Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ασφαλές Spark

Ελέγχοντας την ευτιμότητα των κόμβων σε ένα Δημόσιο Spark Cluster

---

Διπλωματικη Εργασια

του

Φοίβου Ευάγγελου Ανδριώτη

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ασφαλές Spark

Ελέγχοντας την εντιμότητα των κόμβων σε ένα Δημόσιο Spark Cluster

---

Διπλωματικη Εργασια

του

Φοίβου Ευάγγελου Ανδριώτη

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Ιουλίου 2024.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Νεκτάριος Κοζύρης	Γεώργιος Γκούμας	Διονύσιος Πνευματικάτος
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Φοίβος Ευάγγελος
Ανδριώτης Διπλωματούχος
Ηλεκτρολόγος Μηχανικός
και Μηχανικός
Υπολογιστών Ε.Μ.Π.

23η Ιουλίου 2024

Περίληψη

Η διπλωματική εργασία με τίτλο "Safe Spark" παρουσιάζει μια καινοτόμα επέκταση του Apache Spark που στοχεύει στην αξιολόγηση της νομιμότητας κάθε κόμβου, εργαζόμενου σε ένα δημόσιο Spark cluster. Αυτή η εργασία σχεδιάζει και υλοποιεί ένα σύστημα ελέγχου και επαλήθευσης της αξιοπιστίας των executor nodes στην εκτέλεση των εργασιών. Η αρχική διερεύνηση της δυνατότητας speculative execution του Spark για συγκρίσεις εργασιών μεταξύ κόμβων δεν απέδωσε τα αναμενόμενα αποτελέσματα, οδηγώντας στην ανάπτυξη μιας προσαρμοσμένης διαδικασίας ελέγχου.

Η διαδικασία ελέγχου περιλαμβάνει τρία στάδια: επιλογή ελεγχόμενης εργασίας, δημιουργία ελεγκτικών εργασιών και τη φάση ελέγχου. Η εστίαση στις εργασίες ως αντικείμενο ελέγχου επιλέγεται αντί του ελέγχου μεμονωμένων executor nodes λόγω της έλλειψης οριστικών κριτηρίων για την επιλογή κόμβων. Κατά τον προγραμματισμό, κάθε εργασία συνδέεται με τον executor node που την εκτελεί, ευθυγραμμιζόμενη με τη διαδικασία εκτέλεσης του Σπαρκ. Επιπλέον, με τη σύγκριση των αποτελεσμάτων των ελεγχόμενων και ελεγκτικών εργασιών, η νομιμότητα του executor node τίθεται υπό αμφισβήτηση.

Η ελεγχόμενη εργασία επιλέγεται τυχαία, εξασφαλίζοντας αμερόληπτη επιλογή. Οι ελεγκτικές εργασίες δημιουργούνται με βάση μια καθορισμένη ποσότητα ώστε να επιτευχθεί πλειοψηφία για την ορθότητα του αποτελέσματος, ισορροπώντας την ανθεκτικότητα και τους χρόνους εκτέλεσης. Η φάση ελέγχου εξετάζει τα αποτελέσματα των executor nodes, τα οποία επικυρώνονται έναντι της πλειοψηφίας. Σε περίπτωση ισοπαλίας, εκδίδονται διαδοχικές εργασίες μέχρι να επιτευχθεί πλειοψηφία, περιοριζόμενες από τους διαθέσιμους executor nodes.

Αυτός ο καινοτόμος μηχανισμός ελέγχου υποθέτει ότι η πλειοψηφία των κόμβων στο cluster παράγει νόμιμα αποτελέσματα. Αδυναμία επίτευξης συναίνεσης πριν εξαντληθούν οι επιλέξιμοι executor nodes οδηγεί σε ακύρωση του cluster, διασφαλίζοντας την τήρηση της αρχής της πλειοψηφίας των καλοπροαίρετων executor nodes στο ςλυστερ.

Αυτή η διπλωματική εργασία προσφέρει λεπτομερή ανάλυση του σχεδιασμού, της υλοποίησης και της λειτουργικότητας του προτεινόμενου συστήματος ελέγχου, αντιμετωπίζοντας την ανάγκη για έναν αξιόπιστο μηχανισμό αξιολόγησης της νομιμότητας των executor nodes σε ένα δημόσιο Spark cluster.

Λέξεις Κλειδιά

Spark, Executor, Cluster, Έλεγχος, Ασφάλεια, Εκτέλεση, Εργασία

Abstract

The thesis, entitled "Safe Spark" presents an innovative extension to Apache Spark aimed at evaluating the legitimacy of each worker node in a public Spark cluster. Inspired by Hyper-ledger technologies, this work designs and implements a system to audit and verify the credibility of worker nodes in executing tasks. Initially exploring Spark's speculative execution feature for cross-node task comparisons proved unfruitful, leading to the development of a tailored audit process.

The audit process involves three stages: auditee task selection, auditor task creation, and the pivotal audit phase. Focusing on tasks as the audit subjects is chosen over auditing individual worker nodes due to the absence of definitive criteria for node selection. During scheduling, each task is linked to the worker node executing it, aligning with Spark's execution process. Moreover, by comparing auditee and auditor task results, worker node legitimacy is inherently questioned.

The auditee task is chosen randomly, ensuring impartial selection. Auditor tasks are created based on a specified quantity to achieve a majority vote for result correctness, balancing robustness and execution times. The audit phase scrutinizes executor nodes' results, validating them against the majority consensus. In the event of a tie, successive tasks are issued until a majority is reached, limited by the available executor nodes.

This novel audit mechanism assumes the majority of nodes in the cluster produce legitimate results. Failure to reach a consensus before exhausting eligible executor nodes results in cluster invalidation, ensuring adherence to the principle of a majority of benevolent executor nodes in the cluster.

This thesis offers a detailed insight into the design, implementation, and functionality of the proposed audit system, addressing the need for a robust mechanism to evaluate worker node legitimacy in a public Spark cluster.

Keywords

Spark, Audit, Task, Executor, Comparison, Safe

στους γονείς μου

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Πρόλογος

Καλωσορίσατε στην τεκμηρίωση της διπλωματικής εργασίας με τίτλο ¨Safe Spark: Έλεγχος της εντιμότητας των κόμβων σε ενα Δημόσιο Spark Clusters¨. Αυτό το έγγραφο λειτουργεί ως ολοκληρωμένος οδηγός για την καινοτόμα επέκταση του Apache Spark, σχεδιασμένη για την αξιολόγηση της νομιμότητας των κόμβων εκτελεστών(executor nodes σε ένα Spark cluster.

Στο ταχέως εξελισσόμενο τοπίο της επεξεργασίας μεγάλων δεδομένων, το Apache Spark έχει αναδειχθεί σε ένα ισχυρό εργαλείο για την εκτέλεση πολύπλοκων υπολογισμών σε μεγάλα σύνολα δεδομένων που κατανέμονται σε clusters. Ωστόσο, η ανοιχτότητα των δημόσιων Spark clusters, όπου οποιοσδήποτε μπορεί να εγγραφεί ως executor node, εγείρει ανησυχίες σχετικά με την αξιοπιστία των κόμβων που εκτελούν εργασίες.

Αυτή η διπλωματική εργασία εξετάζει τον σχεδιασμό και την υλοποίηση ενός καινοτόμου συστήματος ελέγχου. Η διαδικασία ελέγχου στοχεύει στην επαλήθευση της νομιμότητας κάθε executor node, διασταυρώνοντας τα αποτελέσματα που παράγει για μια συγκεκριμένη εργασία με τα αποτελέσματα άλλων κόμβων που εκτελούν την ίδια εργασία.

Το ταξίδι ξεκινά με την εξερεύνηση της δυνατότητας speculative execution του Σπαρκ για συγκρίσεις εργασιών μεταξύ κόμβων, που οδήγησε στην ανάπτυξη μιας προσαρμοσμένης διαδικασίας ελέγχου. Ο μηχανισμός ελέγχου επικεντρώνεται στις εργασίες αντί για μεμονωμένους executor nodes, ευθυγραμμιζόμενος με την αμεροληψία της τυχαιότητας κατά την επιλογή εργασιών.

Τα κεφάλαια αυτού του εγγράφου εμβαθύνουν στη θεωρία του Σπαρκ, την εισαγωγή της ιδέας του ελέγχου, τα στάδια της διαδικασίας ελέγχου, τις τροποποιήσεις στον πηγαίο κώδικα, και τα πειράματα που επιβεβαιώνουν την αποτελεσματικότητα του προτεινόμενου μηχανισμού. Θέματα όπως η απόλυτη πλειοψηφία έναντι του αποτελέσματος με τις περισσότερες ψήφους, ο υπολογισμός της ανοχής σε κακόβουλους κόμβους και η εξισορρόπηση μεταξύ ασφάλειας και απόδοσης συζητούνται διεξοδικά.

Το έγγραφο ολοκληρώνεται με μια προτεινόμενη στρατηγική ελέγχου, εκτιμήσεις για τη διατήρηση της ακεραιότητας του cluster, και σκέψεις για το πόσες φορές μπορεί ένας κακόβουλος κόμβος να αποφύγει την ανίχνευση.

Αυτή η τεκμηρίωση στοχεύει να προσφέρει στους αναγνώστες μια λεπτομερή κατανόηση του σχεδιασμού, της υλοποίησης και της λειτουργικότητας του προτεινόμενου συστήματος ελέγχου. Αποτελεί έναν πολύτιμο πόρο για ερευνητές, προγραμματιστές και επαγγελματίες στον τομέα της κατανεμημένης πληροφορικής και της ανάλυσης μεγάλων δεδομένων.

Σας ευχαριστούμε που ξεκινήσατε αυτή την εξερεύνηση του Safe Spark, και ελπίζουμε ότι αυτό το έγγραφο θα αποδειχθεί διδακτικό και ενημερωτικό.

Κεφάλαιο 1

Safe Spark - Εισαγωγή

Το Big Data είναι ένας ταχέως εξελισσόμενος τομέας που έχει φέρει επανάσταση στον τρόπο με τον οποίο οι οργανισμοί επεξεργάζονται, αναλύουν και εξάγουν πληροφορίες από τεράστιες ποσότητες δεδομένων. Ο συνεχώς αυξανόμενος όγκος, η ταχύτητα και η ποικιλία των δεδομένων έχουν καταστήσει αναγκαία την ανάπτυξη κλιμακούμενων και αποδοτικών πλαισίων επεξεργασίας δεδομένων. Αυτά τα πλαίσια επιτρέπουν την κατανεμημένη υπολογιστική επεξεργασία σε μεγάλες βάσεις δεδομένων σε συστοιχίες μηχανών, παρέχοντας την απαραίτητη υπολογιστική ισχύ για την αντιμετώπιση των φορτίων εργασίας του Big Data.

