Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Πειραματική αξιολόγηση προσεγγιστικών βάσεων δεδομένων

Διπλωματικη Εργασια

της

ΕΙΡΗΝΗΣ ΜΙΧΕΛΑΚΑΚΗ

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εργαστηριο Υπολογιστικων Συστηματων
Αθήνα, Φεβρουάριος 2018

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Πειραματική αξιολόγηση προσεγγιστικών βάσεων δεδομένων

Διπλωματικη Εργασια

της

ΕΙΡΗΝΗΣ ΜΙΧΕΛΑΚΑΚΗ

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 28η Μαρτίου 2018.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Νεκτάριος Κοζύρης	Δημήτριος Τσουμάκος	Βασιλική Καντερέ
Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ιονίου Πανεπιστημίου	Επίκουρη Καθηγήτρια Ε.Μ.Π

Αθήνα, Φεβρουάριος 2018

(Υπογραφή)

.....

Ειρηνη Μιχελακακη

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

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Περίληψη

Η μεγάλη αύξηση των παραγόμενων δεδομένων την περασμένη δεκαετία δημιούργησε πρωτοφανείς ευκαιρίες τόσο στην ανάπτυξη ερευνητικών ιδεών όσο και στη λήψη αποφάσεων στον επιχειρηματικό κόσμο. Τελευταία δίνεται ακόμα μεγαλύτερη έμφαση στη διαδραστική αλληλεπίδραση με μεγάλους όγκους δεδομένων. Δοθέντος ενός συνόλου δεδομένων, ο αναλυτής που καλείται να τα επεξεργαστεί, θα πρέπει ιδανικά να είναι σε θέση να εξάγει πληροφορία από αυτά εντός μερικών milliseconds ή λίγων δευτερολέπτων. Έχει παρατηρηθεί ότι όταν το σύνολο των δεδομένων είναι αρκετά μεγάλο, η κλιμάκωση της αρχιτεκτονικής και του hardware δεν είναι ικανή από μόνη της για να ικανοποιήσει αυτόν τον περιορισμό. Στο πλαίσιο αυτό έχουν αναπτυχθεί τεχνικές που βασίζονται στην προσεγγιστική επεξεργασία των ερωτημάτων. Τα συστήματα προσεγγιστικής επεξεργασίας επιτυγχάνουν υψηλή διαδραστικότητα θυσιάζοντας την ακρίβεια του αποτελέσματος που επιστρέφουν. Στην διπλωματική αυτή, μελετάμε τη συμπεριφορά τριών διαφορετικών συστημάτων προσεγγιστικής επεξεργασίας που βασίζονται σε αλγορίθμους δειγματοληψίας. Τα συστήματα αξιολογούνται (i) ως προς τον χρόνο απόκρισης σε απλά συναθροιστικά ερωτήματα και (ii) ως προς την ακρίβεια που επιτυγχάνουν κατά την προσέγγιση του αποτελέσματος. Τα πειράματά μας έδειξαν πως πράγματι, για ορισμένες κατανομές δεδομένων ή όταν υπάρχει πρότερη γνώση της κατανομής των ερωτημάτων, τα συστήματα αυτά είναι ικανά να παρέχουν σχεδόν ακριβείς απαντήσεις ενώ παράλληλα διατηρούν διαδραστική απόκριση.

Λέξεις Κλειδιά

Προσεγγιστική επεξεργασία ερωτημάτων, βάσεις δεδομένων, κατανεμημένα συστήματα, διαδραστική αναλυτική επεξεργασία, δειγματοληψία, διαδραστική συνάθροιση, διερεύνηση δεδομένων.

Abstract

The data deluge of the last decade created new opportunities for both the academic and the business world. New ideas and techniques were flourished and decision making adopted a data-oriented fashion. As these decisions should often be taken in real-time, nowadays, more emphasis is put on the interactive processing of large data volumes. Ideally, an analyst should be able to extract information from a dataset within a few seconds. Satisfying this constraint by scaling only the hardware falls short when large datasets are the case. To this end, approximate query processing proves to be a very useful technique. In approximate systems, result quality is traded for interactive responses. In this diploma thesis, we study the behavior of three sampling-based approximate query processing systems. Our evaluation is carried out with respect to: (i) the response time in simple aggregate SQL queries and (ii) the achieved accuracy of the approximate answers. Our experiments show that these systems are highly favored by some data distributions or in the case where workload is known a-priori. In these two cases, interactive yet accurate results can be provided for arbitrary dataset sizes.

Keywords

Approximate query processing, databases, distributed systems, online analytical processing, sampling, online aggregation, data exploration.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Νεκτάριο Κοζύρη για την ευκαιρία που μου έδωσε, ώστε να υλοποιήσω την παρούσα μελέτη στο Εργαστήριο Υπολογιστικών Συστημάτων.

Επίσης, ευχαριστώ ιδιαίτερα τον υποψήφιο διδάκτορα κ. Γιάγκο Μυτιλήνη για τη βοήθεια που μου προσέφερε, σε όλα τα στάδια της διπλωματικής μου εργασίας. Η καθοδήγησή του ήταν πολύτιμη.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και όλους τους κοντινούς μου ανθρώπους που μου στάθηκαν καθόλη τη διάρκεια της παρούσας εργασίας, καθώς και της υπόλοιπης φοιτητικής μου πορείας.

Περιεχόμενα

Κατάλογος Σχημάτων

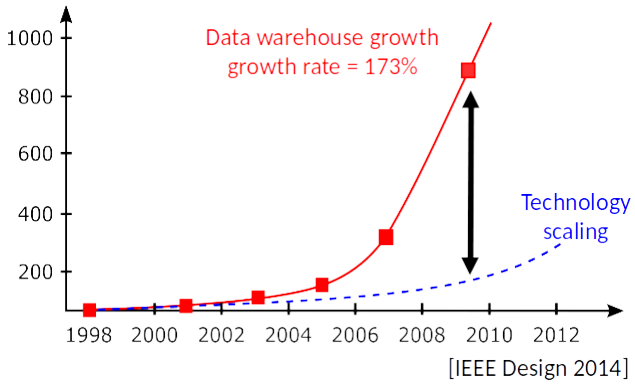
Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Τα τελευταία έτη έχει παρατηρηθεί μια ραγδαία αύξηση των δεδομένων. Τα κοινωνικά δίκτυα, το Internet of Things, οι επιστημονικές προσομοιώσεις κ.α. παράγουν καθημερινά TB δεδομένων. Επίσης, στον τομέα των επιχειρήσεων, η λήψη αποφάσεων βασίζεται όλο και περισσότερο στην συλλογή κι ανάλυση δεδομένων. Για παράδειγμα, ένας αναλυτής σε μια εμπορική επιχείρηση χρειάζεται να επεξεργαστεί τα δεδομένα των πωλήσεων για να εξάγει στατιστικά σε διάφορες διαστάσεις, όπως ανά προιόν, ανά γεωγραφική περιοχή, κλπ.

