59:49.838+0000	1.27400016784667
18	53	20	42	11	10	30	12	2019-10-20T22:59:49.838+0000	1.08599996566772
47	31	2	19	63	4	40	33	2019-10-20T22:59:49.838+0000	0.89800000190734
11	52	46	13	52	31	15	43	2019-10-20T22:59:49.838+0000	0.71099996566772
15	34	16	33	5	3	45	8	2019-10-20T22:59:50.623+0000	1.29299998283386
35	37	22	29	47	13	17	52	2019-10-20T22:59:50.670+0000	1.15200018882751
13	40	7	43	32	39	20	63	2019-10-20T22:59:50.670+0000	0.87100005149841
4	27	60	2	30	50	62	43	2019-10-20T22:59:50.670+0000	1.05800008773803
20	1	6	3	42	23	22	14	2019-10-20T22:59:51.421+0000	1.34100008010864
3	4	19	50	33	45	31	8	2019-10-20T22:59:51.437+0000	1.16900014877319
22	39	3	48	18	11	57	52	2019-10-20T22:59:51.437+0000	0.98099994659423

Εικόνα 43: Πάροχος 2 - Δεδομένα εξόδου (2/2)

Επεξήγηση δεδομένων - Ενδεικτική

Για το δεύτερο πάροχο, παίρνουμε ως αναφορά την πρώτη γραμμή, και παρατηρούμε ότι το πεδίο “Time” δεν έχει αποσταλεί και ορθώς καθώς η αντίστοιχη συγκατάθεση “Time_consent” το υποδηλώνει με την τιμή “33” που στο δυαδικό είναι “0010000**0**1”, με το δεύτερο bit να είναι 0. Σε αντίθεση, το πεδίο “Speed” αποστέλλεται αφού το δεύτερο bit της αντίστοιχης συγκατάθεσης “Speed_consent” είναι 1, καθώς το “11” αντιστοιχεί στο δυαδικό “0000101**1**”.

3ος Πάροχος

VehicleID	Time	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density	enqueuedTime	Time_consent
42	60	-1	-1	597962	-1	-1	-1	1	2019-10-20T22:58:43.903+0000	5
3	-1	-1	-1	601311	-1	-1	89	2	2019-10-20T22:58:44.121+0000	10
81	60	-1	-1	601321	-1	-1	69	2	2019-10-20T22:58:44.309+0000	14
14	-1	-1	-1	-1	5676289	-1	112	-1	2019-10-20T22:58:44.637+0000	50
10	-1	111	0	-1	5676811	-1	-1	-1	2019-10-20T22:58:44.731+0000	19
52	-1	91	-1	599563	5679106	-1	108	4	2019-10-20T22:58:44.918+0000	40
5	-1	74	-1	599615	-1	1176	-1	-1	2019-10-20T22:58:45.231+0000	3
97	60	37	-1	-1	5672015	-1	37	-1	2019-10-20T22:58:45.528+0000	23
85	60	-1	111	599640	5679354	-1	108	-1	2019-10-20T22:58:45.121+0000	36
89	-1	-1	0	586634	-1	-1	-1	2	2019-10-20T22:58:45.715+0000	9
24	60	56	-1	586997	-1	4812	56	-1	2019-10-20T22:58:45.934+0000	52
130	60	-1	-1	598247	5683314	12935	-1	1	2019-10-20T22:58:46.137+0000	29
138	-1	-1	0	-1	5684550	34218	60	1	2019-10-20T22:58:46.324+0000	25

Εικόνα 44: Πάροχος 3 - Δεδομένα εξόδου (1/2)

Time_consent	Speed_consent	Dist_front_consent	xCoord_consent	yCoord_consent	SectionID_consent	Avg_Sect_Speed_consent	Sect_Density_consent	OutputTime	Latency
5	59	25	20	16	51	9	14	2019-10-20T22:58:44.664+0000	0.760999917984001
10	48	17	60	40	40	46	14	2019-10-20T22:58:45.445+0000	1.32399988174438
14	56	35	15	27	24	28	62	2019-10-20T22:58:45.460+0000	1.151000022888181
50	25	56	50	61	27	47	0	2019-10-20T22:58:45.460+0000	0.822999954223633
19	37	54	34	47	11	9	2	2019-10-20T22:58:45.460+0000	0.729000091552734
40	20	58	38	39	59	14	13	2019-10-20T22:58:46.242+0000	1.324000120162963
3	14	26	15	57	29	26	11	2019-10-20T22:58:46.257+0000	1.026000022888183
23	60	17	11	36	8	54	10	2019-10-20T22:58:46.257+0000	0.728999853134155
36	0	47	28	13	58	61	2	2019-10-20T22:58:46.257+0000	1.135999917984009
9	33	46	54	17	51	51	53	2019-10-20T22:58:47.243+0000	1.528000116348267
52	46	35	62	25	61	53	3	2019-10-20T22:58:47.260+0000	1.325999975204467
29	43	3	63	38	28	42	38	2019-10-20T22:58:47.260+0000	1.122999906539917
25	50	60	10	55	52	14	21	2019-10-20T22:58:47.260+0000	0.936000108718872

Εικόνα 45: Πάροχος 3 - Δεδομένα εξόδου (2/2)

Επεξήγηση δεδομένων - Ενδεικτική

Για τον τρίτο πάροχο παίρνουμε ως σημείο αναφοράς την πρώτη γραμμή του πίνακα. Παρατηρούμε ότι για το πεδίο “Speed” η τιμή είναι “-1” υποδεικνύοντας ότι η συγκατάθεση του χρήστη δεν επιτρέπει τη χρήση αυτού του πεδίου για τον τρίτο πάροχο. Αν ανατρέξουμε στο πεδίο με την αντίστοιχη συγκατάθεση, δηλαδή το “Speed_consent” βλέπουμε ότι έχει την τιμή “59” που σε δυαδική αναπαράσταση είναι 00111**0**11” και επομένως ορθώς η τιμή δεν αποστέλλεται καθώς το τρίτο bit είναι 0. Εν αντιθέσει το πεδίο “xCoord” της ίδιας γραμμής έχει την τιμή “597962” και επομένως θα πρέπει το τρίτο bit της αντίστοιχης συγκατάθεσης να είναι “1”. Κάτι που επαληθεύεται κοιτώντας το πεδίο “xCoord_consent”, του οποίου η τιμή “20” είναι στο δυαδικό “00010**1**00”.