Μεγάλες οργανώσεις και ερευνητικά ιδρύματα επενδύουν σημαντικά στην ανάπτυξη και διατήρηση υποδομών κατανεμημένων υπολογισμών για να αξιοποιήσουν τις δυνατότητες των αναλύσεων Big Data. Ωστόσο, η χρήση τέτοιων υποδομών σε δημόσιο περιβάλλον, όπου μη έμπιστοι χρήστες μπορούν να εγγράφονται ως executor nodes, εγείρει ανησυχίες σχετικά με την αξιοπιστία των κόμβων που εκτελούν εργασίες. Η διασφάλιση της νομιμότητας των executor nodes σε μια δημόσια πλατφόρμα Big Data είναι ζωτικής σημασίας για τη διατήρηση της ακεραιότητας και της ασφάλειας των λειτουργιών επεξεργασίας δεδομένων.

Δεδομένου ότι το Apache Spark είναι ένα δημοφιλές πλαίσιο κατανεμημένων υπολογισμών που χρησιμοποιείται σε διάφορες βιομηχανίες και ερευνητικούς τομείς, είναι απαραίτητο να αντιμετωπιστεί το ζήτημα του ελέγχου της νομιμότητας των executor nodes σε δημόσια Spark clusters. Τα δημόσια Spark clusters αποζημιώνουν τους executor nodes για την εκτέλεση εργασιών, καθιστώντας τα ευάλωτους σε κακόβουλες συμπεριφορές. Κακόβουλοι κόμβοι μπορούν να παραποιήσουν τα αποτελέσματα των εργασιών, να θέσουν σε κίνδυνο την ακεραιότητα των δεδομένων και να διαταράξουν τη συνολική διαδικασία επεξεργασίας δεδομένων. Για να μετριαστούν αυτοί οι κίνδυνοι, είναι απαραίτητος ένας μηχανισμός ελέγχου που να επαληθεύει τη νομιμότητα των executor nodes και να διασφαλίζει την ακρίβεια των αποτελεσμάτων των εργασιών.

Αυτή η διατριβή παρουσιάζει μια καινοτόμα επέκταση στο Apache Spark, που ονομάζεται ¨Safe Spark¨, σχεδιασμένη να αξιολογεί την αξιοπιστία των executor nodes σε ένα δημόσιο Spark cluster. Το Safe Spark εισάγει έναν νέο μηχανισμό ελέγχου για να επαληθεύει τη νομιμότητα των executor nodes διασταυρώνοντας τα αποτελέσματα που παράγουν για μια δεδομένη εργασία με τα αποτελέσματα άλλων κόμβων που εκτελούν την ίδια εργασία. Αυτό χρησιμοποιείται για να εντοπιστούν διαφορές που υποδηλώνουν αν ο executor node που τα παρήγαγε είναι καλόβουλος ή όχι.

Το Spark έχει ήδη εισαγάγει τη φιλοσοφία της εκτέλεσης Εργασιών σε περισσότερους

από έναν κόμβους όταν ένας κόμβος καθυστερεί την εκτέλεση μιας Εργασίας (speulative execution). Κατά τον σχεδιασμό, αυτή η μέθοδος εξετάστηκε μαζί με τις αποδείξεις μηδενικής γνώσης και την ομομορφική κρυπτογράφηση. Η απλότητα και η ανθεκτικότητα της προτεινόμενης μεθόδου υπερείχαν σημαντικά του επιπλέον κόστους που εισήγαγαν οι αποδείξεις μηδενικής γνώσης, καθώς και της πολυπλοκότητας της ομομορφικής κρυπτογράφησης.

Κεφάλαιο 2

Safe Spark - Θεωρία του Spark

2.1 Εισαγωγή

Το Apache Spark αποτελεί μια θεμελιώδη τεχνολογία στον τομέα των κατανεμημένων υπολογισμών και της ανάλυσης big data. Προσφέρει μια ενοποιημένη πλατφόρμα για την επεξεργασία μεγάλων συνόλων δεδομένων και την εφαρμογή σύνθετων αγωγών ανάλυσης δεδομένων [1, 2]. Σε αυτό το ακαδημαϊκό έγγραφο, εμβαθύνουμε στην ουσία του Apache Spark, αποσαφηνίζοντας τα αρχιτεκτονικά του στοιχεία, τα υπολογιστικά του παραδείγματα και τους επιχειρησιακούς μηχανισμούς του.

2.2 Τι είναι το Apache Spark·

Το Apache Spark είναι ένα πλαίσιο κατανεμημένων υπολογισμών ανοιχτού κώδικα, σχεδιασμένο για κλιμακούμενη, ανθεκτική σε σφάλματα και υψηλής απόδοσης επεξεργασία δεδομένων [1]. Αναπτύχθηκε στο AMPLab του UC Berkeley το 2009 και αργότερα δωρήθηκε στο Apache Software Foundation. Σε αντίθεση με τα παραδοσιακά πλαίσια επεξεργασίας παρτίδων όπως το Hadoop MapReduce, το Apache Spark χρησιμοποιεί υπολογισμούς στη μνήμη για να επιταχύνει την επεξεργασία δεδομένων και προσφέρει ένα ευέλικτο προγραμματιστικό μοντέλο που υποστηρίζει διάφορα φορτία επεξεργασίας δεδομένων [3].

2.3 Κύρια Στοιχεία του Apache Spark

Το Apache Spark αποτελείται από αρκετά βασικά στοιχεία που επιτρέπουν την αποδοτική κατανεμημένη επεξεργασία δεδομένων:

2.3.1 Resilient Distributed Datasets (RDDs)

Τα RDDs είναι η θεμελιώδης λογική αφαίρεση στο Apache Spark, αντιπροσωπεύοντας αμετάβλητες κατανεμημένες συλλογές δεδομένων που κατατμούνται σε ένα σύμπλεγμα μηχανών. Τα RDDs υποστηρίζουν ανοχή σε σφάλματα και επιτρέπουν παράλληλη επεξεργασία, αξιοποιώντας πληροφορίες γενεαλογίας για την ανάκτηση χαμένων τμημάτων [2].

2.3.2 Directed Acyclic Graphs (DAGs)

Το Apache Spark χρησιμοποιεί DAGs για τη βελτιστοποίηση της εκτέλεσης εργασιών επεξεργασίας δεδομένων. Με την αναπαράσταση των υπολογισμών ως ένα κατευθυνόμενο γράφημα εξαρτήσεων, η μηχανή εκτέλεσης του Spark μπορεί να οργανώσει παράλληλη και διαδοχική εκτέλεση, ελαχιστοποιώντας την ανταλλαγή δεδομένων και μεγιστοποιώντας τη χρήση των πόρων [1].

2.3.3 Μετασχηματισμοί και Ενέργειες

Το Apache Spark παρέχει ένα πλούσιο σύνολο μετασχηματισμών και ενεργειών που επιτρέπουν στους χρήστες να χειρίζονται και να αναλύουν κατανεμημένα σύνολα δεδομένων [3]. Οι μετασχηματισμοί, όπως το map, το filter και το reduce, επιτρέπουν τη δημιουργία νέων RDDs από υπάρχοντα, ενώ οι ενέργειες, όπως το count, το collect και το save, ενεργοποιούν υπολογισμούς και επιστρέφουν αποτελέσματα στο πρόγραμμα driver [2].

2.4 Πώς Λειτουργεί το Apache Spark

Το Apache Spark λειτουργεί με κατανεμημένο και ανθεκτικό τρόπο, χρησιμοποιώντας μια αρχιτεκτονική master-worker για το συντονισμό και την εκτέλεση εργασιών επεξεργασίας δεδομένων σε ένα σύμπλεγμα μηχανών:

2.4.1 Cluster Manager

Το Apache Spark μπορεί να λειτουργήσει με διάφορους cluster managers, όπως το Apache Mesos, το Hadoop YARN, ή τον αυτόνομο cluster manager του. Ο cluster manager κατανέμει πόρους και προγραμματίζει εργασίες στους κόμβους εργασίας του συμπλέγματος [3].

2.4.2 Driver Program

Το πρόγραμμα driver είναι το σημείο εισόδου για τις εφαρμογές Spark, συντονίζοντας την εκτέλεση των εργασιών και επικοινωνώντας με τον cluster manager για την κατανομή των πόρων. Το πρόγραμμα driver ορίζει τη λογική της εφαρμογής και συντονίζει την εκτέλεση των μετασχηματισμών και των ενεργειών σε κατανεμημένα σύνολα δεδομένων [2].

2.4.3 Worker Nodes

Οι κόμβοι εργασίας φιλοξενούν διεργασίες executor, οι οποίες εκτελούν εργασίες για λογαριασμό του προγράμματος driver. Κάθε κόμβος εργασίας είναι υπεύθυνος για τη διαχείριση των κατανεμημένων πόρων του και την εκτέλεση υπολογισμών σε τοπικά τμήματα δεδομένων [1].

2.4.4 Executor Processes

Οι διεργασίες executor εκτελούνται στους κόμβους εργασίας και αναλαμβάνουν τις εργασίες που τους αναθέτει το πρόγραμμα driver. Οι executors διαχειρίζονται την αποθήκευση δεδομένων στη μνήμη, αποθηκεύουν προσωρινά συχνά προσπελάσιμα τμήματα RDD και εκτελούν υπολογισμούς παράλληλα σε διαθέσιμους πυρήνες CPU [3].

2.4.5 Εκτέλεση Εργασιών

Το Apache Spark διασπά τις εργασίες επεξεργασίας δεδομένων σε μικρότερες μονάδες εργασίας που ονομάζονται tasks, οι οποίες εκτελούνται παράλληλα από τις διεργασίες executor. Η εκτέλεση των tasks περιλαμβάνει τη φόρτωση τμημάτων δεδομένων στη μνήμη, την εφαρμογή μετασχηματισμών και την εκτέλεση υπολογισμών σύμφωνα με το DAG εξαρτήσεων [2].

2.5 Διαδικασία Υποβολής

2.5.1 Υποβολή Εργασίας

Ένας χρήστης γράφει μια εφαρμογή χρησιμοποιώντας τα Spark APIs (όπως το DataFrame API ή το RDD API) και την υποβάλλει σε ένα Spark Cluster. Η εφαρμογή ορίζει μια σειρά από μετασχηματισμούς (π.χ., map, filter) και ενέργειες (π.χ., count, collect) [3].

2.5.2 Driver Program

Το πρόγραμμα driver εκτελείται στον κόμβο Master και είναι υπεύθυνο για το συντονισμό της εκτέλεσης της εφαρμογής Spark. Το driver δημιουργεί ένα SparkContext, το οποίο είναι το κύριο σημείο εισόδου για τις λειτουργίες του Spark. Στη συνέχεια, μετατρέπει τον κώδικα του χρήστη που περιέχει μετασχηματισμούς και ενέργειες σε ένα λογικό DAG σταδίων [1].