Σε τέτοιες εφαρμογές, όπου διερευνώνται τα ποιοτικά και ποσοτικά χαρακτηριστικά ενός συνόλου δεδομένων (data exploration) είναι ιδιαίτερα σημαντικό οι απαντήσεις από το σύστημα διαχείρισης των δεδομένων να παρέχονται σε πραγματικό χρόνο. Μια πρώτη προσέγγιση στο πρόβλημα αυτό ήταν η κλιμάκωση της υποδομής (scale out). Κατά την προσέγγιση αυτή, τα δεδομένα διαμοιράζονται σε περισσότερους του ενός κόμβους. Δυστυχώς όμως, ο ρυθμός αύξησης των δεδομένων έχει παρατηρηθεί ότι είναι σημαντικά μεγαλύτερος από τον ρυθμό με τον οποίο έχουμε τη δυνατότητα να αυξάνουμε την υπολογιστική υποδομή μας.



Σχήμα 1.1: Η κλιμακωσιμότητα της τεχνολογίας και των δεδομένων

Κατά αυτόν τον τρόπο, η ανάγκη για αποτελεσματική αποθήκευση, δεικτοδότηση κι επεξεργασία δεδομένων θέτει νέες προκλήσεις στον σχεδιασμό των συστημάτων διαχείρισης δεδομένων. Ένα ερώτημα που τίθεται είναι "Κατά πόσο μπορούμε να απαλλαγούμε από την ανάγκη για νέους πόρους και αύξηση της υποδομής, αξιοποιώντας το γεγονός ότι σε διερευνη-

τικά ερωτήματα (exploratory queries) μπορούμε να ανεχθούμε μικρά σφάλματα". Πράγματι, για κάποιες κατηγορίες ερωτημάτων, ο παραπάνω ισχυρισμός ισχύει και μπορεί να επιφέρει μεγάλες βελτιώσεις στον τρόπο που αναλύουμε τα δεδομένα. Τα συστήματα που λειτουργούν με αυτόν τον τρόπο καλούνται Προσεγγιστικά Συστήματα Επεξεργασίας και επιτυγχάνουν να δίνουν απαντήσεις σε πραγματικό χρόνο, θυσιάζοντας μέρος της ακρίβειας του αποτελέσματος.

Μία δημοφιλής μέθοδος προσεγγιστικής επεξεργασίας αποτελεί η δημιουργία δειγμάτων από τα δεδομένα και η επιλογή ενός εξ΄ αυτών κατά την εκτέλεση του προσεγγιστικού ερωτήματος. Με αυτόν τον τρόπο, αντί να εκτελείται το ερώτημα στο πιθανώς μεγάλο αρχικό σύνολο δεδομένων, εκτελείται σε ένα μικρό δείγμα, το οποίο χωράει στη μνήμη και κατά αυτόν τον τρόπο επιτρέπει την γρήγορη προσπέλαση και επεξεργασία του. Οπότε, το ερώτημα πλέον είναι: α) ποιός αλγόριθμός δειγματοληψίας πρέπει να χρησιμοποιηθεί και τι δείγματα χρειάζεται να κατασκευαστούν και β) στην περίπτωση που υπάρχουν πολλά διαθέσιμα δείγματα, με ποια λογική επιλέγουμε το καταλληλότερο κατά τον χρόνο εκτέλεσης.

Στην παρούσα εργασία γίνεται μελέτη της συμπεριφοράς τριών συστημάτων προσεγγιστικής επεξεργασίας τα οποία βασίζονται σε δειγματοληψία. Τα συστήματα αυτά είναι τα εξής:

- BlinkDB
- SnappyData
- XDB

Πιο συγκεκριμένα μελετάμε τον τρόπο κατασκευής δειγμάτων, το χρόνο απόκρισης και το σφάλμα σε απλά συναθροιστικά SQL ερωτήματα. Λέγοντας "απλά" εννοούμε SUM και COUNT ερωτήματα πάνω σε έναν πίνακα δεδομένων. Τα ερωτήματα αυτά θα μπορούν επίσης να υποστηρίξουν conjunctive φίλτρα και ομαδοποίηση (group by) ως προς μία ή περισσότερες κολώνες. Τα υποστηριζόμενα φίλτρα είναι της μορφής "WHERE p1 and p2 and ..", όπου για κάθε pi θεωρούμε: $pi : a < xi < b$.

Η οργάνωση του κειμένου γίνεται ως εξής: Στο Κεφάλαιο 2 παρουσιάζουμε την λειτουργία της BlinkDB. Αντίστοιχα, στα Κεφάλαια 3 και 4, παρουσιάζουμε τις SnappyData και XDB. Στο Κεφάλαιο 5 παρουσιάζονται οι πειραματικές μέθοδοι που χρησιμοποιήθηκαν για την αξιολόγηση και τη σύγκριση των παραπάνω συστημάτων. Τέλος, στο Κεφάλαιο 6 παρουσιάζουμε συγκεντρωμένα και σχολιάζουμε τα αποτελέσματα των πειραμάτων.

Κεφάλαιο 2

BlinkDB

Εισαγωγή

Η BlinkDB αποτελεί ένα κατανεμημένο σύστημα προσεγγιστικής επεξεργασίας ερωτημάτων (queries) που βασίζεται στην δημιουργία δειγμάτων. Το σύστημα είναι διαδραστικό και ο χρήστης έχει τη δυνατότητα να τρέξει συναθροιστικά SQL ερωτήματα, τα οποία εκτελούνται σχεδόν σε πραγματικό χρόνο. Για να επιτευχθεί αυτό, ο χρήστης μπορεί να παραμετροποιήσει το ερώτημα θέτοντας είτε τον επιθυμητό χρόνο απόκρισης είτε το μέγιστο σφάλμα και την εμπιστοσύνη σε αυτό. Σε αντίθεση με άλλες λύσεις που είχαν προταθεί για την προσεγγιστική επεξεργασία ερωτημάτων, για παράδειγμα [6], η BlinkDB δεν βάζει περιορισμούς για τις τιμές των WHERE, GROUP BY και HAVING δηλώσεων. Η μοναδική υπόθεση που γίνεται είναι ότι οι στήλες των πινάκων που εμφανίζονται στις παραπάνω εντολές είναι χρονικά αμετάβλητες.

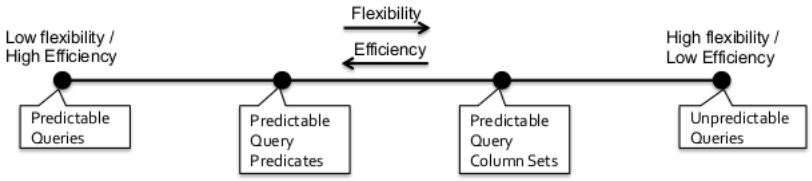
Ερωτήματα που υποστηρίζει το σύστημα

Η BlinkDB μπορεί να δώσει προσεγγιστικά αποτελέσματα για ερωτήματα όπως SUM, COUNT, AVG και QUANTILE. Ωστόσο, arbitrary joins και εμφωλευμένα SQL ερωτήματα δεν υποστηρίζονται. Προσεγγίσεις όμως μπορούν να δωθούν για ενώσεις (joins) όπου στο μεγαλύτερο πίνακα έχει πραγματοποιηθεί δειγματοληψία και ο δεύτερος μικρότερος πίνακας χωράει στη μνήμη cluster.