11.7 Μέτρηση μέσης καθυστέρησης συστήματος

Η μέση καθυστέρηση του συστήματος ορίζεται ως η χρονική διάρκεια από τη στιγμή που ένα μήνυμα εισέρχεται στο Event Hub εισόδου έως τη στιγμή που εξέρχεται δηλαδή τη στιγμή της εγγραφής στο Event Hub εξόδου. Για τη μέτρηση αυτή διαβάζουμε τα δεδομένα από τα Event Hubs εξόδου και ακολουθούμε τη διαδικασία της παραγράφου 10.8.

Η καθυστέρηση υπολογίζεται ξεχωριστά για τους τρεις παρόχους του παραδείγματος και στο εν λόγω παράδειγμα είναι της τάξης των 900 ms.

```
1 outDf.select(avg('Latency')).show()
```

▸ (1) Spark Jobs

```
+------------------+
|      avg(Latency)|
+------------------+
|0.8762666702270507|
+------------------+
```

```
1 outDf2.select(avg('Latency')).show()
```

▸ (1) Spark Jobs

```
+------------------+
|      avg(Latency)|
+------------------+
|0.8731000065803528|
+------------------+
```

```
1 outDf3.select(avg('Latency')).show()
```

▸ (1) Spark Jobs

```
+------------------+
|      avg(Latency)|
+------------------+
|0.864592001914978|
+------------------+
```

Εικόνα 46: Καθυστέρηση επεξεργασίας ανά πάροχο

11.8 Δεδομένα Blob Storage

Τα δεδομένα αποθηκεύονται σε μακροπρόθεσμη βάση στο Blob Storage για διατήρηση και για μελλοντική ανάλυση. Αυτό επιτυγχάνεται με τη βοήθεια της λειτουργίας Capture του Event Hubs.

Τα δεδομένα θα αποθηκευτούν κάτω από τον container της επιλογής μας με την εξής δομή:

«{Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}».

Η λειτουργία Capture χρησιμοποιεί τη μορφή αποθήκευσης “Avro”, η οποία είναι βασισμένη στην αποθήκευση γραμμής και χρησιμοποιείται συχνά σαν πλατφόρμα σειριοποίησης.

Συγκεκριμένα το σχήμα των δεδομένων αποθηκεύεται σε JSON μορφή διευκολύνοντας την ανάγνωση και την ερμηνεία του από άλλα προγράμματα, ενώ τα δεδομένα αποθηκεύονται σε δυαδική μορφή καθιστώντας τη διαχείρισή τους αποδοτικότερη.

Στη συνέχεια, διαβάζουμε τα δεδομένα που έχουν αποθηκευτεί στο Blob Storage για τον πρώτο πάροχο και επιβεβαιώνουμε τη δυαδική αναπαράσταση των δεδομένων από το πεδίο “body:binary”. Επίσης, παρατηρούμε ότι το πεδίο με τη χρονοσφραγίδα εγγραφής στο Event hub είναι τύπου “string” και όχι τύπου “timestamp”.

```
# Read Captured Data
input_path = "wasbs://captureddata@trialdatawithconsents.blob.core.windows.net/trialeventhubs/trialoutput1/0/2019/10/*/*/*/*.avro"
captured = spark.read.format("avro").load(input_path)
```

```
captured:  pyspark.sql.dataframe.DataFrame
    SequenceNumber: long
    Offset: string
    EnqueuedTimeUtc: string
  ▸ SystemProperties: map
  ▸ Properties: map
    Body: binary
```

Εικόνα 47 : Σχήμα Δεδομένων διατήρησης (retention data) (Capture-Event Hubs)

Ακολούθως μετατρέπουμε τα δεδομένα από δυαδική σε αλφαριθμητική αναπαράσταση και εμφανίζοντας τα, επιβεβαιώνουμε τη JSON μορφή τους.

```
data = captured.select(captured["body"].cast("string"))
display(data)
```

▸ (1) Spark Jobs
▸ data: pyspark.sql.dataframe.DataFrame = [body: string]

body
{"VehicleID":1,"Time":0,"Speed":-1,"Dist_front":0,"xCoord":-1,"yCoord":-1,"SectionID":95246,"Avg_Sect_Speed":-1,"Sect_Density":1,"enqueuedTime":"2019-10-26T09:03:24.784Z","Time_consent":45,"Speed_consent":28,"Dist_front_consent":29,"xCoord_consent":50,"yCoord_consent":28,"SectionID_consent":15,"Avg_Sect_Speed_consent":28,"Sect_Density_consent":51}
{"VehicleID":3,"Time":-1,"Speed":-1,"Dist_front":0,"xCoord":-1,"yCoord":-1,"SectionID":-1,"Avg_Sect_Speed":-1,"Sect_Density":-1,"enqueuedTime":"2019-10-26T09:03:24.375Z","Time_consent":10,"Speed_consent":48,"Dist_front_consent":17,"xCoord_consent":60,"yCoord_consent":40,"SectionID_consent":40,"Avg_Sect_Speed_consent":46,"Sect_Density_consent":14}
{"VehicleID":57,"Time":-1,"Speed":-1,"Dist_front":-1,"xCoord":595825,"yCoord":-1,"SectionID":510,"Avg_Sect_Speed":-1,"Sect_Density":2,"enqueuedTime":"2019-10-26T09:03:25.112Z","Time_consent":16,"Speed_consent":52,"Dist_front_consent":6,"xCoord_consent":5,"yCoord_consent":12,"SectionID_consent":13,"Avg_Sect_Speed_consent":14,"Sect_Density_consent":43}

Εικόνα 48: Αλφαριθμητική αναπαράσταση δεδομένων διατήρησης (retention data)

Στη συνέχεια τα δομούμε βάσει του σχήματός τους, ώστε να έρθουν σε μορφή πίνακα όπως φαίνεται στις εικόνες που ακολουθούν. Ο μετασχηματισμός αυτός είναι χρήσιμος για τις υπηρεσίες που επιθυμούν να χρησιμοποιήσουν τα δεδομένα αυτά για ανάλυση.

Εξαιτίας της ανάγκης παράθεσης του σε πολλαπλές εικόνες, επαναλαμβάνεται μεταξύ των εικόνων η στήλη “Time_consent” για βοήθεια στον αναγνώστη.