2.5.3 DAG Scheduler

Ο DAG scheduler διαιρεί το λογικό γράφημα σε μικρότερα σύνολα εργασιών, γνωστά ως στάδια. Κάθε στάδιο αποτελείται από εργασίες που μπορούν να εκτελεστούν παράλληλα. Καθορίζει επίσης το βέλτιστο σχέδιο εκτέλεσης, διαχειριζόμενος τον προγραμματισμό και την ανάκαμψη από σφάλματα [2].

2.5.4 Προγραμματισμός και Εκτέλεση Εργασιών

Οι εργασίες αποστέλλονται στους κόμβους εργασίας (executors) του συμπλέγματος από τον cluster manager (π.χ., YARN, Mesos, Kubernetes, ή τον αυτόνομο cluster manager του Spark). Κάθε κόμβος εργασίας εκτελεί τις εργασίες που του έχουν ανατεθεί. Οι εργασίες εκτελούν τον πραγματικό υπολογισμό στα τμήματα δεδομένων [1].

2.5.5 Shuffling

Αν μια εργασία απαιτεί δεδομένα από άλλο στάδιο (π.χ., κατά τη διάρκεια μιας λειτουργίας συνένωσης ή μείωσης), τα δεδομένα ανταλλάσσονται μεταξύ των κόμβων. Η ανταλλαγή (shuffling) περιλαμβάνει την ανακατανομή των δεδομένων στο σύμπλεγμα, κάτι που μπορεί να είναι χρονοβόρο και κοστοβόρο σε όρους πόρων [2].

2.5.6 Εκτέλεση και Παρακολούθηση

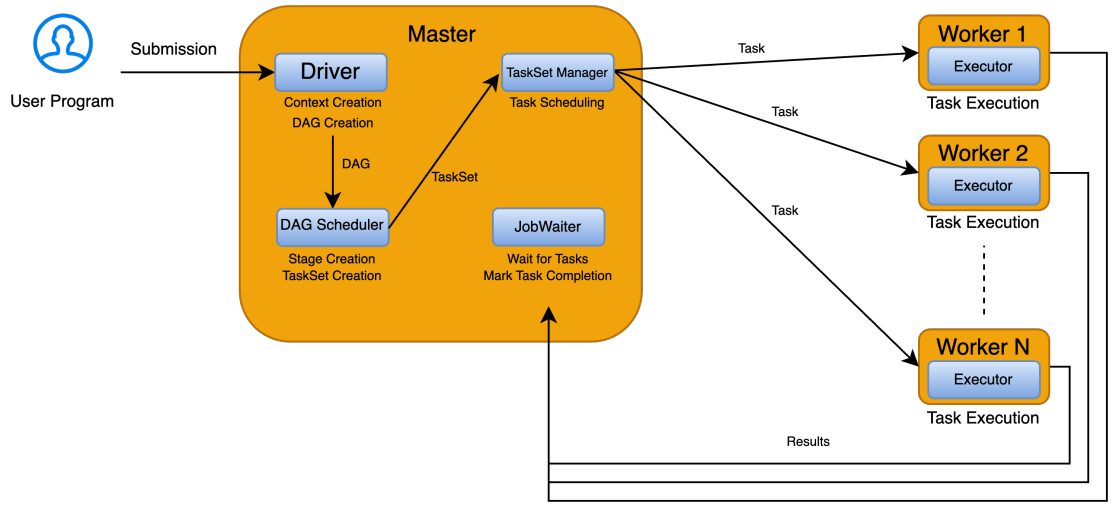
Οι executors εκτελούν τις εργασίες και γράφουν τα ενδιάμεσα αποτελέσματα στη μνήμη ή στο δίσκο, ανάλογα με τις ανάγκες. Το driver παρακολουθεί την πρόοδο των εργασιών και διαχειρίζεται τυχόν αποτυχίες εργασιών, επαναυποβάλλοντας τις αποτυχημένες εργασίες [1].

2.5.7 Συλλογή Αποτελεσμάτων

Όταν όλες οι εργασίες ολοκληρωθούν, το τελικό αποτέλεσμα επιστρέφεται στο πρόγραμμα driver. Το driver μπορεί στη συνέχεια να πραγματοποιήσει περαιτέρω ενέργειες στο αποτέλεσμα, όπως τη συλλογή του στον κόμβο πελάτη, την αποθήκευσή του σε ένα σύστημα αρχείων ή την παρουσίασή του στον χρήστη [3].

2.5.8 Διάγραμμα

Το παρακάτω διάγραμμα απεικονίζει τη ροή της διαδικασίας υποβολής μιας εργασίας σε ένα Apache Spark Cluster. Ξεκινώντας από την υποβολή της εργασίας από τον χρήστη, το πρόγραμμα driver συντονίζει την εκτέλεση των εργασιών στους κόμβους εργασίας. Με μαύρο περιγράφονται οι ενέργειες που εκτελούνται από την αρχική διαδικασία. Με πράσινο περιγράφονται οι ενέργειες που προστίθενται από την επέκταση Safe Spark.



Σχήμα 2.1: *Η Διαδικασία Υποβολής*

Κεφάλαιο 3

Safe Spark: Εισαγωγή στην Ιδέα

Αυτή η υλοποίηση του Spark στοχεύει στην αξιολόγηση της νομιμότητας κάθε executor node, διασταυρώνοντας τα αποτελέσματα που παράγει με αυτά που παράγουν άλλοι κόμβοι για την ίδια εργασία. Η ιδέα αυτή εμπνέεται από ένα δημόσιο σύστημα Spark, όπου οποιοσδήποτε χρήστης μπορεί να εγγραφεί ως executor node, να προσφέρει τους υπολογιστικούς του πόρους για την εκτέλεση εργασιών και να αποζημιωθεί για τη χρήση τους. Από όλα τα θέματα που εισάγει αυτή η ιδέα, η παρούσα διατριβή επικεντρώνεται στον σχεδιασμό και την υλοποίηση ενός συστήματος που ελέγχει και επαληθεύει τη νομιμότητα κάθε executor node.

Το Spark από τον σχεδιασμό του έχει τη δυνατότητα να προγραμματίζει την εκτέλεση πολλαπλών εμφανίσεων της ίδιας εργασίας ως μέτρο κατά των εργασιών που παραμένουν σε κατάσταση επεξεργασίας για μεγάλο χρονικό διάστημα. Αυτό το χαρακτηριστικό ονομάζεται speculative εκτέλεση. Στην αρχή της διαδικασίας σχεδιασμού, ο στόχος ήταν να μελετηθεί αυτό το χαρακτηριστικό και να αξιοποιηθεί για να προγραμματίσει εμφανίσεις της ίδιας εργασίας με τον περιορισμό ότι κάθε εμφάνιση εκτελείται σε διαφορετικό executor node. Αυτός ο περιορισμός εξασφαλίζει ότι τα παραγόμενα αποτελέσματα για κάθε εκτέλεση είναι συγκρίσιμα. Ωστόσο, η αξιοποίηση του χαρακτηριστικού της speculative εκτέλεσης αποδείχθηκε ατελέσφορη. Επιστρέφοντας στην αρχή, έγινε σαφές ότι η διαδικασία έπρεπε να σχεδιαστεί από το μηδέν. Για να δοθεί ένα πλαίσιο στη διαδικασία, ακολουθούν οι ορισμοί τόσο για τη διαδικασία όσο και για τους συμμετέχοντες.

Η διαδικασία ελέγχου περιλαμβάνει τα εξής στάδια:

- Η επιλογή μιας εργασίας στην οποία θα πραγματοποιηθεί ο έλεγχος, η οποία θα ονομάζεται στο εξής *ελεγχόμενη* εργασία.
- Ο προγραμματισμός πολλαπλών εμφανίσεων της ίδιας εργασίας σε διαφορετικούς κόμβους εργασίας, που θα ονομάζονται στο εξής *εργασίες ελεγκτών*.
- Η σύγκριση των αποτελεσμάτων που παράγονται από τις ελεγχόμενες και τις ελεγκτικές εργασίες, που θα ονομάζεται στο εξής *φάση ελέγχου*.

Σε αυτό το σημείο, ένα εύλογο ερώτημα είναι γιατί η διαδικασία ελέγχου επικεντρώνεται στις εργασίες, παρόλο που ο στόχος είναι να επαληθευτεί η νομιμότητα κάθε executor node. Η εξήγηση πίσω από την εστίαση στις εργασίες είναι ότι δεν υπάρχουν προφανή κριτήρια για την επιλογή ενός executor node για έλεγχο έναντι κάποιου άλλου. Επίσης, η ανάπτυξη ενός πρωτοκόλλου που επισημαίνει έναν executor node ως ύποπτο και τον σηματοδοτεί για

έλεγχο φαίνεται υπερβολική σε σύγκριση με την απλή και αποτελεσματική τυχαία επιλογή. Κατά τον προγραμματισμό, κάθε εργασία συνδέεται με τον executor nodes στον οποίο θα εκτελεστεί. Κατά συνέπεια, η τυχαία επιλογή μιας εργασίας για έλεγχο συνδέεται άμεσα με την τυχαία επιλογή ενός executor node για έλεγχο. Επίσης, αυτό ευθυγραμμίζεται απόλυτα με τη διαδικασία που ακολουθεί το Σπαρκ κατά την εκτέλεση εργασιών. Επιπλέον, καθώς τα αποτελέσματα των ελεγκτικών εργασιών επίσης συγκρίνονται κατά τη διάρκεια της διαδικασίας, οι executor nodes που τις εκτελούν τίθενται επίσης υπό αμφισβήτηση ως προς τη νομιμότητά τους. Θα ακολουθήσει περαιτέρω ανάλυση σχετικά με τα κριτήρια ελέγχου.

Αφού επιλεγεί η ελεγχόμενη εργασία και προγραμματιστεί το σύνολο των εργασιών, προγραμματίζεται ένα δεύτερο σύνολο εργασιών που περιέχει τις ελεγκτικές εργασίες με βάση τον προαναφερθέντα περιορισμό προγραμματισμού. Για κάθε εργασία, το Σπαρκ λαμβάνει προσφορές από τους διαθέσιμους executor nodes και στη συνέχεια ελέγχει την επιλεξιμότητα του κάθε ενός όσον αφορά τους διαθέσιμους πόρους. Για τις ελεγκτικές εργασίες, έχει προστεθεί μια επιπλέον συνθήκη για να ελεγχθεί αν ένας executor node έχει ήδη εκτελέσει την ελεγχόμενη εργασία ή μια ελεγκτική εργασία. Για να ξεκινήσει η φάση ελέγχου, πρέπει τόσο οι ελεγχόμενες όσο και οι ελεγκτικές εργασίες να έχουν παράξει αποτελέσματα. Το επόμενο βήμα μετά την παραγωγή των αποτελεσμάτων είναι η ολοκλήρωση. Ωστόσο, αυτές οι εργασίες, όντας μέρος του ελέγχου, δεν σημειώνονται ως ολοκληρωμένες μέχρι να ολοκληρωθεί ο έλεγχος και να επαληθευτεί η νομιμότητά τους.