Στα συστήματα που βασίζονται σε προσεγγιστική επεξεργασία, η πρότερη γνώση του φόρτου εργασίας (workload) μπορεί να βελτιώσει αρκετά την ποιότητα των αποτελεσμάτων. Με βάση το κατά πόσο είναι προβλεψιμο το workload, μπορούμε να ταξινομήσουμε τα ερωτήματα σε τέσσερεις κατηγορίες, όπως φαίνεται στο Σχήμα 2.1.

Από τη μία μεριά, ορισμένα συστήματα υποθέτουν ακριβή γνώση των μελλοντικών ερωτημάτων (Predictable Queries) και χρησιμοποιούν ειδικές δομές δεδομένων για αυτά. Η λειτουργία αυτών των συστημάτων προσφέρει πολύ γρήγορες απαντήσεις αλλά περιορίζεται στα συγκεκριμένα ερωτήματα.

Μία λίγο πιο γενικευμένη υπόθεση αποτελούν τα ερωτήματα με προβλέψιμα κατηγορήματα



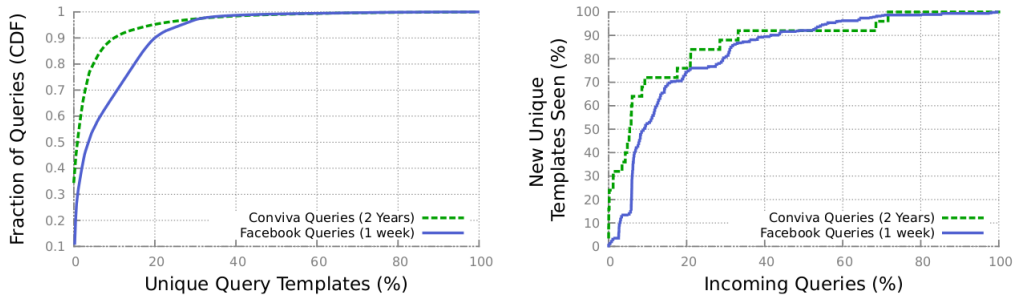
Σχήμα 2.1: Κατηγορίες ερωτημάτων (Πηγή: [1])

(Predictable Query Predicates). Εδώ τα συστήματα υποθέτουν ότι τόσο οι στήλες όσο και οι παράμετροι των WHERE, GROUP BY και HAVING δηλώσεων, παραμένουν σταθερές στο χρόνο. Για παράδειγμα, έαν η δήλωση 'WHERE age <18' εμφανίστηκε μόνη της στο 20% των ερωτημάτων που ερωτήθηκαν, το ίδιο ποσοστό των ερωτημάτων που θα εκτελεστούν θα έχει μόνο τη δήλωση 'WHERE age <18". Με αυτό τον τρόπο, υπάρχοντα συστήματα χρησιμοποιούν τα παλαιότερα ερωτήματα για να προβλέψουν τις ακριβείς γραμμές που θα ζητηθούν στο μέλλον και για να δημιουργήσουν δείγματα που θα τις εμπεριέχουν.

Μία ακόμα κατηγορία workload αποτελούν οι προβλέψιμες ομάδες στηλων που εμφανίζονται στα ερωτήματα (Predictable Query Column Sets). Εδώ θεωρείται γνωστή η συχνότητα εμφάνισης των στηλών στα ερωτήματα αλλά δεν μπορούμε να προβλέψουμε τις ακριβείς τιμές των WHERE, GROUP BY και HAVING δηλώσεων. Οι στήλες που χρησιμοποιούνται για φιλτράρισμα και ομαδοποιήση σε ένα ερώτημα αναφέρονται ως Query Column Set (QCS). Αντίστοιχα με το προηγούμενο παράδειγμα, εδώ γνωρίζουμε ότι το 20% των ερωτημάτων χρησιμοποίησαν τη στήλη age για φιλτράρισμα ή ομαδοποίηση, και προβλέπουμε ότι και το 20% θα αναφέρονται στην ίδια στήλη. Αυτή η μέθοδος δείχνει ποίες στήλες είναι χρήσιμο να έχουν δείκτες για τη βελτιστοποίηση της πρόσβασης στα δεδομένα.

Το πιο γενικό μοντέλο αποτελούν τα απρόβλεπτα ερωτήματα (Unpredictable Queries). Καμία πρόβλεψη δεν μπορεί να γίνει για τα μελλοντικά ερωτήματα και τα συστήματα βασίζονται στους βελτιστοποιητές (optimizers) των ερωτημάτων για καλύτερα αποτελέσματα. Η δημιουργία στοχευμένων δειγμάτων εδώ δεν είναι δυνατή και η δειγματοληψία είναι ομοιόμορφη (uniform). Παρότι αυτό το μοντέλο αποτελεί και το πιο γενικό, δεν επιτρέπει σε ένα σύστημα προσεγγιστικής επεξεργασίας δεδομένων να φτιάξει αποτελεσματικά δείγματα. Επιπλέον, ο συγχρονισμός που απαιτείται για την τυχαία πρόσβαση γραμμών, και η επιβάρυνση λόγω επικοινωνίας, καθιστούν ένα μεγάλο cluster ακατάλληλο περιβάλλον για την εκτέλεση τέτοιων ερωτημάτων.

Εξαιτίας του μειονεκτήματος του τελευταίου, η BlinkDB χρησιμοποιεί το μοντέλο Predictable Query Column Sets για να πάρει πληροφορίες και να δημιουργήσει αποτελεσματικά δείγματα για τα μελλοντικά ερωτήματα. Επίσης, έχει παρατηρηθεί ότι σε πραγματικά ερωτήματα ένα μικρό ποσοστό των QCS καλύπτει την πλειονότητα των ερωτημάτων [1]. Συγκεκριμένα διεξήχθηκαν μετρήσεις πάνω σε δεδομένα από το Facebook και το Conviva τα αποτελέσματα των οποίων φαίνονται στο Σχήμα 2.2. Αριστερά αναπαρίσταται η κατανομή των QCS ως προς τα παλιά ερωτήματα και βλέπουμε ότι με ένα μικρό ποσοστό των QCS, της τάξης του 20-30%, καλύπτεται πάνω από το 90% των ερωτημάτων. Για όλα τα ερωτήματα χρειάζονται 182 και



Σχήμα 2.2: Η σχέση μεταξύ των ερωτημάτων και QCS (Πηγή: [1])

455 μοναδικά QCS για το Conviva και το Facebook αντίστοιχα, ενώ αν αγνοήσουμε τα QCS που εμφανίζονται σε λιγότερα από 10 ερωτήματα, καταλήγουμε με 102 και 211 QCS που καλύπτουν 17.437 και 68.785 ερωτήματα αντίστοιχα. Στο δεξί γράφημα φαίνεται το ποσοστό των μοναδικών QCS, που ανιχνεύθηκαν πριν, συγκριτικά με νέα ερωτήματα που επεξεργάζονται τα συστήματα. Είναι εμφανές ότι περίπου στο 10% των ερωτημάτων έχει εμφανιστεί το 60% των QCS που δεν έχουν παρατηρηθεί στο παρελθόν, και με λίγο πάνω από 30% των ερωτημάτων, έχουμε δει σχεδόν το 100% των καινούργιων QCS. Αυτό αποδεικνύει ότι τα QCS είναι σταθερά στο χρόνο και σε πραγματικά workloads.