▸ (2) Spark Jobs

▸ data: pyspark.sql.dataframe.DataFrame = [VehicleID: integer, Time: integer ... 17 more fields]

VehicleID	Time	Speed	Dist_front	xCoord	yCoord	SectionID	Avg_Sect_Speed	Sect_Density	enqueuedTime	Time_consent
1	0	-1	0	-1	-1	95246	-1	1	2019-10-26T09:03:24.784+0000	45
3	-1	-1	0	-1	-1	-1	-1	-1	2019-10-26T09:03:24.375+0000	10
57	-1	-1	-1	595825	-1	510	-1	2	2019-10-26T09:03:25.112+0000	16
44	60	95	-1	-1	-1	510	-1	2	2019-10-26T09:03:25.015+0000	9
106	-1	-1	-1	-1	-1	-1	-1	-1	2019-10-26T09:03:25.331+0000	40
141	-1	-1	0	594481	-1	852	-1	1	2019-10-26T09:03:25.534+0000	60
42	60	56	0	-1	-1	980	56	-1	2019-10-26T09:03:25.768+0000	5
3	-1	-1	0	-1	-1	-1	-1	-1	2019-10-26T09:03:25.987+0000	10

Εικόνα 49 : Δεδομένα Διατήρησης (Retention data) (1/2)

Time_consent	Speed_consent	Dist_front_consent	xCoord_consent	yCoord_consent	SectionID_consent	Avg_Sect_Speed_consent	Sect_Density_consent	outputTimeUtc
45	28	29	50	28	15	28	51	10/26/2019 9:03:26 AM
10	48	17	60	40	40	46	14	10/26/2019 9:03:26 AM
16	52	6	5	12	13	14	43	10/26/2019 9:03:27 AM
9	31	40	30	42	11	16	31	10/26/2019 9:03:27 AM
40	62	20	24	36	6	38	44	10/26/2019 9:03:27 AM
60	14	3	23	12	5	24	1	10/26/2019 9:03:27 AM
5	59	25	20	16	51	9	14	10/26/2019 9:03:27 AM
10	48	17	60	40	40	46	14	10/26/2019 9:03:27 AM

Εικόνα 50 : Δεδομένα Διατήρησης (Retention data) (2/2)

Παρατηρώντας τα δεδομένα διατήρησης αντιλαμβανόμαστε ότι η χρονοσφραγίδα εξόδου “outputTimeUtc” περιορίζεται σε ακρίβεια δευτερολέπτων. Για το λόγο αυτό ο υπολογισμός της καθυστέρησης δεν γίνεται πάνω στο συγκεκριμένο σύνολο δεδομένων.

12 Συμπεράσματα

12.1 Συνεισφορά της παρούσας εργασίας

Στην παρούσα εργασία παρουσιάστηκε το πεδίο μελέτης των διασυνδεδεμένων οχημάτων συνδυαστικά με τις δυνατότητες αξιοποίησης των παραγόμενων δεδομένων.

Γίνεται κατανοητή η ανάγκη για ισότιμη πρόσβαση και αξιοποίηση των δεδομένων, η οποία διασφαλίζει με τη σειρά της το δίκαιο ανταγωνισμό μεταξύ των επιχειρήσεων και τη μεγιστοποίηση της παρεχόμενης αξίας στον τελικό καταναλωτή.

Όπως προκύπτει από τις ήδη υλοποιημένες μελέτες, απαιτείται μια κεντρική πρωτοβουλία σε επίπεδο ΕΕ για τον καθορισμό μιας κοινά αποδεκτής τεχνικής λύσης η οποία θα παρεμβάλλεται μεταξύ των παραγόμενων από τα οχήματα δεδομένων και των παρόχων υπηρεσιών και θα αναλαμβάνει τη διαχείρισή τους.

Η λύση αυτή θα πρέπει να διαχειρίζεται τα δεδομένα των χρηστών σύμφωνα με τη συγκατάθεσή τους και να εξασφαλίζει τη μετάδοση στους εκάστοτε εξουσιοδοτημένους παρόχους, με τρόπο ισότιμο, σε επίπεδο χρόνου, πληρότητας και ποιότητας δεδομένων.

Στο σχεδιασμό αυτής της λύσης, κρίσιμο ζήτημα αποτελεί η διασφάλιση της ιδιωτικότητας των δεδομένων των χρηστών και η αποφυγή μετάδοσης προσωπικών δεδομένων. Πρακτικά, η διαχείριση της πρόσβασης θέτει μια νέα τεχνική απαίτηση προς επίλυση, για να επιτραπεί η χρήση και αξιοποίηση των δεδομένων που προέρχονται από τα οχήματα.

Η παρούσα διπλωματική συνεισφέρει σημαντικά προς την κάλυψη αυτής της τεχνικής απαίτησης, αφού βασικό αντικείμενό της είναι ο σχεδιασμός και η ανάπτυξη μιας προτεινόμενης αρχιτεκτονικής λύσης, η οποία επιτρέπει τη διαχείριση της συγκατάθεσης του χρήστη σε ρεύματα δεδομένων σε πραγματικό χρόνο και παρέχει μια δυνητική υλοποίηση της.

Συγκεκριμένα, στη παρούσα εργασία, μοντελοποιήσαμε το πρόβλημα της διαχείρισης της πρόσβασης πάνω στο ρεύμα των δεδομένων των οχημάτων βάσει της συγκατάθεσης των χρηστών και σχεδιάσαμε τον τρόπο με τον οποίο οι πάροχοι υπηρεσιών θα αποκτούν πρόσβαση στο ρεύμα δεδομένων των οχημάτων που εισέρχεται στον Κοινόχρηστο Επεξεργαστή. Τα παραπάνω γίνονται με τρόπο που διασφαλίζεται η συγκατάθεση των χρηστών και η ιδιωτικότητα των δεδομένων.

Η υλοποίηση της προτεινόμενης αρχιτεκτονικής στηρίζεται σε υφιστάμενα, διαδεδομένα και αξιόπιστα εργαλεία ανάπτυξης εφαρμογών. Συγκεκριμένα, σχεδιάσαμε την αρχιτεκτονική λύση του συστήματος σε επίπεδο υπηρεσιών αλλά και σε τεχνολογικό επίπεδο κάνοντας χρήση των υπηρεσιών της πλατφόρμας MS Azure και επιπλέον υπηρεσιών λογισμικού ανοικτού κώδικα.

Τέλος, κατά την παρούσα διπλωματική πραγματοποιήθηκε και σχετική υλοποίηση, στο πλαίσιο της οποίας αναπτύχθηκε υπηρεσία λογισμικού η οποία δέχεται ως είσοδο το ρεύμα δεδομένων των οχημάτων και το καθιστά διαθέσιμο για κατανάλωση στους παρόχους υπηρεσιών, βάσει της συγκατάθεσης των χρηστών. Για τις ανάγκες της προγραμματιστικής υλοποίησης αξιοποιήθηκαν και μελετήθηκαν τεχνολογίες επεξεργασίας ρευμάτων δεδομένων

και δεδομένων μεγάλης κλίμακας, όπως το Apache Spark, το Event Hubs και άλλες υπηρεσίες της πλατφόρμας MS Azure. Παράλληλα, στις υπηρεσίες που χρησιμοποιήθηκαν εντοπίστηκαν οι παράμετροι και ο τρόπος με τον οποίο πρέπει αυτές να μεταβάλλονται ώστε να συναντούν τις διαφορετικές απαιτήσεις φορτίου.