Κεφάλαιο 4

Safe Spark - Ο Έλεγχος

Όπως αναφέρθηκε παραπάνω, η διαδικασία ελέγχου περιλαμβάνει τρεις φάσεις. Ενώ η επιλογή της ελεγχόμενης εργασίας και η δημιουργία των ελεγκτικών εργασιών δεν είναι περίπλοκες και βασίζονται στην αμεροληψία της τυχαιότητας, η φάση του ελέγχου απαιτούσε στρατηγική. Για λόγους σαφήνειας, θα περιγραφούν και οι τρεις φάσεις, ωστόσο το επίκεντρο θα βρίσκεται στη φάση του ελέγχου.

4.1 Επιλογή Ελεγχόμενης Εργασίας

Η ελεγχόμενη εργασία επιλέγεται χρησιμοποιώντας τη συνάρτηση τυχαιότητας της Scala. Μια απλή αλλά αποτελεσματική ρίψη μεταξύ 1 και του αριθμού των εργασιών στο σύνολο εργασιών.

4.2 Δημιουργία Ελεγκτικών Εργασιών

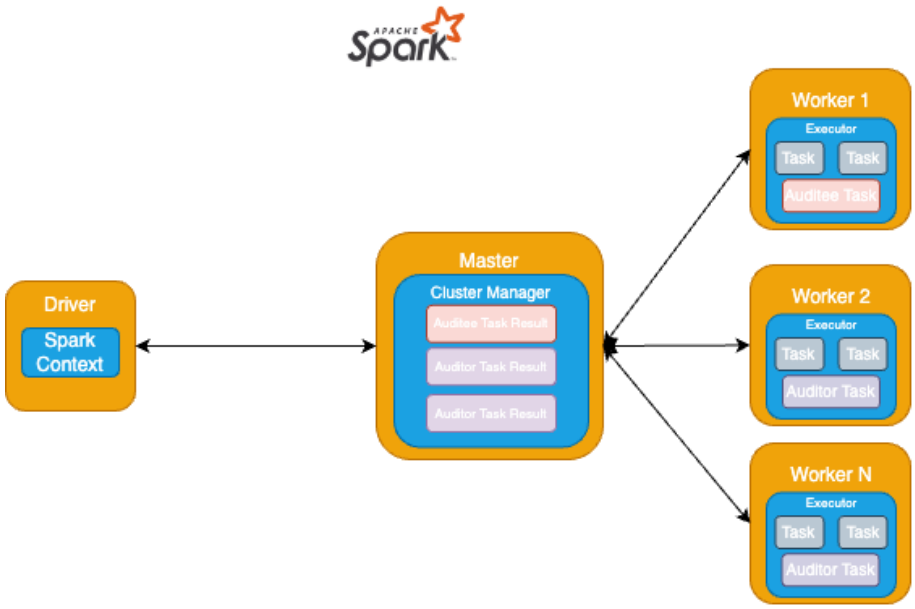
Το σύνολο εργασιών που περιέχει τις ελεγκτικές εργασίες καθορίζεται από μια μοναδική μεταβλητή: το μέγεθός του. Ο αριθμός των ελεγκτικών εργασιών πρέπει να είναι ζυγός και μεγαλύτερος από το μηδέν, ώστε να μπορεί να επιτευχθεί πλειοψηφία σχετικά με την ορθότητα του αποτελέσματος. Όσο μεγαλύτερος είναι ο αριθμός των ελεγκτικών εργασιών, τόσο πιο αξιόπιστη γίνεται η διαδικασία ελέγχου. Ωστόσο, αυτό έχει ως αποτέλεσμα την αύξηση των χρόνων εκτέλεσης του Σταδίου.

4.3 Φάση Ελέγχου

Κάθε εκτέλεση Εργασίας κατά τη διάρκεια της φάσης ελέγχου σημαίνει ότι ο εκχωρημένος κόμβος εκτελεστή για την εργασία θα ελεγχθεί ως προς τη νομιμότητα του αποτελέσματός του συγκρίνοντάς το με τα αποτελέσματα των άλλων κόμβων εκτελεστών που εκτελούν την ίδια Εργασία. Το κριτήριο νομιμότητας βασίζεται στην παραδοχή ότι η πλειοψηφία των κόμβων εκτελεστών στο συγκεκριμένο δημόσιο Spark cluster είναι καλοπροαίρετη, παράγοντας έτσι νόμιμα αποτελέσματα. Σε αυτή την περίπτωση, δεδομένου ενός συνόλου αποτελεσμάτων που παράγονται από κόμβους εκτελεστών για την εκτέλεση της ίδιας εργασίας, ο έλεγχος θα αναγνωρίσει το αποτέλεσμα που παράγεται από την πλειοψηφία των κόμβων εκτελεστών ως το σωστό και θα χαρακτηρίσει όλους τους κόμβους εκτελεστές που αποκλίνουν από αυτό το

αποτέλεσμα ως κακόβουλους. Αυτό το σχήμα επικύρωσης απαιτεί να μην υπάρχει ισοπαλία στο τέλος της φάσης ελέγχου. Για την κάλυψη αυτής της απαίτησης, έχει υλοποιηθεί ένας μηχανισμός που εκδίδει, διαδοχικά, επιπλέον εκτελέσεις της ελεγκτικής εργασίας μέχρι να εμφανιστεί πλειοψηφία. Αν προκύψει ισοπαλία μετά την εκτέλεση των αρχικών ελεγκτικών εργασιών, εκδίδεται μια νέα εργασία για να εκτελεστεί από έναν εκτελεστή που δεν έχει ποτέ παράγει αποτέλεσμα για αυτήν την εργασία. Το αποτέλεσμα αυτό προστίθεται στα παραχθέντα αποτελέσματα, προκαλώντας το μηχανισμό ελέγχου να επανεκτιμήσει αν μπορεί να καταλήξει σε συμπέρασμα ή αν θα ζητήσει μια νέα εργασία tie-breaker. Η έκδοση των εργασιών tie-breaker περιορίζεται από τον αριθμό των εκτελεστών σε ένα cluster. Αν η φάση ελέγχου δεν καταφέρει να φτάσει στην απαιτούμενη πλειοψηφία πριν εξαντληθεί ο αριθμός των επιλέξιμων κόμβων εκτελεστών, τότε όλο το cluster ακυρώνεται καθώς παραβιάζει την αρχή ότι η πλειοψηφία των κόμβων εκτελεστών σε ένα τέτοιο cluster είναι καλοπροαίρετη.

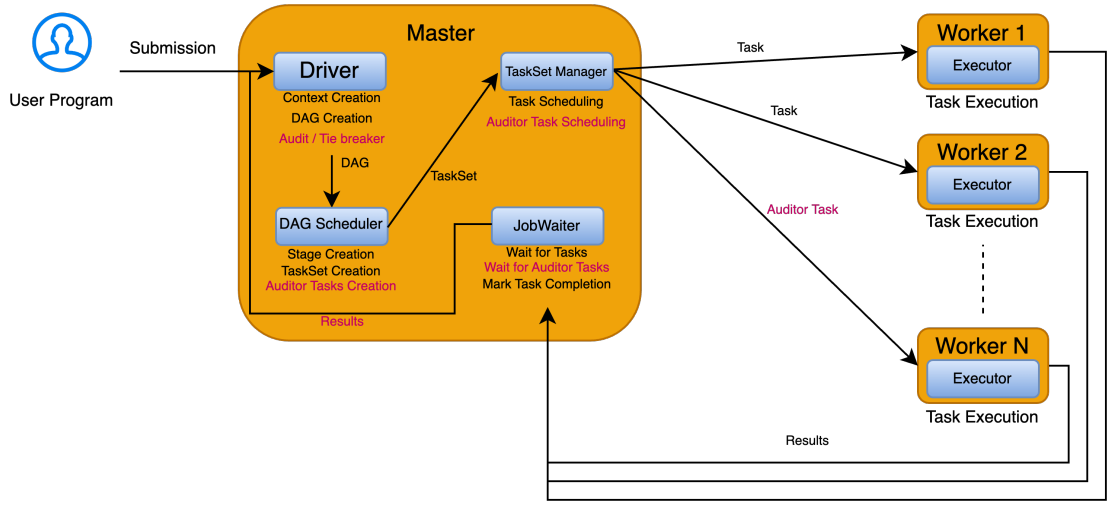
4.4 Διάγραμμα Διαδικασίας Ελέγχου



Σχήμα 4.1: *Η Διαδικασία Ελέγχου*

4.5 Τροποποιημένη Ροή Υποβολής

Αυτό το διάγραμμα Ροής Υποβολής τονίζει τις αλλαγές που έγιναν στην αρχική ροή υποβολής [2.1] για την υλοποίηση της διαδικασίας ελέγχου.