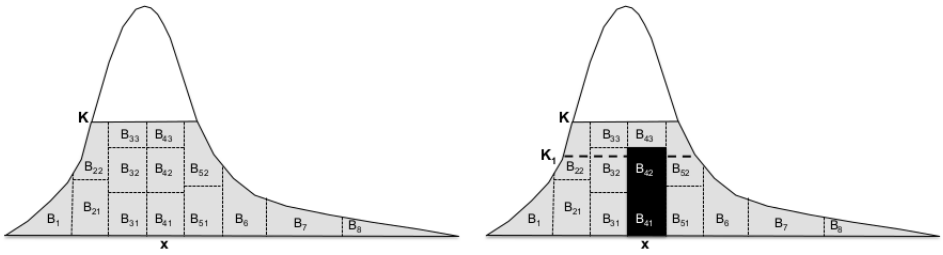
Προσεγγιστική Εκτέλεση

Για να πετύχει μικρούς χρόνους εκτέλεσης και τη μεγαλύτερη δυνατή ακρίβεια η BlinkDB χρησιμοποιεί έναν αλγόριθμο για τη δημιουργία δειγμάτων και έναν για την επιλογή ενός εξ΄ αυτών, ανάλογα με τις απαιτήσεις του ερωτήματος.

Δημιουργία των Δειγμάτων

Η δημιουργία δειγμάτων αποτελεί ένα πρόβλημα βελτιστοποίησης. Δεδομένου του συνόλου των στηλών που εμφανίστηκαν σε προηγούμενα ερωτήματα και των συχνοτήτων εμφάνισής τους, πρέπει να επιλέξουμε ένα σύνολο δειγμάτων τέτοιο ώστε να αναπαραστήσουμε όσο καλύτερα γίνεται την κατανομή των δεδομένων και παράλληλα να μην υπερβούμε τον διαθέσιμο χώρο αποθήκευσης. Για την αποτελεσματική λειτουργία του συστήματος, τα δείγματα δημιουργούνται με στόχο να απαντούν αποτελεσματικά σε ερωτήματα που αναφέρονται σε QCSs που εχουν εμφανιστεί στο παρελθόν και να παρέχουν ικανοποιητική κάλυψη για μελλοντικά ερωτήματα με παρόμοιες ομάδες στηλών.

Σε περιπτώσεις όπου ένα ερώτημα δεν ομαδοποιεί ή φιλτράρει τα δεδομένα, ένα ομοιόμορφο δείγμα δίνει ικανοποιητικά αποτελέσματα. Όμως, στην περίπτωση φιλτραρίσματος οι απαντήσεις που δίνει αυτό το είδος δείγματος μπορεί να απέχουν αρκετά από τις πραγματικές. Ειδικά στις περιπτώσεις όπου οι στήλες περιέχουν σπάνιες τιμές, αυτές μπορεί να μην



Σχήμα 2.3: Παράδειγμα αποθήκευσης δειγμάτων (Πηγή: [1])

αναπαρίστανται από ένα ομοιόμορφο δείγμα. Η λύση αυτού του προβλήματος είναι η χρήση στρωματοποιημένων (stratified) δειγμάτων. Αυτά λαμβάνονται από τις στήλες που εμφανίζονται πιο συχνά στα παρελθοντικά ερωτήματα. Εξαιτίας της στρωματοποιημένης δειγματοληψίας, σπάνιες τιμές αναπαριστώνται καλύτερα από ότι σε ένα ομοιόμορφο δείγμα και επομένως δίνεται η ακριβέστερη απάντηση για οποιεσδήποτε τιμές ανεξαρτήτως της κατανομής τους στο συνολικό όγκο δεδομένων.

Ας θεωρήσουμε ένα ερώτημα Q πάνω σε ένα πίνακα T και ένα QCS ϕ. Επιπλέον, έστω ότι ο χρήστης δίνει ένα χρονικό όριο t ή ένα όριο σφάλματος e. Το χρονικό όριο t αντιστοιχεί στο μέγιστο μέγεθος δείγματος n που μπορεί να επεξεργαστεί σε αυτό το διάστημα. Όσο μεγαλύτερο το δείγμα που χρησιμοποιείται, τόσο πιο ακριβή είναι τα αποτελέσματα που δίνει. Άρα το δείγμα μεγέθους n αποτελεί το βέλτιστο δείγμα για το χρονικό περιορισμό t. Έπισης, στην περίπτωση του ορίου στο σφάλμα, το δείγμα μεγέθους n είναι αυτό που ικανοποιεί τον περιορισμό στο μικρότερο δυνατό χρόνο.

Στον ολικό πίνακα T υπάρχουν ομάδες διαφορετικών τιμών x για κάθε QCS ϕ, και κάθε ομάδα τιμών εμφανίζεται σε κάποιες γραμμές T_x. Το ερώτημα πρέπει να υπολογιστεί σε αυτές τις γραμμές. Επειδή η εκτέλεση σε όλα τα δεδομένα απαιτεί πολύ χρόνο, πραγματοποιείται πάνω σε ένα δείγμα S με n γραμμές, το οποίο αποτελεί την ένωση όλων των δειγμάτων S_x που αντιστοιχούν στις ομάδες x. Το σφάλμα του ερωτήματος εξαρτάται από τα μεγέθη του συνολικού δείγματος και των επιμέρους δειγμάτων. Λόγω περιορισμών στο χώρο, χρησιμοποιείται σαν ανώτατο όριο μεγέθους κάθε υπόδειγματος S_x ένας αριθμός K.

Ένα σημαντικό πλεονέκτημα αυτής της στρατηγικής δειγματοληψίας είναι ότι ερωτήματα με το ίδιο QCS ϕ αλλά διαφορετικούς περιορισμούς t ή e, χρησιμοποιούν την ίδια οικογένεια δειγμάτων για τη εκτίμηση προσεγγιστικών απαντήσεων. Η αποθήκευση των δειγμάτων γίνεται όπως φαίνεται στην Εικόνα 2.3. Σε κάθε block αποθηκεύεται το δείγμα S_x και όπως φαίνεται στο Σχήμα, σε περίπτωση που αυτό είναι μεγάλο, καταλαμβάνει περισσότερα από ένα blocks. Όταν ένα ερώτημα χρειάζεται να διαβάσει n γραμμές από το S, επειδή οι τιμές έχουν αποθηκευτεί τυχαία στα blocks, αρκεί να διαβάσει τις n_x γραμμές από κάθε μικρότερο δείγμα.