Ως συμπέρασμα, προκύπτει ότι με τη χρήση των τεχνολογιών που μελετήθηκαν είναι εφικτή η διαχείριση της συγκατάθεσης πρόσβασης σε ρεύματα δεδομένων σε πραγματικό χρόνο. Συνεπώς, τα ρεύματα δεδομένων δύναται να διατεθούν στους παρόχους υπηρεσιών εξασφαλίζοντας τις προτιμήσεις των χρηστών και την ιδιωτικότητα των δεδομένων τους και κατ' επέκταση δίνοντας τη δυνατότητα για ανάπτυξη και παροχή υπηρεσιών σχεδόν πραγματικού χρόνου στους χρήστες.

12.2 Προτάσεις για βελτίωση

Στην προτεινόμενη αρχιτεκτονική αλλά και στην υλοποίηση που πραγματοποιήθηκε στο πλαίσιο της διπλωματικής εργασίας, η διαχείριση και η κατάτμηση του φόρτου εργασίας που προκύπτει από την επεξεργασία των ρευμάτων δεδομένων των οχημάτων επιτυγχάνεται με τη χρήση τεχνολογιών και μεθοδολογιών προσανατολισμένων στα δεδομένα μεγάλης κλίμακας, όπως είναι το Apache Spark, το Event hubs αλλά και η εκτέλεση της επεξεργασίας σε συστοιχία υπολογιστικών μηχανών.

Για την επίτευξη περαιτέρω κατάτμησης του φόρτου εργασίας που καλείται να διαχειριστεί το σύστημα αλλά και βελτίωσης των χρόνων μετάδοσης των δεδομένων από τα οχήματα στο περιβάλλον του κοινόχρηστου επεξεργαστή προτείνεται η γεωχωρική κλιμάκωση της επεξεργασίας. Στην περίπτωση αυτή η δρομολόγηση των δεδομένων των οχημάτων θα γίνεται βάσει της τοποθεσίας του οχήματος στον κατάλληλο εξυπηρετητή όπου και θα γίνεται η επεξεργασία τους βάσει της συγκατάθεσης των χρηστών. Για να πραγματοποιηθεί κάτι τέτοιο θα πρέπει ο ενιαίος χώρος εμβέλειας του συστήματος να χωριστεί κατάλληλα σε γεωγραφικά τμήματα. Για την τμηματοποίηση αυτή απαιτείται να ληφθούν υπόψιν συνδυαστικά οι εξής παράγοντες:

- Οι τοποθεσίες των κέντρων δεδομένων της πλατφόρμας Azure, καθώς αποτελούν τις πιθανές επιλογές για τη φυσική παρουσία του απαραίτητου υλικού εξοπλισμού του συστήματος
- Η πυκνότητα των διασυνδεδεμένων οχημάτων στις διάφορες γεωγραφικές περιοχές του συνολικού χώρου εμβέλειας του συστήματος, βάσει της οποίας δύναται να οριοθετηθούν γεωγραφικές περιοχές με πλήθος οχημάτων, και κατ' επέκταση αποστολέων δεδομένων, ίδιας τάξης μεγέθους. Κατά αυτόν τον τρόπο ο συνολικός όγκος των δεδομένων των οχημάτων που αποστέλλονται τμηματοποιείται σχεδόν ισόποσα και μπορεί εν συνεχεία να επεξεργαστεί σε διαφορετικές μηχανές Spark εισάγοντας έτσι ένα νέο επίπεδο κατάτμησης του συνολικού φόρτου εργασίας και συνεισφέροντας στη μείωση της καθυστέρησης της επεξεργασίας.

Η συγκεκριμένη επέκταση του συστήματος, εκτός της μείωσης του χρόνου που απαιτείται για την επεξεργασία των ρευμάτων δεδομένων, δύναται να συνεισφέρει επίσης στη μείωση της διάρκειας κατά την οποία η πληροφορία ταξιδεύει από τα οχήματα στο περιβάλλον του

κοινόχρηστου επεξεργαστή μέσω του δικτύου τηλεπικοινωνιών, καθώς η πληροφορία θα δρομολογείται πλέον στον κοντινότερο, υπό την έννοια του χρόνου μετάδοσης, εξυπηρετητή.

13 Παράρτημα 1: Τεχνοδιαμόρφωση πλατφόρμας

13.1 Τεχνοδιαμόρφωση πλατφόρμας Azure και περιβάλλοντος υλοποίησης

Για τις ανάγκες της υλοποίησης, έγινε χρήση της πλατφόρμας Azure συμπεριλαμβανομένης και της πλατφόρμας Azure Databricks. Αρχικά δημιουργήθηκε ένα Resource Group έτσι ώστε να τοποθετηθούν όλοι οι πόροι που είναι σχετικοί με την επιθυμητή εφαρμογή. Συνεπώς, στο Resource Group αυτό ανήκουν το σύνολο των Event Hubs, το Blob Storage και η πλατφόρμα Azure Databricks.

13.1.1 Storage Account

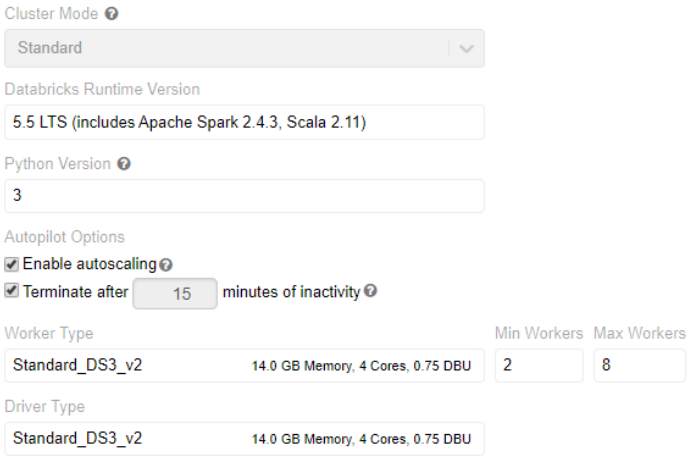
Στη συνέχεια, δημιουργήθηκε ένα Storage Account ώστε να αποθηκευτούν τα δεδομένα που απαιτούνται για την εφαρμογή, όπως τα δεδομένα συγκατάθεσης των χρηστών αλλά και τα δεδομένα των οχημάτων τα οποία χρησιμοποιήθηκαν υποστηρικτικά για τη δημιουργία των δεδομένων συγκατάθεσης.. Το συγκεκριμένο Account ήταν τύπου “StorageV2”, και αποτελεί λύση για τα περισσότερα σενάρια χρήσης, υποστηρίζοντας τις υπηρεσίες Blobs, Files, Disks, Queues, Tables. Ορίστηκε η επιλογή πρόσβασης στο επίπεδο “hot” καθώς ενδείκνυται για συχνή πρόσβαση στα δεδομένα. Τέλος το Storage Account ήταν τύπου Read-access geo-redundant storage (RA-GRS) που σημαίνει ότι δημιουργήθηκε στην περιοχή της επιλογής μας που ήταν το κέντρο δεδομένων της Βόρειας Ευρώπης, αλλά δημιουργήθηκε επίσης ένα αντίγραφο του στην περιοχή της Δυτικής Ευρώπης με επιπλέον δυνατότητα ανάγνωσης.