Σχήμα 4.2: *Η Τροποποιημένη Διαδικασία Υποβολής*

4.6 Μοντέλο Αντιπάλου

Αυτή η υλοποίηση του Apache Spark έχει σχεδιαστεί για να χρησιμοποιείται σε δημόσιο περιβάλλον, όπου ένα Spark cluster αρχικοποιείται με έναν **έμπιστο Κεντρικό κόμβο**. Ο Κεντρικός κόμβος είναι υπεύθυνος για τη διαχείριση του cluster, τον προγραμματισμό των εργασιών που θα εκτελεστούν από τους executor nodes, την έκδοση των ελεγκτικών εργασιών και τη συλλογή των αποτελεσμάτων. Οι κόμβοι εργαζομένων είναι υπεύθυνοι για την εκτέλεση των εργασιών που τους αναθέτει ο Κεντρικός κόμβος και για τη συμμετοχή στη διαδικασία ελέγχου. Το Safe Spark υποθέτει ότι οι executor nodes στο cluster μπορεί να είναι κακόβουλοι και να προσπαθήσουν να λάβουν αμοιβή για εργασίες που δεν εκτέλεσαν ή να παραποιήσουν τα αποτελέσματα των εργασιών που εκτέλεσαν. Για να επεξηγήσουμε το μοντέλο αντιπάλου, θα εξετάσουμε τις κακόβουλες συμπεριφορές που μπορεί να επιδείξει ένας executor node. Ένας κακόβουλος executor node έχει ενδιαφέρον μόνο όταν προσπαθεί να στείλει ένα προϋπολογισμένο ψευδές αποτέλεσμα στον Κεντρικό κόμβο. Ο κακόβουλος executor node μπορεί να είναι είτε άπληστος, στέλνοντας ένα ψευδές αποτέλεσμα κάθε φορά που του ανατίθεται μια εργασία, είτε στρατηγικός, στέλνοντας ένα ψευδές αποτέλεσμα μόνο σε ένα ποσοστό των φορών που του ανατίθεται να εκτελέσει μια εργασία. Ένας στρατηγικός κακόβουλος κόμβος μπορεί να είναι δυσδιάκριτος αν η διαδικασία ελέγχου χαρακτηρίζει έναν κόμβο ως έμπιστο μετά από μια επιτυχημένη διαδικασία ελέγχου και δεν αμφισβητεί ξανά τη νομιμότητά του. Ένας άλλος τύπος αντιπάλου είναι ο συνεργατικός κακόβουλος κόμβος, αυτοί οι κόμβοι συνεργάζονται με άλλους συνεργατικούς κακόβουλους κόμβους για να στείλουν το ίδιο αποτέλεσμα, είτε αυτό είναι το σωστό είτε όχι. Ο προτεινόμενος μηχανισμός ελέγχου έχει σχεδιαστεί για να ανιχνεύει και τους δύο τύπους κακόβουλων executor nodes, διασταυρώνοντας τα αποτελέσματα που παράγουν για μια δεδομένη εργασία με τα αποτελέσματα άλλων κόμβων που εκτελούν την ίδια εργασία. Η διαδικασία ελέγχου υποθέτει ότι η πλειοψηφία των κόμβων στο cluster είναι καλοπροαίρετοι και παράγουν νόμιμα αποτελέσματα. Η διαδικασία ελέγχου έχει σχεδιαστεί για να ανιχνεύει κακόβουλους executor nodes συγκρίνοντας τα αποτελέσματά τους με την πλειοψηφία και χαρακτηρίζοντάς τους ως

κακόβουλους αν αποκλίνουν από την πλειοψηφία.

4.7 Παραδείγματα

4.7.1 5 κόμβοι με διαφορετικό αποτέλεσμα για καθέναν

Αυτό το πείραμα στοχεύει να επιδείξει τη διαδικασία ελέγχου σε ένα σενάριο όπου όλοι οι κόμβοι εκτέλεσης παράγουν διαφορετικά αποτελέσματα. Τα αποτελέσματα είναι τα εξής:

Αποτέλεσμα της διεργασίας του ελεγχόμενου: 1
Αποτέλεσμα της διεργασίας του ελεγκτή: 2
Αποτέλεσμα της διεργασίας του ελεγκτή: 3
Αποτέλεσμα της διεργασίας του ελεγκτή: 4
Αποτέλεσμα της διεργασίας του ελεγκτή: 5
Αποτέλεσμα της διεργασίας του ελεγκτή: 6

Το παράδειγμα αναφέρεται σε ένα σύνολο 5 κόμβων. Η διεργασία του ελεγχόμενου επιλέχθηκε τυχαία και οι διεργασίες των ελεγκτών δημιουργήθηκαν με μέγεθος 2. Τα αποτελέσματα που παρήχθησαν από τις διεργασίες των ελεγκτών ήταν όλα διαφορετικά, με αποτέλεσμα ισοπαλία. Η διαδικασία ελέγχου εκδίδει μια νέα διεργασία ελεγκτή και το αποτέλεσμα, όντας ξανά διαφορετικό από τα προηγούμενα, οδηγεί σε νέα τετραπλή ισοπαλία. Στη συνέχεια, για τελευταία φορά, εκδίδεται νέα διεργασία ελεγκτή και το αποτέλεσμα, όντας και πάλι διαφορετικό από τα προηγούμενα, οδηγεί σε πενταπλή ισοπαλία. Σε αυτό το σημείο, η διαδικασία ελέγχου έχει εξαντλήσει όλους τους διαθέσιμους κόμβους εκτέλεσης και το σύνολο κόμβων ακυρώνεται.

Είναι σημαντικό να σημειωθεί ότι ένα τέτοιο σενάριο είναι εξαιρετικά απίθανο να συμβεί σε πραγματικές συνθήκες. Ωστόσο, είναι σημαντικό να παρουσιαστεί η διαδικασία ελέγχου σε ένα σενάριο χειρότερης περίπτωσης.

4.7.2 5 κόμβοι με κατανομή 2 - 2 - 1

Αυτό το πείραμα στοχεύει να επιδείξει τη διαδικασία ελέγχου σε ένα σενάριο όπου οι κόμβοι εκτέλεσης κατανέμονται σε δύο ομάδες των δύο και μία ομάδα. Τα αποτελέσματα είναι τα εξής:

Αποτέλεσμα της διεργασίας του ελεγχόμενου: 1
Αποτέλεσμα της διεργασίας του ελεγκτή: 1
Αποτέλεσμα της διεργασίας του ελεγκτή: 2
Αποτέλεσμα της διεργασίας του ελεγκτή: 2
Αποτέλεσμα της διεργασίας του ελεγκτή: 3

Το παράδειγμα αναφέρεται σε ένα σύνολο 5 κόμβων. Η διεργασία του ελεγχόμενου επιλέχθηκε τυχαία και οι διεργασίες των ελεγκτών δημιουργήθηκαν με μέγεθος 2. Τα αποτελέσματα που παρήχθησαν από τις διεργασίες των ελεγκτών χωρίζονται σε δύο ομάδες των δύο και μία ομάδα. Η διαδικασία ελέγχου εκδίδει νέα διεργασία ελεγκτή και το αποτέλεσμα, όντας διαφορετικό από τα προηγούμενα, οδηγεί σε τριπλή ισοπαλία. Σε αυτό το σημείο, η διαδικασία

ελέγχου έχει εξαντλήσει όλους τους διαθέσιμους κόμβους εκτέλεσης και το σύνολο κόμβων ακυρώνεται.

Είναι σημαντικό να σημειωθεί ότι ένα τέτοιο σενάριο είναι εξαιρετικά απίθανο να συμβεί σε πραγματικές συνθήκες. Ωστόσο, είναι σημαντικό να παρουσιαστεί η διαδικασία ελέγχου σε ένα σενάριο χειρότερης περίπτωσης.

Κεφάλαιο 5

Safe Spark - Τροποποιήσεις Πηγαίου Κώδικα

Η τροποποίηση του πηγαίου κώδικα του Spark για την υλοποίηση του προαναφερθέντος χαρακτηριστικού ήταν μια απαιτητική διαδικασία. Ο κύριος λόγος ήταν ότι, για να προχωρήσει η υλοποίηση της διαδικασίας ελέγχου, απαιτείτο μια βαθιά κατανόηση της αρχιτεκτονικής του Spark. Επίσης, ο πηγαίος κώδικας χρειαζόταν εκτενή ανάλυση για την ταυτοποίηση των μηχανισμών που χρησιμοποιούνται για την εκτέλεση μιας εργασίας Spark και των συστατικών που εμπλέκονται στη διαδικασία. Η κατανόηση αυτών των συστατικών και των αλληλεπιδράσεών τους ήταν κρίσιμη για τον εντοπισμό των σημείων όπου θα μπορούσε να υλοποιηθεί η διαδικασία ελέγχου. Η διαδικασία ελέγχου υλοποιήθηκε στις παρακάτω κλάσεις:

- DAGScheduler.scala
- Stage.scala
- ResultStage.scala
- ShuffleMapStage.scala
- Task.scala
- ResultTask.scala
- ShuffleMapTask.scala
- TaskSetManager.scala
- TaskSchedulerImpl.scala
- thesisExtension.scala

5.1 DAGScheduler

Αυτή η κλάση αποτελεί το υψηλού επιπέδου επίπεδο προγραμματισμού του Spark. Χρησιμοποιώντας μια στρατηγική προγραμματισμού με βάση τα στάδια, υπολογίζει ένα DAG για κάθε στάδιο μιας εργασίας. Αυτή η οντότητα είναι υπεύθυνη για την παρακολούθηση των RDDs, των αποτελεσμάτων των σταδίων και για τη δημιουργία ενός ελάχιστου προγράμματος

για την εργασία. Κάθε στάδιο απεικονίζεται από ένα σύνολο εργασιών, το οποίο στη συνέχεια υποβάλλεται στον TaskScheduler.

Δεδομένου ότι το DAGScheduler είναι επίσης υπεύθυνο για την αναμονή για την ολοκλήρωση του Σταδίου, εδώ ορίζονται οι μεταβλητές **taskSelectionLimit** και **multiplicity** και μεταδίδονται σε όλες τις άλλες δομές.

Το TaskSelectionLimit καθορίζει τον αριθμό των Εργασιών που θα επιλεγούν για έλεγχο. Η Multiplicity καθορίζει τον αριθμό των κόμβων στους οποίους οι ελεγχόμενες εργασίες θα επανεκτελεστούν. Αυτές οι δύο παράμετροι μεταβιβάζονται στη δημιουργία Σταδίου, καθώς και στη δημιουργία Εργασιών αργότερα.

submitMissingTasks

Αυτή η συνάρτηση είναι υπεύθυνη για την υποβολή των ελλειπόντων εργασιών στον TaskScheduler. Έχει τροποποιηθεί ώστε να περιλαμβάνει και τις εργασίες ελέγχου. Οι εργασίες ελέγχου δημιουργούνται και υποβάλλονται στον TaskScheduler μετά την υποβολή των ελεγχόμενων εργασιών. Οι εργασίες ελέγχου δημιουργούνται βάσει του αριθμού των ελεγκτικών εργασιών που πρέπει να δημιουργηθούν. Οι εργασίες ελέγχου δημιουργούνται για το ίδιο διαμέρισμα όπως και η ελεγχόμενη εργασία. Σε αυτή τη συνάρτηση καλείται η findMissingPartitions για να βρει τα ελλείποντα διαμερίσματα που πρέπει να υπολογιστούν. Επίσης, ο χάρτης partitionMultiplicityMap συμπληρώνεται με τον αριθμό των εργασιών που πρέπει να δημιουργηθούν για κάθε διαμέρισμα. Κάθε Εργασία πρέπει επίσης να γνωρίζει τη πολλαπλότητα της, έτσι ώστε να καλείται η συνάρτηση calculateMultiplicity για να παρέχει αυτές τις πληροφορίες.