Σε αντίθεση με προηγούμενα συστήματα, για παράδειγμα τα [7], [8], η BlinkDB δημιουργεί πολυδιάστατα στρωματοποιημένα δείγματα. Επειδή όμως, όπως αναφέραμε, ο χώρος αποθήκευσης είναι περιορισμένος και το πλήθος όλων των πιθανών δειγμάτων εξαρτάται εκθετικά από τον αριθμό των στηλών, πρέπει να επιλέγονται συγκεκριμένα πεδία στηλών, από τα οποία θα φτίαχνονται τα δείγματα. Για αυτή την απόφαση λαμβάνονται υπόψη η διασπορά των

δεδομένων, δηλαδή το πλήθος των ξεχωριστών ομάδων τιμών, η πιθανότητα εμφάνισης των QCS, η οποία εκτιμάται από τα προηγούμενα ερωτήματα, και το κόστος αποθήκευσης για κάθε δείγμα.

Τα στρωματοποιημένα δείγματα εισάγουν πόλωση (bias) στο τελικό αποτέλεσμα, καθώς μικρές ομάδες τιμών εμφανίζονται σε μεγαλύτερη αναλογία και μεγάλες ομάδες τιμών εμφανίζονται λιγότερο στο δείγμα. Για τη διόρθωση αυτού του σφάλματος, η BlinkDB καταγράφει τον πραγματικό ρυθμό δειγματοληψίας κάθε ομάδας τιμών σε μία κρυφή στήλη στο σχήμα (schema) κάθε δείγματος.

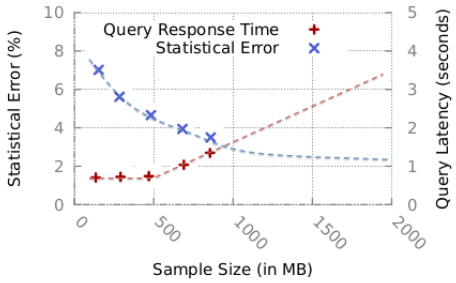
Πέρα από τη δημιουργία τους, η BlinkDB διατηρεί τα δείγματα που χρησιμοποιούνται από όλα τα ερωτήματα, σε αντίθεση με άλλα συστήματα που φτιάχνουν καινούργια για κάθε ερώτημα που εκτελείται. Εξαιτίας αυτού, στην περίπτωση όπου ένα δείγμα έχει μη αναμενόμενες τιμές σε κάποια στήλη, όλα τα ερωτήματα που χρησιμοποιούν το συγκεκριμένο δείγμα θα έχουν κάποια απόκλιση στα αποτελέσματά τους. Ωστόσο, η δημιουργία δειγμάτων εκ των προτέρων αποτελεί σημαντικό στάδιο για τη βελτιωμένη λειτουργία της BlinkDB και η αντιμετώπιση αυτού του προβλήματος γίνεται μέσα από την περιοδική αντικατάσταση των δειγμάτων. Συγκεκριμένα, μια εφαρμογή χαμηλής προτεραιότητας τρέχει στο background και ανανεώνει καθημερινά τα δείγματα.

Εάν τα ερωτήματα που εκτελούνται, αλλάξουν με το χρόνο από τα προηγούμενα, τα δείγματα που έχουν φτιαχτεί μπορεί να μην είναι πλέον τα κατάλληλα. Η BlinkDB καταγράφει στατιστικές τιμές των συνολικών δεδομένων και τρέχει τον αλγόριθμο της δημιουργίας δειγμάτων εάν προκύψει η ανάγκη αλλαγής των υπάρχοντων. Για να περιοριστεί ο χρόνος που χρησιμοποιείται γι΄ αυτό, μπορεί να οριστεί σαν παράμετρος το συνολικό ποσοστό των δειγμάτων που επιτρέπεται να δημιουργείται από την αρχή.

Χρήση Δειγμάτων κατά την Εκτέλεση

Η επιλογή ενός δείγματος για την εκτέλεση του ερωτήματος βασίζεται στους περιορισμούς σφάλματος ή χρόνου απόκρισης που έχει θέσει ο χρήστης. Η BlinkDB τρέχει το ερώτημα σε πολλά μικρά μέρη των δειγμάτων για να εξάγει συμπεράσματα για την επιλεκτικότητα του (selectivity) και για να διαλέξει το καταλληλότερο δείγμα, έτσι ώστε να πληροί τις προδιαγραφές σφάλματος ή χρόνου.

Κατά τη διάρκεια εκτέλεσης, σκόπος είναι να βρεθεί το καταλληλότερο δείγμα. Η επιλογή του δείγματος εξαρτάται από τις στήλες που εμφανίζονται στο ερώτημα, τις παραμέτρους των WHERE και GROUP BY δηλώσεων, τη θέση των δεδομένων και την κατανομή τους στο cluster, δηλαδή αν βρίσκονται στο δίσκο ή στη μνήμη. Αν υπάρχουν παραπάνω από ένα δείγματα πάνω σε ένα σύνολο στηλών, το οποίο περιέχει και τις στήλες που εμφανίζονται στο ερώτημα, τότε επιλέγεται το δείγμα με το μικρότερο αριθμό στηλών. Αν κανένα δείγμα δεν περιέχει όλες τις στήλες, τότε εκτελείται παράλληλα σε μικρά μέρη όλων των δειγμάτων που βρίσκονται στη μνήμη του cluster εκείνη τη στιγμή. Από τα αποτελέσματα ορίζεται η επιλεκτικότητα του δείγματος ως η αναλογία των γραμμών που επιλέχθηκαν προς τις γραμμές που



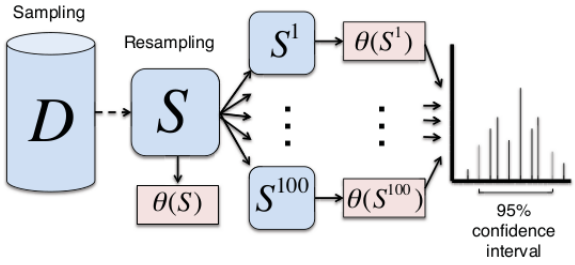
Σχήμα 2.4: Παράδειγμα ELP (Πηγή: [1])

διαβάστηκαν. Έπειτα επιλέγεται το δείγμα με την υψηλότερη επιλεκτικότητα, γιατί από τη μία, όσο μικρότερος ο χρόνος που παραχωρεί ο χρήστης, τόσο λιγότερες γραμμές θα διαβαστούν και μεγαλύτερη επιλεκτικότητα θα έχει το ερώτημα, ενώ από την άλλη μικρότερο σφάλμα επιτυγχάνεται με περισσότερες επιλεγόμενες γραμμές και επομένως υψηλότερη επιλεκτικότητα.