Εικόνα 51: Storage Account (primary & replication)

Στο storage account δημιουργήθηκαν τρεις container. Ο πρώτος χρησιμοποιήθηκε για να αποθηκευτούν τα δεδομένα των οχημάτων ώστε να επεξεργαστούν και να δημιουργηθούν τα κατάλληλα δεδομένα συγκατάθεσης των χρηστών τα οποία και αποθηκεύτηκαν σε έναν δεύτερο container. Ο τρίτος container χρησιμοποιήθηκε για την αποθήκευση των δεδομένων κατόπιν της επεξεργασίας.

13.1.2 Azure Databricks

Για την επεξεργασία των δεδομένων γίνεται χρήση του Apache Spark μέσω της πλατφόρμας Databricks. Η χρησιμοποιούμενη έκδοση του Spark είναι η Apache Spark 2.4.3. Η επεξεργασία γίνεται μέσω Notebooks κάνοντας χρήση της έκδοσης Python 3. Επίσης απαιτείται η χρήση της βιβλιοθήκης "azure-eventhubs-spark" και για το λόγο αυτό εισάγεται η βιβλιοθήκη αυτή μέσω της συνιστώσας Maven "com.microsoft.azure:azure-eventhubs-spark_2.11:2.3.13" στο σύνολο των βιβλιοθηκών του cluster. Στη συνέχεια φαίνεται η παραμετροποίηση του Spark Cluster.



Εικόνα 52: Spark Cluster

13.2 Event hubs

Για τη δημιουργία των Event Hubs απαιτείται αρχικά η δημιουργία ενός ονοματοχώρου (namespace) Event hubs, ο οποίος εκχωρείται στο Resource Group της εφαρμογής. Επίσης ορίζονται τα TUs να ξεκινούν από την τιμή 2 και να μπορούν να φτάσουν κατά απαίτηση φορτίου στη μέγιστη τιμή των 20 μέσω της επιλογής Auto-inflate.

Κατόπιν δημιουργούνται τα απαραίτητα Event Hubs, ένα για την είσοδο των δεδομένων και ισάριθμο πλήθος με τους παρόχους υπηρεσιών για την έξοδο. Ο αριθμός των διαμερίσεων ορίζεται κάθε φορά ανάλογα με τον μέσο εκτιμώμενο ρυθμό εισόδου των δεδομένων.

Για την αποθήκευση των δεδομένων μακροπρόθεσμα ενεργοποιείται η δυνατότητα "Capture" των Event Hubs στην έξοδο, με τις επιλογές "Time window" και "Size window" να καθορίζουν τη στιγμή εκτέλεσης της λειτουργίας "capture". Συγκεκριμένα, όταν συμπληρώνεται ο ελάχιστος χρόνος που ορίζει η επιλογή "Time window" ή το ελάχιστο μέγεθος δεδομένων που έχουν εισαχθεί στο Event hub "Size window" εκτελείται η λειτουργία συλλογής και αποθήκευσης των γεγονότων "capture". Συνεπώς, εφόσον δεν απαιτείται η συχνή συλλογή των δεδομένων καθώς προορίζονται για μακροπρόθεσμη διατήρηση, χρησιμοποιούνται οι μέγιστες τιμές των επιλογών αυτών.

Για την πρόσβαση στα Event Hubs δημιουργήθηκε πολιτική κοινής πρόσβασης, μία για κάθε Hub, με δικαιώματα αποστολής για την είσοδο και δικαιώματα σύλληψης για την έξοδο. Εναλλακτικά μπορεί να γίνει απονομή ρόλου "Azure Event Hubs Data Receiver" για τους παρόχους υπηρεσιών σε επίπεδο "Azure AD, user, group or service principal" εφόσον έχουν εγγράψει τις υπηρεσίες τους στην εφαρμογή.

14 Παράρτημα 2: Παράθεση κώδικα και αποτελεσμάτων

14.1 Κώδικας Παραγωγού

```
import java.io.*;
import java.util.*;

import com.microsoft.azure.eventhubs.ConnectionStringBuilder;
import com.microsoft.azure.eventhubs.EventData;
import com.microsoft.azure.eventhubs.EventHubClient;
import com.microsoft.azure.eventhubs.EventHubException;

import java.io.IOException;
import java.nio.charset.Charset;
import java.time.Instant;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;

import com.fasterxml.jackson.databind.ObjectMapper;
import org.json.*;

public class Producer {

   public static void main(String[] args) {
      final ConnectionStringBuilder connStr = new ConnectionStringBuilder()
            .setNamespaceName("trialeventhubs")
            .setEventHubName("trialinput")
            .setSasKeyName("inputPolicy")
            .setSasKey("***");

      // The Executor handles all asynchronous tasks and this is passed to the EventHubClient instance.
      // This enables the user to segregate their thread pool based on the work load.
      // This pool can then be shared across multiple EventHubClient instances.
      // The following sample uses a single thread executor, as there is only one EventHubClient instance,
      // handling different flavors of ingestion to Event Hubs here.
      final ScheduledExecutorService executorService = Executors.newScheduledThreadPool(4);

      // Each EventHubClient instance spins up a new TCP/SSL connection, which is expensive.
      // It is always a best practice to reuse these instances. The following sample shows this.
      EventHubClient ehClient = null;
      try {
         ehClient = EventHubClient.createSync(connStr.toString(), executorService);
      } catch (EventHubException | IOException e) {
         e.printStackTrace();
      }

      int count = 0;
      List<String> fieldNames;
      List<String> headers = null;
      //List<Map<String, String>> list = new ArrayList<>();

      try {
         try (Scanner scanner = new Scanner(new File("C:/Users/George/Desktop/Antwerp_subset.csv"))) {
            if (scanner.hasNextLine())
               headers = getRecordFromLine(scanner.nextLine());
            while (scanner.hasNextLine()) {
               fieldNames = getRecordFromLine(scanner.nextLine());

               Map<String, Integer> obj = new LinkedHashMap<>();

               for (int i = 0; i < fieldNames.size(); i++) {
                  obj.put(headers.get(i), Integer.parseInt(fieldNames.get(i)));
               }
```

```
                ObjectMapper mapper = new ObjectMapper();
                try {
                    StringWriter jsonObj = new StringWriter();
                    mapper.writeValue(jsonObj, obj);
                    System.out.println(jsonObj);
                    //check if is valid JSON
                    try {
                        new JSONObject(jsonObj);
                    } catch (JSONException ex) {
                        try {
                            new JSONArray(jsonObj);
                        } catch (JSONException ex1) {
                            System.out.println("No json");
                        }
                    }

                    byte[] payloadBytes = jsonObj.toString().getBytes(Charset.defaultCharset());
                    EventData sendEvent = EventData.create(payloadBytes);

                    // Send - not tied to any partition
                    // Event Hubs service will round-robin the events across all Event Hubs partitions.
                    // This is the recommended & most reliable way to send to Event Hubs.
                    try {
                        ehClient.sendSync(sendEvent);
                    } catch (EventHubException e) {
                        e.printStackTrace();
                    }
                    // End of code to event hub
                } catch (IOException e) {
                    System.out.println("catch mapper");
                    e.printStackTrace();
                }
                count++;
                if (count == 200)
                    break;
            }
            System.out.println(Instant.now() + ": Send Complete...");

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    } finally {
        try {
            ehClient.closeSync();
        } catch (EventHubException e) {
            e.printStackTrace();
        }
        executorService.shutdown();
    }


}
private static List<String> getRecordFromLine(String line) {
    List<String> values = new ArrayList<>();
    try (Scanner rowScanner = new Scanner(line)) {
        rowScanner.useDelimiter(",");
        while (rowScanner.hasNext()) {
            values.add(rowScanner.next());
        }
    }
    return values;
}

}
```