5.1.1 submitJob

Αυτή η συνάρτηση είναι υπεύθυνη για την υποβολή μιας εργασίας στο DAGScheduler. Επιστρέφει έναν JobWaiter που χρησιμοποιείται για την αναμονή της ολοκλήρωσης της εργασίας. Το JobWaiter έχει τροποποιηθεί ώστε να περιμένει και την ολοκλήρωση των εργασιών ελέγχου.

5.1.2 handleTaskCompletion

Αυτή η συνάρτηση είναι ένας χειριστής του CompletionEvent. Έχει τροποποιηθεί ώστε να συμβουλεύεται τον χάρτη partitionMultiplicityMap για να ελέγξει αν όλες οι εργασίες για ένα δεδομένο διαμέρισμα έχουν ολοκληρωθεί. Αυτή η αλλαγή ήταν απαραίτητη για να επιτραπεί η ολοκλήρωση της διαδικασίας ελέγχου, δεδομένου ότι το DAGScheduler αρχικά έλεγχε αν είχε ολοκληρωθεί μία εργασία για κάθε ταυτότητα διαμερίσματος πριν σημειώσει την εργασία ως ολοκληρωμένη και ακυρώσει όλες τις άλλες εκκρεμείς εργασίες. Όταν αυτός ο χειριστής λάβει το τελευταίο συμβάν ολοκλήρωσης για ένα διαμέρισμα που έχει επιλεγεί για έλεγχο, θα ενεργοποιήσει τη διαδικασία ελέγχου.

Με βάση τα αποτελέσματα του ελέγχου, θα αποδεχθεί το αποτέλεσμα με τις περισσότερες ψήφους. Αν δεν επιτευχθεί πλειοψηφία, θα δημιουργηθεί και θα υποβληθεί μια εργασία tie breaker στον TaskScheduler.

5.2 Stage.scala

Αυτή η κλάση αντιπροσωπεύει ένα στάδιο στην εργασία του Spark. Δημιουργείται από το DAGScheduler ως μέρος του προγραμματισμού με βάση τα στάδια για μια εργασία. Κάθε στάδιο αναπαρίσταται από ένα TaskSet που περιέχει πλήρως ανεξάρτητες εργασίες. Τα στάδια χαρακτηρίζονται ως ResultStages ή ShuffleMapStages. Το πρώτο είναι το τελικό στάδιο μιας εργασίας και το δεύτερο είναι το στάδιο που παράγει δεδομένα για το επόμενο στάδιο. Τα ResultStage.scala και ShuffleMapStage.scala είναι υποκατηγορίες της Stage.scala.

Αυτή η κλάση έχει τροποποιηθεί για να φέρει, μεταξύ άλλων, μεταδεδομένα όπως η πολλαπλότητα και το taskSelectionLimit. Η πολλαπλότητα είναι ο αριθμός των κόμβων στους οποίους οι ελεγχόμενες εργασίες θα επανεκτελεστούν. Το taskSelectionLimit είναι ο αριθμός των Εργασιών που θα επιλεγούν για έλεγχο. Φέρει επίσης τον χάρτη partitionMultiplicityMap, ο οποίος περιέχει πόσες εργασίες πρέπει να δημιουργηθούν για κάθε διαμέρισμα. Ο χάρτης partitionMultiplicityMap συμπληρώνεται αφού βρεθούν τα ελλείποντα διαμερίσματα.

5.3 ResultStage.scala

Αυτή η κλάση αντιπροσωπεύει το τελικό στάδιο μιας εργασίας. Επεκτείνει την κλάση Stage και είναι υπεύθυνη για την εύρεση της ακολουθίας των ταυτοτήτων διαμερίσματος που πρέπει να υπολογιστούν.

5.3.1 findMissingPartitions

Αυτή η συνάρτηση είναι υπεύθυνη για την εύρεση των ελλειπόντων διαμερισμάτων που πρέπει να υπολογιστούν. Έχει τροποποιηθεί ώστε να περιλαμβάνει τα διαμερίσματα που πρέπει να ελεγχθούν. Αυτό συμβαίνει με την κλήση της συνάρτησης selectForEval από το αρχείο thesisExtension.scala.

5.4 ShuffleMapStage.scala

Αυτή η κλάση αντιπροσωπεύει το στάδιο που παράγει δεδομένα για το επόμενο στάδιο. Επεκτείνει την κλάση Stage και είναι υπεύθυνη για την εύρεση της ακολουθίας των ταυτοτήτων διαμερίσματος που πρέπει να υπολογιστούν.

5.4.1 findMissingPartitions

Αυτή η συνάρτηση είναι υπεύθυνη για την εύρεση των ελλειπόντων διαμερισμάτων που πρέπει να υπολογιστούν. Έχει τροποποιηθεί ώστε να περιλαμβάνει τα διαμερίσματα που πρέπει να ελεγχθούν. Αυτό συμβαίνει με την κλήση της συνάρτησης selectForEval από το αρχείο thesisExtension.scala.

Κεφάλαιο 6

Safe Spark - Προσομοιώσεις

Δεδομένης της μη διαθεσιμότητας ενός επαρκώς μεγάλου Spark cluster για να δοκιμαστεί η διαδικασία ελέγχου (Audit process) στα όρια της, κρίθηκε απαραίτητη η πραγματοποίηση μιας προσομοίωσης. Για τον σκοπό αυτό, αναπτύχθηκε ένα σενάριο σε Python, το οποίο προσομοιώνει τη διαδικασία. Το σενάριο αρχικοποιεί ένα cluster από N κόμβους, οι οποίοι κατηγοριοποιούνται σε τέσσερις διακριτούς τύπους:

- **Νόμιμοι Κόμβοι (L)**: Κόμβοι που παράγουν σταθερά σωστά αποτελέσματα.
- **Πάντα Κακόβουλοι Κόμβοι (A)**: Κόμβοι που παράγουν σταθερά τυχαία λανθασμένα αποτελέσματα.
- **Περιστασιακά Κακόβουλοι Κόμβοι (S)**: Κόμβοι που παράγουν τυχαία λανθασμένα αποτελέσματα με βάση μια πιθανολογική συχνότητα.
- **Συνεργαζόμενοι Περιστασιακά Κακόβουλοι Κόμβοι (C)**: Κόμβοι που συνεργάζονται για να παράγουν τυχαία λανθασμένα αποτελέσματα περιστασιακά, με βάση μια κοινή πιθανολογική συχνότητα.

Μόλις δημιουργηθεί το cluster, το σενάριο παράγει αποτελέσματα για όλους τους κόμβους του cluster για μια δεδομένη εργασία. Στη συνέχεια, ένας κόμβος ορίζεται ως ο υπό έλεγχο (auditee), και M κόμβοι επιλέγονται ως ελεγκτές (auditors), σύμφωνα με την παράμετρο πολλαπλότητας (multiplicity). Η διαδικασία ελέγχου ξεκινά, όπου τα αποτελέσματα των κόμβων διασταυρώνονται. Αν προκύψει ένα αποτέλεσμα συναίνεσης, οι κόμβοι που συμμετείχαν στον έλεγχο αλλά δεν παρήγαγαν το επικρατέστερο αποτέλεσμα, αποβάλλονται από το cluster. Αυτή η επαναληπτική διαδικασία συνεχίζεται μέχρι να εκκαθαριστούν όλοι οι κακόβουλοι κόμβοι από το cluster.

Για να αξιολογηθεί η απόδοση των διαφορετικών τιμών πολλαπλότητας (multiplicity), η προσομοίωση εκτελείται 100 φορές για κάθε τιμή, ώστε να μετριαστεί η επίδραση των ακραίων περιπτώσεων. Οι παρακάτω Key Performance Indicators (KPIs) υπολογίζονται:

- Ο μέσος αριθμός γύρων που απαιτούνται για την εκκαθάριση του cluster.
- Ο μέσος αριθμός ελεγκτικών εργασιών που δημιουργούνται μέχρι την εκκαθάριση του cluster.

- Οι μέσες περιπτώσεις στις οποίες ένας κακόβουλος κόμβος παρήγαγε μη ανιχνεύσιμο ψευδές αποτέλεσμα.
- Οι μέσες περιπτώσεις όπου οι κακόβουλοι κόμβοι ανέτρεψαν επιτυχώς νόμιμους κόμβους.

6.1 Προσομοίωση Προσπάθειας Εξέγερσης

Αυτή η προσομοίωση έχει ως στόχο να δοκιμάσει την ικανότητα ενός cluster να προστατεύσει τον εαυτό του από μεγάλο αριθμό συνεργαζόμενων περιστασιακά κακόβουλων κόμβων.

6.1.1 Ρυθμίσεις

- 60 Νόμιμοι Κόμβοι Εκτέλεσης (Executor Nodes)
- 0 Πάντα Κακόβουλοι Κόμβοι
- 0 Περιστασιακά Κακόβουλοι Κόμβοι
- 40 Συνεργαζόμενοι Περιστασιακά Κακόβουλοι Κόμβοι

Πίνακας 6.1: *Προσομοίωση Προσπάθειας Εξέγερσης*

Multiplici	AVG Rounds Until Clean	AVG Total Tasks	AVG Auditor Tasks	PCT Auditor Tasks	AVG Malicious Got Away	PCT Malicious Got Away	PCT Malicious Took Over
5	281,54	29.561,7	1.407,7	5%	881,33	3%	0%
10	119,97	13.196,7	1.201,14	9%	335,6	2,5%	1%
20	136,28	16.353,6	2.725,83	17%	145,12	0,88%	3%
30	50,66	6.585,8	1.519,89	23%	78,24	1,1%	2%
50	15,65	2.347,5	782,08	34%	33,33	1,4%	2%
80	9,49	1.708,2	706,82	41%	7,88	0,4%	0%

- **Συμπεριληπτικότητα και ανθεκτικότητα της διαδικασίας ελέγχου**
  - Καθώς αυξάνεται η πολλαπλότητα (multiplicity), το ποσοστό των ελεγκτικών εργασιών αυξάνεται σημαντικά, φτάνοντας έως και το 44% στη πολλαπλότητα 80. Αυτό υποδηλώνει μια ανθεκτική διαδικασία ελέγχου, καθώς ελέγχονται περισσότερες εργασίες, ενισχύοντας τη συνολική ανθεκτικότητα του συστήματος.
- **Συμπεριληπτικότητα και αποδοτικότητα κόστους της διαδικασίας ελέγχου**
  - Αν και το ποσοστό των ελεγκτικών εργασιών αυξάνεται, οι συνολικές εργασίες και οι γύροι μέχρι την εκκαθάριση ποικίλλουν. Αρχικά, ο μέσος αριθμός γύρων μέχρι την εκκαθάριση μειώνεται με την αύξηση της πολλαπλότητας, υποδεικνύοντας βελτιωμένη αποδοτικότητα. Ωστόσο, στη πολλαπλότητα 50, υπάρχει μια απότομη αύξηση στους γύρους, υποδεικνύοντας πιθανή αναποτελεσματικότητα

σε εκείνο το σημείο. Συνολικά, η μεγαλύτερη πολλαπλότητα τείνει να ισορροπεί το συμβιβασμό μεταξύ συμπεριληπτικότητας και αποδοτικότητας.