Έπειτα, πρέπει να επιλεγεί το κομμάτι του δείγματος πάνω στο οποίο θα τρέξει το ερώτημα ανάλογα με τους περιορισμούς. Για αυτό το στάδιο δημιουργείται το προφίλ σφάλματος-χρόνου απόκρισης (Error-Latency Profile - ELP) εκτελώντας το ερώτημα σε μικρά δείγματα και παίρνοντας πληροφορίες για το ρυθμό με τον οποίο μειώνεται το σφάλμα και αυξάνεται ο χρόνος απόκρισης σε σχέση με το μέγεθος των δειγμάτων. Το Error Profile χρησιμοποιείται για όλα τα ερωτήματα με περιορισμούς στο σφάλμα με σκοπό την εκτίμηση του μικρότερου μεγέθους δείγματος που ικανοποιεί αυτόν τον περιορισμό e. Η διακύμανση και τα διαστήματα εμπιστοσύνης υπολογίζονται με τη βοήθεια της στατιστικής για τα ερωτήματα που υποστηρίζονται, και η πρώτη είναι αντιστρόφως ανάλογη με το πλήθος των επιλεγόμενων γραμμών του δείγματος. Από αυτή την εκτέλεση σε μικρά κομμάτια του δείγματος, η BlinkDB εκτιμάει την επιλεκτικότητα του ερωτήματος, τη διακύμανση του δείγματος για AVG/SUM και την κατανομή των δεδομένων για QUANTILE, και με χρήση μαθηματικών τύπων υπολογίζει το πλήθος γραμμών n που είναι απαραίτητο για να ικανοποιήσει τον περιορισμό.

Το Latency Profile χρησιμοποιείται για όλα τα ερωτήματα με χρονικό περιορισμό και αντίστοιχα εκτιμάει το μέγιστο αριθμό γραμμών n που μπορούν να επεξεργαστούν μέσα σε αυτό το διάστημα. Η τιμή του n εξαρτάται από πολλές παραμέτρους, όπως η φυσική θέση των δεδομένων και οι διαθέσιμοι πόροι του cluster για την εκτέλεση του ερωτήματος. Σαν απλοποίηση των παραπάνω, η BlinkDB υποθέτει γραμμική εξάρτηση ανάμεσα στις γραμμές n και στο χρόνο απόκρισης, όπως έχει παρατηρηθεί σε περιβάλλοντα κατανεμημένης παράλληλης επεξεργασίας για ερωτήματα. Για μικρά δείγματα που βρίσκονται στη μνήμη, η BlinkDB τρέχει μικρότερα δείγματα μέχρι η απόδοση να είναι σχεδόν γραμμική και τότε υπολογίζει τις σταθερές για το μοντέλο πρόβλεψης, όπως ο ρυθμός επεξεργασίας δεδομένων και ο ρυθμός Ε/Ε στο δίσκο.

Στο Σχήμα 2.4 δίνεται ένα παράδειγμα ELP. Ο αριστερός άξονας δείχνει το σφάλμα επί τοις εκατό και ο δεξιά το χρόνο εκτέλεσης του ερωτήματος. Ο οριζόντιος άξονας αναπαριστά το μέγεθος του δείγματος. Βλέπουμε ότι όσο μεγαλύτερο είναι το δείγμα το σφάλμα μειώνεται, επομένως επιτυγχάνεται μεγαλύτερη ακρίβεια, ενώ ο χρόνος απόκρισης του ερωτήματος



Σχήμα 2.5: Μέθοδος εκτίμησης σφάλματος bootstrap (Πηγή: [3])

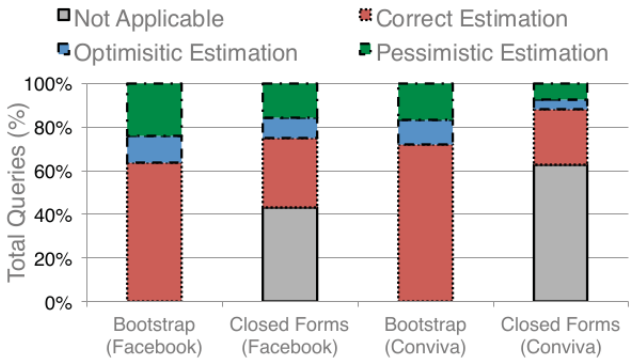
αυξάνεται. Για μέγεθος δείγματος πάνω από 500 MB η σχέση χρόνου και μεγέθους είναι γραμμική, όπως προαναφέρθηκε.

Πρόβλεψη Διαστημάτων Εμπιστοσύνης

Όπως έχει αναφερθεί, η BlinkDB και τα υπόλοιπα συστήματα επεξεργασίας που βασίζονται στα δείγματα (Sampling-based Approximate Query Processing - S-AQP), χρησιμοποιούν μεθόδους της στατιστικής για να προβλέπουν το σφάλμα και να παρέχουν στο χρήστη ένα διάστημα εμπιστοσύνης. Αυτές οι προσεγγίσεις πραγματοποιούνται πάνω στο δείγμα που επιλέχτηκε και βοηθούν το σύστημα να ελέγχει το σφάλμα αλλάζοντας το μέγεθος του δείγματος. Έτσι βρίσκει την ισορροπία μεταξύ της ακρίβειας και της ταχύτητας του αποτελέσματος. Αυτό προϋποθέτει την αξιοπιστία της εκτίμησης του σφάλματος. Η υποτίμηση του σφάλματος δίνει μία λανθασμένη εντύπωση στο χρήστη για την ακρίβεια του αποτελέσματος, κάτι το οποίο μπορεί να επηρεάσει τις αποφάσεις του. Αντίθετα, η υπερτίμηση του σφάλματος οδηγεί το σύστημα στη χρήση ενός μεγαλύτερου δείγματος για την επίτευξη της επιθυμητής ακρίβειας, το οποίο θα χρειαστεί περισσότερο χρόνο επεξεργασίας, ενώ δεν είναι απαραίτητο.

Μη-παραμετρικό Boostrap

Έχουν προταθεί πολλές μέθοδοι για την εκτίμηση των σφαλμάτων. Μία από αυτές αποτελεί το bootstrap. Αν δεν υπήρχαν περιορισμοί στο χρόνο και στην πρόσβαση στα δεδομένα, το διάστημα εμπιστοσύνης μπορεί να προσεγγιστεί παίρνοντας K δείγματα από τα δεδομένα D και εκτελώντας το ερώτημα σε κάθε ένα από αυτά. Η κατανομή των απαντήσεων για πολύ μεγάλο K προσεγγίζει την κατανομή δειγματοληψίας και παρέχει αξιόπιστα αποτελέσματα του πραγματικού διαστήματος εμπιστοσύνης. Ωστόσο, οι συνεχόμενες προσβάσεις στα δεδομένα καθιστούν αυτή τη μέθοδο χρονοβόρα και υπερνικούν τα πλεονεκτήματα της δημιουργίας δειγμάτων. Για την αντιμετώπιση αυτών των μειονεκτημάτων χρησιμοποιείται το μη-παραμετρικό boostrap [38]. Αυτή η μέθοδος εφαρμόζει τα παραπάνω πάνω στο δείγμα S μεγέθους n, το οποίο έχει ήδη δημιουργηθεί για τον υπολογισμό του προσεγγιστικού αποτελέσματατος. Συ-



Σχήμα 2.6: Εκτίμηση των μεθόδων bootstrap και closed-form estimation πάνω σε πραγματικά σύνολα εργασιών (Πηγή: [3])

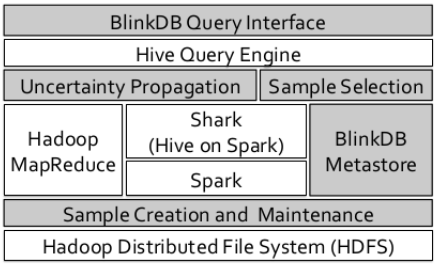
γκεκριμένα, λαμβάνει K δείγματα S^i με επανατοποθέτηση μεγέθους n, από το αρχικό δείγμα S και τρέχει το ερώτημα και σε αυτά. Τα αποτελέσματα που υπολογίζονται από τα S^i ακολουθούν μία κατανομή, από την οποία η μέθοδος εξάγει το προσεγγιστικό διάστημα εμπιστοσύνης που θα εμφανιστεί τελικά στο χρήστη. Το Σχήμα 2.5 αποτελεί μια σχηματική απεικόνιση της μεθόδου.