Επίσης παρατίθεται ο κώδικας για τα dependencies του συγκεκριμένου project.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <dependencies>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.9.8</version>
        </dependency>
        <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20180813</version>
        </dependency>
        <dependency>
            <groupId>com.microsoft.azure</groupId>
            <artifactId>azure-eventhubs</artifactId>
            <version>2.2.0</version>
        </dependency>
    </dependencies>

    <modelVersion>4.0.0</modelVersion>

    <groupId>org.apache.maven.simple-producer</groupId>
    <artifactId>producer</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.7.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

14.2 Κώδικας δημιουργίας συγκατάθεσης χρηστών

14.2.1 Σύνδεση με το Blob Storage

```
storage_account_name = "trialdatawithconsents"
storage_account_access_key = "**********"

spark.conf.set(
  "fs.azure.account.key."+storage_account_name+".blob.core.windows.net",
  storage_account_access_key)
#path of data
input_path = "wasbs://antwerpsubset@trialdatawithconsents.blob.core.windows.net/Antwerp_subset.csv"
#path where consents will be stored
output_location = "wasbs://consents@trialdatawithconsents.blob.core.windows.net/"
```

14.2.2 Εύρεση αριθμού διαφορετικών οχημάτων

```
from pyspark.sql.types import StructType, StructField, IntegerType, DecimalType, StringType,
TimestampType
from pyspark.sql.functions import *

#creating the schema of data
schema = StructType([
      StructField('Time',IntegerType()),
      StructField('VehicleID', IntegerType()),
      StructField('Speed', IntegerType()),
      StructField('Dist_front', IntegerType()),
      StructField('xCoord', IntegerType()),
      StructField('yCoord', IntegerType()),
      StructField('SectionID', IntegerType()),
      StructField('Avg_Sect_Speed', IntegerType()),
      StructField('Sect_Density', IntegerType())])

#read data
staticData = spark.read.format("csv").schema(schema).load(input_path)
#drop empty elements
staticData = staticData.dropna()
staticData.show()
#staticData.rdd.getNumPartitions()
# group by id gives us the number of distinct vehicles
distinctVehicles = staticData.groupBy("VehicleID").count()
display(distinctVehicles)
# count rows
numOfVehicles = distinctVehicles.count()
# show number
numOfVehicles
```

14.2.3 Δημιουργία δεδομένων συγκατάθεσης

Στο τμήμα του κώδικα αυτού δημιουργείται ένα DataFrame με τόσες στήλες όσες και οι κατηγορίες των δεδομένων και έπειτα προστίθεται παράλληλα η στήλη με τα μοναδικά αναγνωριστικά “VehicleID”. Για να πραγματοποιηθεί η συνένωση των αναγνωριστικών με τα τυχαία δεδομένα συγκατάθεσης δημιουργείται μία στήλη με αύξοντες αριθμούς στο κάθε σύνολο, η οποία θα αποτελέσει το σημείο της σύνδεσης και έπειτα θα απορριφθεί.

```
import random
random_df_list = [] # create iteratively a list
for i in range(numOfVehicles):
  row = [random.randint(0,63), random.randint(0,63), random.randint(0,63), random.randint(0,63), \
        random.randint(0,63), random.randint(0,63), random.randint(0,63), random.randint(0,63)]
  random_df_list.append(row)
#creating the schema of data
cschema = StructType([
      StructField('Time_consent',IntegerType()),
      StructField('Speed_consent', IntegerType()),
      StructField('Dist_front_consent', IntegerType()),
      StructField('xCoord_consent', IntegerType()),
      StructField('yCoord_consent', IntegerType()),
      StructField('SectionID_consent', IntegerType()),
      StructField('Avg_Sect_Speed_consent', IntegerType()),
      StructField('Sect_Density_consent', IntegerType())])
# create dataframe from random list
```

```
consents = spark.createDataFrame(random_df_list,schema=cschema)
consents.show()
consents.count()

# add a connection point between the two dataframes to merge
i=0
staticDataVehId = distinctVehicles.select("VehicleID")
newDf1 = staticDataVehId.coalesce(1).withColumn("con_id",monotonically_increasing_id())
newDf1.show()
newDf1.count()
newDf2 = consents.coalesce(1).withColumn("con_id",monotonically_increasing_id())
newDf2.show()
newDf2.count()
finalDf = newDf2.join(newDf1, "con_id", "inner").drop("con_id")

finalDf.count()
```

14.2.4 Αποθήκευση δεδομένων σε αρχείο

```
# write file to blob storage
(finalDf
  .coalesce(1)
  .write.mode("overwrite")
  .option("header", "true")
  .format("com.databricks.spark.csv")
  .save(str(output_location)+"folder"))
files = dbutils.fs.ls(output_location+"folder")
output_file = [x for x in files if x.name.startswith("part-")]
display(output_file)
# Move the wrangled-data CSV file from a sub-folder (output_location) to the root of the blob container
# While simultaneously changing the file name
dbutils.fs.mv(output_file[0].path, "%s/FileWithConsents.csv" % output_location)
```