- **Αποτελεσματικότητα στη μείωση των κακόβουλων κόμβων**
  - Η διαδικασία είναι ιδιαίτερα αποτελεσματική στη μείωση του αριθμού των κακόβουλων κόμβων, όπως υποδεικνύεται από τα πολύ χαμηλά ποσοστά κακόβουλων κόμβων που διαφεύγουν και ανατρέπουν νόμιμους κόμβους σε όλες τις πολλαπλότητες. Το ποσοστό των κακόβουλων κόμβων που διαφεύγουν μειώνεται καθώς αυξάνεται η πολλαπλότητα, δείχνοντας ότι η μεγαλύτερη πολλαπλότητα ενισχύει την αποτελεσματικότητα της διαδικασίας ελέγχου.

6.2 Προσομοίωση Τυπικού Σεναρίου

Αυτή η προσομοίωση έχει ως στόχο να δοκιμάσει την ικανότητα ενός cluster να προστατευτεί σε ένα τυπικό σενάριο, όπου όλοι οι τύποι κακόβουλων κόμβων υπάρχουν και οι νόμιμοι κόμβοι αποτελούν την πλειοψηφία.

6.2.1 Ρυθμίσεις

- 80 Νόμιμοι Κόμβοι Εκτέλεσης
- 5 Πάντα Κακόβουλοι Κόμβοι
- 10 Περιστασιακά Κακόβουλοι Κόμβοι
- 5 Συνεργαζόμενοι Περιστασιακά Κακόβουλοι Κόμβοι

Πίνακας 6.2: *Προσομοίωση Τυπικού Σεναρίου*

Multiplic	AVG Rounds Until Clean	AVG Total Tasks	AVG Auditor Tasks	PCT Auditor Tasks	AVG Malicious Got Away	PCT Malicious Got Away	PCT Malicious Took Over
5	790,05	82.955,25	3.950,29	5%	254,04	0,3%	0%
10	790,26	86.927,5	7.902,6	9%	121,14	0,14%	0%
20	226,21	27.145,2	4.524,2	16,6%	54,31	0,20%	0%
30	137,36	17.856,8	4.120,8	23%	30,74	0,17%	0%
50	241,7	36.255	12.085	33%	13,13	0,03%	0%
80	41,18	7.412,4	3.294,4	44%	2,72	0,03%	0%

- **Συμπεριληπτικότητα και ανθεκτικότητα της διαδικασίας ελέγχου**
  - Παρόμοια με τον πρώτο πίνακα, η αύξηση της πολλαπλότητας οδηγεί σε υψηλότερο ποσοστό ελεγκτικών εργασιών. Αυτό υποδηλώνει μια ανθεκτική διαδικασία ελέγχου καθώς ελέγχονται περισσότερες εργασίες, ενισχύοντας την ακεραιότητα του συστήματος.

- **Συμπεριληπτικότητα και αποδοτικότητα κόστους της διαδικασίας ελέγχου**
  - Ο μέσος αριθμός γύρων μέχρι την εκκαθάριση μειώνεται σημαντικά με την αύξηση της πολλαπλότητας, επιδεικνύοντας βελτιωμένη αποδοτικότητα. Για παράδειγμα, στη πολλαπλότητα 5, απαιτούνται 281,54 γύροι για την εκκαθάριση, ενώ στη πολλαπλότητα 80 απαιτούνται μόνο 9,49 γύροι. Αυτό υποδηλώνει ότι η μεγαλύτερη πολλαπλότητα οδηγεί σε πιο αποτελεσματική διαδικασία ελέγχου και εκκαθάρισης.
- **Αποτελεσματικότητα στη μείωση των κακόβουλων κόμβων**
  - Η διαδικασία ελέγχου είναι πολύ αποτελεσματική στη μείωση του αριθμού των κακόβουλων κόμβων, με πολύ χαμηλά ποσοστά κακόβουλων κόμβων που διαφεύγουν και ανατρέπουν. Καθώς αυξάνεται η πολλαπλότητα, ο αριθμός των κακόβουλων κόμβων που διαφεύγουν μειώνεται, υποδεικνύοντας την ενίσχυση της αποτελεσματικότητας της διαδικασίας ελέγχου στις υψηλότερες πολλαπλότητες.

6.3 Σύνοψη

- Σε όλους τους πίνακες, η αύξηση της πολλαπλότητας γενικά βελτιώνει την ανθεκτικότητα, την αποδοτικότητα κόστους και την αποτελεσματικότητα της διαδικασίας ελέγχου.
- Η μεγαλύτερη πολλαπλότητα οδηγεί σε μεγαλύτερο ποσοστό ελεγκτικών εργασιών, λιγότερους γύρους που απαιτούνται για την εκκαθάριση του συστήματος και χαμηλότερα ποσοστά κακόβουλων κόμβων που διαφεύγουν ή ανατρέπουν.
- Αυτό δείχνει ότι μια πιο συμπεριληπτική διαδικασία ελέγχου, που περιλαμβάνει περισσότερες ελεγκτικές εργασίες, συνάδει με τους στόχους της ανθεκτικότητας, της αποδοτικότητας κόστους και της μείωσης των κακόβουλων κόμβων.

Κεφάλαιο 7

Safe Spark - Σκέψεις

7.1 Απόλυτη πλειοψηφία VS Αποτέλεσμα με τις περισσότερες ψήφους

Η διαδικασία ελέγχου, εκ σχεδιασμού, καθιστά τον αριθμό των εκτελεστών σε ένα cluster δυναμικό. Αν ένας κόμβος ελεγχθεί και θεωρηθεί κακόβουλος, τότε αφαιρείται από το cluster. Παρακάτω εξηγούνται και συγκρίνονται τα δύο Κριτήρια Ελέγχου που εξετάστηκαν για τη διαδικασία ελέγχου.

7.1.1 Απόλυτη πλειοψηφία

Η προσέγγιση της απόλυτης πλειοψηφίας απαιτεί ότι μια συγκεκριμένη διαδικασία ελέγχου θα ολοκληρωθεί μόνο όταν ένα αποτέλεσμα έχει την απόλυτη πλειοψηφία έναντι των παραχθέντων αποτελεσμάτων για μια ελεγχόμενη εργασία. Αυτή η στρατηγική απαιτεί να εκδίδεται μια εργασία διάσπασης ισοψηφίας μέχρι να εκπληρωθεί το παραπάνω κριτήριο. Αυτή η προσέγγιση, ενώ είναι πιο αξιόπιστη, είναι επίσης πιο ακριβή, καθώς απαιτεί να διατεθούν περισσότεροι πόροι για την εκτέλεση της διαδικασίας ελέγχου.

7.1.2 Αποτέλεσμα με τις περισσότερες ψήφους

Η προσέγγιση ¨Αποτέλεσμα με τις περισσότερες ψήφους¨ θεωρεί τη διαδικασία ελέγχου ολοκληρωμένη όταν προκύψει το αποτέλεσμα με τις περισσότερες ψήφους. Αυτή η προσέγγιση μπορεί να είναι πιο αποδοτική καθώς απαιτεί λιγότερους πόρους για την εκτέλεση της διαδικασίας ελέγχου. Ωστόσο, είναι λιγότερο αξιόπιστη, καθώς μπορεί να οδηγήσει σε ψευδώς θετικά αποτελέσματα σε σενάριο όπου δύο κακόβουλοι κόμβοι συνεργάζονται για να παράγουν το ίδιο αποτέλεσμα έναντι ενός μόνο ευμενούς κόμβου. Σε ένα πραγματικό σενάριο, το να επιλέγονται τυχαία κακόβουλοι κόμβοι ως ελεγκτές και να συνεργάζονται για το ίδιο αποτέλεσμα μπορεί να συμβει **;;**. Αν συμβεί ένα τέτοιο σενάριο, είναι πολύ πιθανό οι συνεργατικοί κακόβουλοι κόμβοι να εντοπιστούν στους επόμενους γύρους ελέγχου. Αν όχι, αυτό θα σήμαινε ότι η πλειοψηφία των εκτελεστών είναι κακόβουλη, και το cluster έχει παραβιαστεί. Αυτό θα καθιστούσε τη διαδικασία ελέγχου άσκοπη. Ανατρέξτε στο 8.1.1 για μια μελλοντική επέκταση του Safe Spark που μπορεί να ανεχθεί αποτελεσματικά το παραπάνω σενάριο.

7.1.3 Σύγκριση με Παράδειγμα

Για να προσφέρουμε μια καλύτερη κατανόηση των δύο στρατηγικών και να αναδείξουμε τη βασική διαφορά μεταξύ τους, προσφέρεται το ακόλουθο παράδειγμα:

Μια διαδικασία ελέγχου είναι σε εξέλιξη για μια συγκεκριμένη εργασία. Δύο εκτελεστικοί κόμβοι εκτέλεσαν την εργασία και παρήγαγαν το Αποτέλεσμα 1. Τρεις διαφορετικοί κόμβοι εκτέλεσαν την ίδια εργασία και παρήγαγαν το Αποτέλεσμα 2. Τέλος, ένας άλλος κόμβος εκτέλεσε την εργασία και παρήγαγε το Αποτέλεσμα 3. Αυτό μας φέρνει στην ακόλουθη κατάσταση:

- Αποτέλεσμα 1: Δύο ψήφοι
- Αποτέλεσμα 2: Τρεις ψήφοι
- Αποτέλεσμα 3: Μία ψήφος

Η προσέγγιση ¨Αποτέλεσμα με τις περισσότερες ψήφους¨ θα θεωρούσε τη διαδικασία ελέγχου ολοκληρωμένη, επισημαίνοντας το Αποτέλεσμα 2 ως σωστό και τους κόμβους που παρήγαγαν τα Αποτελέσματα 1 και 3 ως κακόβουλους.

Η προσέγγιση της Απόλυτης Πλειοψηφίας θα ζητούσε άλλη μία εκτέλεση της εργασίας για να αποφανθεί αν ο υπολογισμός έφερε το Αποτέλεσμα 2, ή θα συνέχιζε να ζητά εργασίες διάσπασης ισοψηφίας μέχρι να υπάρξει αποτέλεσμα με πάνω από το 50% των ψήφων.