Ένα μειονέκτημα της μεθόδου είναι ότι πραγματοποιούνται πολλοί υπολογισμοί (K εκτελέσεις του ερωτήματος στα δείγματα S^i) με αποτέλεσμα την αύξηση του χρόνου εκτέλεσης του αλγορίθμου, τόσο ώστε για μεγάλα K να χάνονται τα πλεονεκτήματα της δειγματοληψίας.

Εκτίμηση με χρήση κλειστών τύπων

Άλλη μία μέθοδος που χρησιμοποιείται είναι η εκτίμηση με χρήση κλειστών τύπων (closed-form estimation). Η κατανομή δειγματοληψίας (sampling distribution) προσεγγίζεται από μία κανονική κατανομή σύμφωνα με το Κεντρικό οριακό θεώρημα. Η μέση τιμή της κανονικής κατανομής είναι ίση με το το αποτέλεσμα του ερωτήματος πάνω στο δείγμα S. Για τον υπολογισμό της διακύμανσης χρησιμοποιούνται κατάλληλοι τύποι στατιστικης ανάλογα με το ερώτημα. Σε αντίθεση με το bootstrap που μπορει να υποστηρίξει οποιαδήποτε συνάρτηση χρήστη (user defined function - UDF), η χρήση κλειστών τύπων υποστηρίζει μόνο ερωτήματα με SUM, COUNT, AVG και VARIANCE. Παρόλα αυτα είναι αρκετά πιο αποδοτική απο άποψη χρόνου εκτέλεσης, καθώς υπολογίζει απλά έναν τύπο έναντι των K επαναλήψεων της διαδικασίας του bootstrap.

Η σύγκριση των δύο μεθόδων πάνω σε πραγματικά δεδομένα φαίνεται στο Σχήμα 2.6. Τα δεδομένα είναι μεγέθους δεκάδων terabytes και συλλέχθηκαν από τα Conviva και Facebook. Το workload αποτελείται από εκατοντάδες Apache Hive ερωτήματα. Ο οριζόντιος άξονας έχει στήλες για κάθε μέθοδο στις διαφορετικές περιπτώσεις δεδομένων, ενώ ο κατακόρυφος αναπαριστά το ποσοστό των ερωτημάτων. Η γκρι περιοχή είναι για τα ερωτήματα εκείνα για τα οποία δεν μπόρεσε να γίνει εκτίμηση με χρήση κλειστών τύπων. Το γαλάζιο μέρος του διαγράμματος δίνει το ποσοστό των ερωτημάτων όπου το εκτιμώμενο διάστημα εμπιστοσύνης



Σχήμα 2.7: Η αρχιτεκτονική της BlinkDB (Πηγή: [1])

ήταν αρκετά μικρότερο από το πραγματικό, ενώ το πράσινο μέρος αντίστοιχα δείχνει το ποσοστό όπου το εκτιμώμενο διάστημα ήταν αρκετά μεγαλύτερο. Τέλος, το κόκκινο δείχνει τις σωστές εκτιμήσεις των διαστημάτων εμπιστοσύνης.

Υλοποίηση

Στο Σχήμα 2.7 φαίνονται η αρχιτεκτονική της BlinkDB. Η υλοποιήση της έγινε πάνω από το Hive Query Engine [9] και υποστηρίζει για την εκτέλεση των ερωτημάτων το Hadoop MapReduce [10] και το Spark [11]. Επίσης χρησιμοποιείται το Shark, το οποίο αποτελεί ένα βελτιωμένο πλαίσιο για τη μεταφορά δεδομένων στην μνήμη (cache memory) [2]. Η αποθήκευση των δεδομένων γίνεται στο κατανεμημένο σύστημα αρχείων του Hadoop (Hadoop Distributed Filesystem - HDFS). Για την υποστήριξη ερωτημάτων με όρια στο χρόνο ή στο σφάλμα, επεκτάθηκε η HiveQL και πλέον υπάρχει το BlinkDB Query Interface. Επιπλέον, σε αυτό το επίπεδο ανιχνεύεται η εισαγωγή δεδομένων, η οποία ενεργοποιεί τη δημιουργία και τη συντήρηση των δειγμάτων (Sample Creation and Maintenance). Ο HiveQL parser επεκτάθηκε ώστε να επιλέγει ένα ομοιόμορφο ή στρωματοποιημένο δείγμα κατάλληλου μεγέθους για την εκτέλεση (Sample Selection). Τέλος, προστέθηκε μία μονάδα για τον υπολογισμό διαστημάτων εμπιστοσύνης και σφάλματος μαζί με το αποτέλεσμα (Uncertainty Propagation).

Κεφάλαιο 3

SnappyData

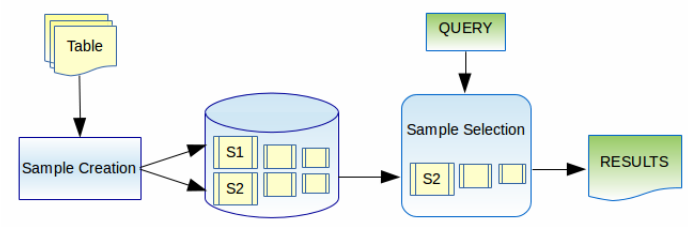
Εισαγωγή

Ο σύγχρονος χώρος εργασίας απαιτεί ετερογενείς αρχιτεκτονικές ώστε να μπορεί να διαχειριστεί αποδοτικά διαφορετικά σενάρια επεξεργασίας όπως: αναλυτικά ερωτήματα (Online Analytical Processing-OLAP), ερωτήματα με συναλλαγές (Online Transactional Processing-OLTP) και ροές δεδομένων (streams). Οι λύσεις που έχουν προταθεί ως τώρα στοχεύουν στην καλύτερη διαχείριση των δεδομένων χωρίς να πληρούν ωστόσο όλες τις παραπάνω απαιτήσεις. Ένας τύπος συστημάτων βασίζεται στο Hadoop, για παράδειγμα τα Hive [9], Impala [26] και Spark SQL [27], τα οποία χρησιμοποιούν βελτιστοποιήσεις για OLAP ερωτήματα και πίνακες-στήλες για να επεξεργαστούν μεγάλους όγκους αμετάβλητων δεδομένων. Υβριδικά συστήματα για επεξεργασία συναλλαγών και αναλυτικών ερωτημάτων (Hybrid Transaction/Analytical Processing - HTAP), όπως για παράδειγμα το MemSQL [28], υποστηρίζουν τόσο συναλλαγές σε πραγματικό χρόνο (OLTP) όσο OLAP ερωτήματα με το να αποθηκεύουν τα δεδομένα και σε πίνακες γραμμές και σε πίνακες στήλες. Ο περιορισμός αυτών των συστημάτων έγκειται στην ανάγκη τους για μια εξωτερική μηχανή επεξεργασίας ροών δεδομένων (streams). Τέλος, συστήματα επεξεργασίας ροών δεδομένων που παρέχουν επεξεργασία συναλλαγών, [29], [30] και [31], δεν υποστηρίζουν πολύπλοκα ερωτήματα πάνω σε αυτές.