14.3 Επεξεργασία ρεύματος εισόδου και εγγραφή εξόδου

14.3.1 Ανάγνωση ρεύματος εισόδου

```
#Event hub for input
event_hub_connection_string1 =
"Endpoint=sb://trialeventhubs.servicebus.windows.net/;SharedAccessKeyName=inputPolicy1;SharedAccessKe
y=****;EntityPath=trialinput"
import json
from pyspark.sql.types import StructType, StructField, IntegerType, DecimalType, StringType,
TimestampType
from pyspark.sql.functions import *
from datetime import datetime as dt

# Event hubs Source with default settings
connectionString = event_hub_connection_string1

ehConf = {
  'eventhubs.connectionString' : connectionString,
  'eventhubs.startingPosition' : json.dumps({"offset":"@latest", "seqNo":-1,"enqueuedTime":
None,"isInclusive": True})
}

# Read streaming data
streaming_df = spark \
  .readStream \
```

```
  .format("eventhubs") \
  .options(**ehConf) \
  .load()

# data are on "body" (eventhubs structure)
# we should define the JSON schema of the "body" content and cast body as string because because of its binary format
schema = StructType([
        StructField('Time',IntegerType()),
        StructField('VehicleID', IntegerType()),
        StructField('Speed', IntegerType()),
        StructField('Dist_front', IntegerType()),
        StructField('xCoord', IntegerType()),
        StructField('yCoord', IntegerType()),
        StructField('SectionID',IntegerType()),
        StructField('Avg_Sect_Speed', IntegerType()),
        StructField('Sect_Density', IntegerType())])
#select body(json data) and enqueuedTime to compute metrics
df = streaming_df.select(streaming_df["body"].cast("string"),streaming_df["enqueuedTime"])
#check schema
df.schema
#parse json object according to schema
df = df.select(from_json(col("body"),schema).alias("tmp"),col("enqueuedTime"))
#finally select all columns produced from parsing
df = df.select("tmp.*",col("enqueuedTime"))
#display(df)
```

14.3.2 Επεξεργασία δεδομένων

```
storage_account_name = "trialdatawithconsents"
storage_account_access_key = "*******"

spark.conf.set(
  "fs.azure.account.key."+storage_account_name+".blob.core.windows.net",
  storage_account_access_key)
# Read Consents
input_path = "wasbs://consents@trialdatawithconsents.blob.core.windows.net/FileWithConsents.csv"
consents = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load(input_path)
consents.schema
consents.cache()
joinStream = df.join(consents,"VehicleID")
cols = df.columns #return columns names as a list
cols.remove("VehicleID")
cols.remove("enqueuedTime")
writeConnectionString1="Endpoint=sb://trialeventhubs.servicebus.windows.net/;SharedAccessKeyName=outputPolicy1;SharedAccessKey=**;EntityPath=trialoutput1"
writeConnectionString2="Endpoint=sb://trialeventhubs.servicebus.windows.net/;SharedAccessKeyName=outputPolicy2;SharedAccessKey=**;EntityPath=trialoutput2"
writeConnectionString3="Endpoint=sb://trialeventhubs.servicebus.windows.net/;SharedAccessKeyName=out3;SharedAccessKey=**=;EntityPath=trialoutput3"

ehWriteConf1 = {
   'eventhubs.connectionString' : writeConnectionString1
}
ehWriteConf2 = {
  'eventhubs.connectionString' : writeConnectionString2
}
ehWriteConf3 = {
  'eventhubs.connectionString' : writeConnectionString3
}
```

```
ehConfArr = [ehWriteConf1,ehWriteConf2,ehWriteConf3]
index = -1
for sp in sp_list:
  index = index+1
  pos = 1
  pos = pos << (sp_list.index(sp))
  sp_df = joinStream
  for col_name in cols:
    sp_df = sp_df.withColumn(col_name, when( (col("%s_consent"%col_name).bitwiseAND(pos) == pos),
col(col_name) ).otherwise(-1))
  # Write body data from a DataFrame to EventHubs. Events are distributed across partitions using round-robin
model.
  new = sp_df.select(to_json(struct(sp_df.columns)).alias("body"))
  #Set up the Event Hub config dictionary with default settings
  ehWriteConfN = ehConfArr[index]
  ds = new \
    .select("body") \
    .writeStream \
    .format("eventhubs") \
    .options(**ehWriteConfN) \
    .option("checkpointLocation", "///output"+str(index)+".txt") \
    .start()
```

14.3.3 Υπολογισμός καθυστέρησης

```
from datetime import datetime as dt
#change writeConnectionString to select Service Provider Hub
ehConfSP = {
  'eventhubs.connectionString' : writeConnectionString1,
  'eventhubs.startingPosition' : json.dumps({"offset":None, "seqNo":0 ,"enqueuedTime":None,"isInclusive":
True})
}
#use endTime to choose data till a specific moment
time = dt.strptime("2019-10-18T17:43:10.133914+0000Z","%Y-%m-%dT%H:%M:%S.%f+0000Z")
endTime =time.strftime("%Y-%m-%dT%H:%M:%S.%fZ")
# # Create the positions
# startingEventPosition = {
#   "offset": None,
#   "seqNo": -1,
#   "enqueuedTime": endTime,
#   "isInclusive": True
# }
# ehConfSP["eventhubs.startingPosition"] = json.dumps(startingEventPosition)

data = spark \
  .read \
  .format("eventhubs") \
  .options(**ehConfSP) \
  .load()

# data are on "body" (eventhubs structure)
# we should define the JSON schema of the "body" content and cast body as string because it is in binary format
schema = StructType([
        StructField('VehicleID', IntegerType()),
        StructField('Time',IntegerType()),
        StructField('Speed', IntegerType()),
        StructField('Dist_front', IntegerType()),
        StructField('xCoord', IntegerType()),
        StructField('yCoord', IntegerType()),
        StructField('SectionID',IntegerType()),
```

```
        StructField('Avg_Sect_Speed', IntegerType()),
        StructField('Sect_Density', IntegerType()),
        StructField('enqueuedTime', TimestampType()),
        StructField('Time_consent',IntegerType()),
        StructField('Speed_consent', IntegerType()),
        StructField('Dist_front_consent', IntegerType()),
        StructField('xCoord_consent', IntegerType()),
        StructField('yCoord_consent', IntegerType()),
        StructField('SectionID_consent',IntegerType()),
        StructField('Avg_Sect_Speed_consent', IntegerType()),
        StructField('Sect_Density_consent', IntegerType())])
outDf = data.select(data["body"].cast("string"),data["enqueuedTime"])
outDf.schema
outDf = outDf.select(from_json(col("body"),schema).alias("tmp"),col("enqueuedTime"))
outDf = outDf.select("tmp.*",col("enqueuedTime").alias("OutputTime"))
#column latency is the result of substraction between input and output timestamp
outDf = outDf.withColumn("Latency",col("OutputTime").cast("double")-col("enqueuedTime").cast("double"))
outDf.select(avg('Latency')).show()
```