7.2 Υπολογισμός ανοχής σε κακόβουλους κόμβους

Τα πειράματα που πραγματοποιήθηκαν οδηγούν στο συμπέρασμα ότι είναι κρίσιμο να υπολογιστεί η ανοχή σε κακόβουλους κόμβους σε ένα cluster. Για να πραγματοποιηθεί αυτός ο υπολογισμός, πρέπει να ληφθούν υπόψη οι ακόλουθες μεταβλητές:

- Ο αριθμός των εκτελεστικών κόμβων στο cluster.
- Ο αριθμός των ελεγκτικών εργασιών.

Ο αριθμός των εκτελεστικών κόμβων πρέπει να είναι περιττός για να αποφεύγονται ισοψηφίες. Ο αριθμός των ελεγκτικών εργασιών είναι η μεταβλητή που έχει τη μεγαλύτερη επίδραση στον υπολογισμό. Όσο περισσότερες ελεγκτικές εργασίες δημιουργούνται, τόσο πιο αξιόπιστη γίνεται η διαδικασία ελέγχου. Ωστόσο, όσο περισσότερες ελεγκτικές εργασίες δημιουργούνται, τόσο περισσότεροι πόροι απαιτούνται για την ολοκλήρωση της διαδικασίας ελέγχου. Ο αριθμός των ελεγκτικών εργασιών είναι επίσης άμεσα συνδεδεμένος με τον αριθμό των εκτελεστικών κόμβων στο cluster. Ο αριθμός των ελεγκτικών εργασιών δεν μπορεί να είναι μεγαλύτερος από τον αριθμό των εκτελεστικών κόμβων μείον έναν.

7.2.1 Τυπική περίπτωση:

- N εκτελεστικοί κόμβοι
- κ ελεγχόμενες εργασίες

- ρ ελεγκτικές εργασίες
- φ συχνότητα κακόβουλης συμπεριφοράς

Όπου

$$r \times k < N - r$$

Σε ένα τυπικό σενάριο όπου επιλέγονται k εργασίες για έλεγχο, δημιουργούνται r ελεγκτικές εργασίες ανά ελεγχόμενη εργασία. Κατά τη διάρκεια της διαδικασίας ελέγχου, όλοι οι συμμετέχοντες κόμβοι δοκιμάζονται για τη νομιμότητά τους· ένας κόμβος έχει

$$\frac{r \times (k+1)}{N}$$

πιθανότητα να επιλεγεί για έλεγχο. Λαμβάνοντας υπόψη τη συχνότητα κακόβουλης συμπεριφοράς, όπου ένας κόμβος επιλέγει να είναι κακόβουλος ένα συγκεκριμένο αριθμό φορών, η πιθανότητα να εντοπιστεί ένας τέτοιος κόμβος είναι

$$f \times \frac{r \times (k+1)}{N}$$

όπου

$$0 < f < 1$$

7.2.2 Ακραία περίπτωση:

- N εκτελεστικοί κόμβοι
- 1 ελεγχόμενη εργασία
- N-1 ελεγκτικές εργασίες

Αυτή η περίπτωση θα οδηγήσει σε έλεγχο που θα εξετάσει τη νομιμότητα κάθε εκτελεστικού κόμβου στο cluster. Ωστόσο, θα απαιτήσει επίσης μια πολύ ακριβή διαδικασία ελέγχου, καθώς θα απαιτούνται N-1 εκτελέσεις της ελεγχόμενης εργασίας.

7.3 Ποιο είναι το τίμημα της ασφάλειας έναντι της απόδοσης

Υποθέτοντας ότι το Spark προσπαθεί πάντα να κατακερματίσει μια Εργασία σε εργασίες ίσης πολυπλοκότητας, ο υπολογισμός της επίδρασης της διαδικασίας ελέγχου στην απόδοση βασίζεται στον αριθμό των ελεγκτικών εργασιών.

Όσο περισσότερες ελεγκτικές εργασίες δημιουργούνται, τόσο πιο αξιόπιστη γίνεται η διαδικασία ελέγχου. Ιδανικά, για κάθε διαδικασία ελέγχου, ο αριθμός των ελεγκτικών εργασιών πρέπει να είναι ίσος με τον αριθμό των εκτελεστικών κόμβων στο cluster μείον έναν. Αυτό θα σήμαινε ότι η διαδικασία ελέγχου θα ελέγξει τη νομιμότητα κάθε εκτελεστικού κόμβου στο cluster.

$$W + \frac{W}{N} \times M$$

όπου M ο αριθμός των ελεγκτικών εργασιών είναι μεγαλύτερος από 2 και μικρότερος από N.

Η ερμηνεία του παραπάνω τύπου είναι ότι το τίμημα της ασφάλειας έναντι της απόδοσης είναι ότι όσο πιο περιεκτική είναι η διαδικασία ελέγχου, όσον αφορά τον αριθμό των ελεγκτικών εργασιών, τόσο πιο ακριβή γίνεται.

7.4 Κατανομή πόρων

Η διαδικασία ελέγχου απαιτεί την κατανομή πόρων για την εκτέλεση των ελεγκτικών εργασιών. Σε ένα θεωρητικό περιβάλλον όπου ένα Spark cluster αρχικοποιείται με έναν έμπιστο Master κόμβο και έναν απεριόριστο αριθμό εργατικών κόμβων, με την υπόθεση ότι το 50% του πληθυσμού των εργατικών κόμβων είναι ευμενείς, η διαδικασία ελέγχου θα ήταν πλήρως περιεκτική. Αυτό θα σήμαινε ότι για κάθε σύνολο εργασιών που εκτελούνται, ο αριθμός των ελεγκτικών εργασιών θα ήταν ίσος με τον αριθμό των εκτελεστικών κόμβων στο cluster μείον έναν. Αυτό θα οδηγούσε σε μια πολύ ακριβή αλλά απολύτως αξιόπιστη διαδικασία ελέγχου.

Κεφάλαιο 8

Safe Spark - Συμπεράσματα

8.1 Μελλοντική Εργασία

8.1.1 Έλεγχος Συνθετικού Φορτίου

Χρειάζεται να δημιουργηθεί ένας μηχανισμός που θα εκτελεί ελεγκτικά καθήκοντα βασισμένα σε ένα συνθετικό φορτίο με προκαθορισμένο αποτέλεσμα. Έτσι, η διαδικασία ελέγχου δεν θα προσπαθεί να επιτύχει ένα αποτέλεσμα απόλυτης πλειοψηφίας ή το περισσότερο ψηφισμένο. Θα συγκρίνει τους υπολογισμούς των εκτελεστών με την πραγματική τιμή. Παρόλο που αυτός ο μηχανισμός φαίνεται κατάλληλος για να αντικαταστήσει πλήρως άλλες προσεγγίσεις ελέγχου, έχει σημαντικές αδυναμίες όσον αφορά τη συντήρηση. Επομένως, η συνιστώμενη προσέγγιση θα ήταν, αφού αναπτυχθεί και εφαρμοστεί αυτός ο μηχανισμός, να εκτελείται έλεγχος συνθετικού φορτίου που θα περιλαμβάνει όλους τους κόμβους εκτελεστών με μεταβαλλόμενη συχνότητα.

8.2 Συμπεράσματα

Σε αυτή τη διατριβή, εξερευνήσαμε τη στιβαρότητα και την αποτελεσματικότητα της διαδικασίας ελέγχου εντός ενός Safe Spark cluster. Ο προτεινόμενος μηχανισμός ελέγχου βασίζεται σε μια απλή αλλά ισχυρή στρατηγική για την επαλήθευση της ακεραιότητας των κόμβων εργαζομένων σε ένα δημόσιο περιβάλλον Spark. Ενσωματώνοντας μια περιεκτική διαδικασία ελέγχου, όπου ο αριθμός των ελεγκτικών καθηκόντων σχεδόν ισούται με τον αριθμό των κόμβων εκτελεστών, το σύστημα επιτυγχάνει υψηλό επίπεδο αξιοπιστίας με το κόστος αυξημένων πόρων.

Η αξιολόγηση αποκάλυψε ότι, ενώ μια ελάχιστη προσέγγιση ελέγχου μπορεί να είναι αποδοτική ως προς τους πόρους, είναι σημαντικά λιγότερο στιβαρή και ευάλωτη σε κακόβουλες δραστηριότητες αν οι κακόβουλοι κόμβοι σχηματίσουν μια σημαντική μειοψηφία. Αντίθετα, ένας περιεκτικός έλεγχος, αν και απαιτεί πολλούς πόρους, εξασφαλίζει έναν στιβαρό μηχανισμό κατά των δόλιων υπολογισμών επαληθεύοντας προσεκτικά κάθε κόμβο. Αυτό το μοντέλο περιεκτικού ελέγχου δείχνει τη δυνατότητα να εντοπίσει και να εξαλείψει αποτελεσματικά τους κακόβουλους κόμβους, ενισχύοντας έτσι το πλαίσιο ασφαλείας του cluster.

Η μελλοντική εργασία θα πρέπει να αντιμετωπίσει την πρόκληση των συνεργατικών κακόβουλων κόμβων, οι οποίοι αποτελούν σημαντική απειλή για τη διαδικασία ελέγχου. Η ανάπτυξη ενός προχωρημένου μηχανισμού που θα χρησιμοποιεί συνθετικά φορτία με προκα-

θορισμένα αποτελέσματα θα μπορούσε να βελτιώσει την ακρίβεια του ελέγχου συγκρίνοντας τους υπολογισμούς με γνωστά αποτελέσματα, αντί να βασίζεται στη διαδικασία ψηφοφορίας της πλειοψηφίας. Αυτή η μέθοδος, αν και ελπιδοφόρα, απαιτεί προσεκτική εξέταση των σύνθετων ζητημάτων συντήρησης και κατανομής πόρων.

Συμπερασματικά, η διατριβή αναδεικνύει την ευαίσθητη ισορροπία μεταξύ της επίτευξης στιβαρής ασφάλειας και της διατήρησης των επιδόσεων σε ένα κατανεμημένο περιβάλλον υπολογιστών όπως το Apache Spark. Τα ευρήματα υπογραμμίζουν τη σημασία μιας περιεκτικής στρατηγικής ελέγχου για την προστασία από κακόβουλες δραστηριότητες, εξασφαλίζοντας έτσι την αξιοπιστία και την ασφάλεια των υπολογιστικών διαδικασιών εντός του cluster.

Βιβλιογραφία

[1] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker και Ion Stoica. *Spark: Cluster computing with working sets*. *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, σελίδες 10–10. USENIX Association, 2010.

[2] Holden Karau, Andy Konwinski, Patrick Wendell και Matei Zaharia. *Learning Spark: Lightning-fast big data analysis*. O'Reilly Media, Inc., 2015.

[3] Bill Chambers και Matei Zaharia. *Spark: The Definitive Guide*. O'Reilly Media, Inc., 2018.