14.4 Μετασχηματισμός Δεδομένων Διατήρησης

```
import json
from pyspark.sql.types import StructType, StructField, IntegerType, DecimalType, StringType,
TimestampType
from pyspark.sql.functions import *
storage_account_name = "trialdatawithconsents"
storage_account_access_key = "***"
spark.conf.set(
  "fs.azure.account.key."+storage_account_name+".blob.core.windows.net",
  storage_account_access_key)
# Read Captured Data
input_path =
"wasbs://captureddata@trialdatawithconsents.blob.core.windows.net/trialeventhubs/trialoutput1/0/2019/10/*/*/*
/*.avro"
captured = spark.read.format("avro").load(input_path)
schema = StructType([
        StructField('VehicleID', IntegerType()),
        StructField('Time',IntegerType()),
        StructField('Speed', IntegerType()),
        StructField('Dist_front', IntegerType()),
        StructField('xCoord', IntegerType()),
        StructField('yCoord', IntegerType()),
        StructField('SectionID',IntegerType()),
        StructField('Avg_Sect_Speed', IntegerType()),
        StructField('Sect_Density', IntegerType()),#)])
        StructField('enqueuedTime', TimestampType()),
        StructField('Time_consent',IntegerType()),
        StructField('Speed_consent', IntegerType()),
        StructField('Dist_front_consent', IntegerType()),
        StructField('xCoord_consent', IntegerType()),
        StructField('yCoord_consent', IntegerType()),
        StructField('SectionID_consent',IntegerType()),
        StructField('Avg_Sect_Speed_consent', IntegerType()),
        StructField('Sect_Density_consent', IntegerType())])
data = captured.select(captured["body"].cast("string"))
display(data)
data = captured.select(captured["body"].cast("string"),captured["enqueuedTimeUtc"])
```

```
data =
data.select(from_json(col("body"),schema).alias("tmp"),col("enqueuedTimeUtc").alias("outputTimeUtc"))
#finally select all columns produced from parsing
data = data.select("tmp.*",col("outputTimeUtc"))
```

15 Βιβλιογραφία

[1] https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf
[2] https://www.idc.com/getdoc.jsp?containerId=prUS44998419
[3] C. H. Lee and H.-J. Yoon, "Medical big data: promise and challenges," Kidney Res.Clin. Pract., vol. 36, no. 1, pp. 3–11, 2017
[4] https://en.wikipedia.org/wiki/Big_data
[5] https://en.wikipedia.org/wiki/Computer_cluster
[6] https://site.ieee.org/connected-vehicles/ieee-connected-vechicles/connected-vehicles/
[7]https://europa.eu/youreurope/citizens/travel/security-and-emergencies/emergency-assistance-vehicles-ecall/index_el.htm
[8] https://en.wikipedia.org/wiki/Self-driving_car
[9] https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic
[10] https://www.wsdot.wa.gov/travel/automated-connected/home
[11]https://eur-lex.europa.eu/legal-content/EL/TXT/PDF/?uri=CELEX:52016DC0766&from=EL
[12] http://ec.europa.eu/growth/tools-databases/newsroom/cf/itemdetail.cfm?item_id=8640
[13] https://ec.europa.eu/transport/themes/its/c-its_en
[14] www.acea.be/uploads/publications/ACEA_Strategy_Paper_on_Connectivity.pdf
[15]www.acea.be/uploads/publications/ACEA_Position_Paper_Access_to_vehicle_data_for_third-party_services.pdf
[16]http://ec.europa.eu/competition/scp19/contributions/afcar_alliance_alliance_for_the_freedom_of_car_repair_in_the_eu.pdf
[17] https://ec.europa.eu/transport/sites/transport/files/2017-05-access-to-in-vehicle-data-andresources.pdf
[18] http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=COM:2018:0283:FIN
[19]Sandhu, Ravi & Ferraiolo, David. (2000). The NIST model for role-based access control.
[20] Hu, Vincent & Ferraiolo, David & Kuhn, D. & Schnitzer, A. & Sandlin, K. & Miller, R. & Scarfone, Karen. (2014). Guide to attribute based access control (ABAC) definition and considerations. National Institute of Standards and Technology Special Publication. 800-162.
[21]https://en.wikipedia.org/wiki/Microsoft_Azure
[22]https://azure.microsoft.com/en-us/
[23]Peter Mell and Timothy Grance (September 2011). The NIST Definition of Cloud Computing (Technical report). National Institute of Standards and Technology. Special publication 800-145.
[24] https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-about
[25] https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-scalability#throughput-units
[26]https://spark.apache.org/
[27] https://databricks.com/spark/about
[28] https://spark.apache.org/docs/latest/cluster-overview.html
[29]https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html
[30] https://spark.apache.org/docs/latest/rdd-programming-guide.html
[31]Zaharia Matei et. al. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation.
[32]https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html
[33] https://databricks.com/blog/2015/06/22/understanding-your-spark-application-through-visualization.html
[34] https://databricks.com/glossary/catalyst-optimizer
[35] https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html
[36] https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#continuous-processing
[37] https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#basic-concepts
[38] https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#output-modes
[39] https://en.wikipedia.org/wiki/Databricks
[40] https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-overview
[41] https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-overview-what-is
[42] https://docs.microsoft.com/en-us/azure/sql-data-warehouse/massively-parallel-processing-mpp-architecture
[43] https://azure.microsoft.com/en-us/services/data-factory/
[44] https://docs.microsoft.com/en-us/azure/data-lake-analytics/data-lake-analytics-overview
[45] https://azure.microsoft.com/en-us/services/machine-learning-service/
[46] https://docs.microsoft.com/en-us/azure/data-explorer/

[47] Makridis, Michail; Mattas, Konstantinos; Ciuffo, Biagio, 2017, "Traffic simulation data for Antwerp", https://doi.org/10.7910/DVN/BCUYIR, Harvard Dataverse
[48] https://docs.microsoft.com/en-us/azure/event-hubs/authorize-access-azure-active-directory
[49] https://docs.microsoft.com/en-us/azure/event-hubs/authorize-access-shared-access-signature
[50] https://databricks.com/session/optimizing-apache-spark-sql-joins
[51] https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#triggers