Η SnappyData αποτελεί μια πλατφόρμα ανοικτού κώδικα (open source) που καλύπτει όλες τις παραπάνω απαιτήσεις. Για τη διαχείριση OLAP ερωτημάτων και streams χρησιμοποιεί το Apache Spark, ενώ για την παροχή υποστήριξης για transactional workloads κάνει χρήση του συστήματος Apache GemFire [25]. Στην παρούσα εργασία, θα εστιάσουμε στο μηχανισμό που χρησιμοποιεί για να απαντάει αποδοτικά OLAP ερωτήματα.

Προσεγγιστική Εκτέλεση OLAP ερωτημάτων

Όπως θα δείξουμε και στα πειράματά μας, τα OLAP ερωτήματα μπορεί να χρειάζονται αρκετά λεπτά για να ολοκληρωθούν εάν τα δεδομένα ειναι πολλά, εάν είναι απαραίτητη η ανακατανομή των δεδομένων (shuffling) μεταξύ των κόμβων του cluster ή εάν υπάρχουν πολλοί



Σχήμα 3.1: Απλοποιημένη αναπαράσταση της χρήσης δειγμάτων στη SnappyData (Πηγή: [22])

χρήστες συνδεδεμένοι σε αυτό. Για την εξασφάλιση μικρού χρόνου εκτέλεσης κάτω από οποιεσδήποτε συνθήκες, η SnappyData έχει αναπτύξει ένα σύστημα, τη Synopsis Data Engine (SDE), το οποίο χρησιμοποιεί ειδικές δομές και τεχνικές δειγματοληψίας, μειώνοντας έτσι τον όγκο των δεδομένων και παρέχει προσεγγιστικές αλλά γρήγορες απαντήσεις. Για να το επιτύχει αυτό, η SnappyData χρησιμοποιεί στρωματοποιημένα δείγματα για OLAP ερωτήματα και sketches [13] για ερωτήματα σε streams. Το σχήμα 3.1 αποτελεί μία απλοποιημένη αναπαράσταση της SDE και της χρήσης δειγμάτων.

Οι τεχνικές για την εκτίμηση του σφάλματος που χρησιμοποιούνται αναλύονται σε προϋπάρχουσες εργασίες, και βασίζονται σε κλειστους τυπους [3] και σε μια bootstrap τεχνική, που είναι ιδιαίτερα αποδοτική από άποψης χρόνου και λέγεται analytical bootstrap [15]. Τέλος, σε σχέση με τα ερωτήματα, χρησιμοποιεί ένα μοντέλο παρόμοιο με αυτό της BlinkDB, όπου χτίζει δείγματα πάνω σε συγκεκριμένα Query Column Sets (QCS) και βελτιστοποιεί τα ερωτήματα που γίνονται πάνω σε αυτές τις κολώνες.

Δημιουργία και Ενημέρωση των Στατιστικών Δομών

Η ομοιόμορφη δειγματοληψία επιλέγει στοιχεία ενός πίνακα χωρίς να λαμβάνει υπόψη τα χαρακτηριστικά τους. Κάθε γραμμή του πίνακα έχει την ίδια πιθανότητα να επιλεγεί. Αυτό το είδος δειγματοληψίας δεν αποδίδει ικανοποιητικά για επιλεκτικά ερωτήματα πάνω σε συγκεκριμένη περιοχή των δεδομένων. Από την άλλη, η στρωματοποιημένη δειγματοληψία επιτρέπει στο χρήστη να καθορίσει τις στήλες που εμφανίζονται στα ερωτήματα και εξασφαλίζει την ικανοποιητική αναπαράσταση των τιμών τους στο δείγμα. Η SnappyData επιτρέπει τη δημιουργία τους πάνω σε δεδομένα που βρίσκονται είτε στη μνήμη του cluster είτε σε κάποιο εξωτερικό σύστημα αποθήκευσης, όπως το S3 [32], και το HDFS. Εάν τα αρχικά δεδομένα δεν χωράνε στη μνήμη του cluster, η SnappyData υποστηρίζει τη δημιουργία ενός εξωτερικού πίνακα που δείχνει στην πηγή των δεδομένων και τη δειγματοληψία με βάση αυτόν τον εξωτερικό πίνακα.

Για τη δημιουργία στρωματοποιημένων δειγμάτων ο χρήστης ορίζει το QCS και το λόγο δειγματοληψίας ή κλάσμα (fraction). Το QCS αποτελεί το σύνολο των στηλών εκείνων που εμφανίζονται συχνότερα στα ερωτήματα και μπορεί να κατασκευαστεί χρησιμοποιώντας SQL συναρτήσεις. Στην περίπτωση στρωματοποιημένων δειγμάτων, η SnappyData προσαρμόζει το ρυθμό δειγματοληψίας για κάθε ομάδα (stratum) ώστε το συνολικό μέγεθος του δείγμα-

τος να είναι αυτό που ζήτησε ο χρήστης και παράλληλα να έχει καλυφθεί ικανοποιητικά κάθε περιοχή του συνόλου των δεδομένων. Για την επιλογή των στηλών αυτών, όπως και στην περίπτωση της BlinkDB, απαιτείται στατιστική ανάλυση περασμένων ερωτημάτων. Για το σκοπό αυτό, η ομάδα της SnappyData ανέπτυξε το CliffGuard [16], το οποίο είναι ένα εργαλείο που αναλύει αυτόματα τα αρχεία καταγραφής (logs) των παλαιών ερωτημάτων και προσφέρει στατιστικά αποτελέσματα για τις συχνότητες εμφάνισεις των διαφόρων στηλών του πίνακα σε αυτά.

Χρήση των Δειγμάτων κατά την Εκτέλεση

Η SnappyData χρησιμοποιεί μία διαδικασία για την επιλογή του καταλληλότερου δείγματος κατά τη διάρκεια εκτέλεσης. Αρχικά, η προσεγγιστική επεξεργασία είναι δυνατή μόνο όταν το query βασίζεται στα COUNT, SUM και AVG. Σε αντίθετη περίπτωση το query υπολογίζεται πάνω σε ολόκληρο τον πίνακα. Εάν οι στήλες του δείγματος, πάνω στις οποίες έγινε στρωματοποιημένη δειγματοληψία, είναι ίδιες με αυτές του ερωτήματος, το τελευταίο εκτελείται πάνω στο δείγμα. Στην περίπτωση που δεν υπάρχει τέτοιο δείγμα, αμέσως επόμενη επιλογή αποτελεί αυτό, που σαν σύνολο στηλών έχει ένα υπερσύνολο των στηλών που εμφανίζονται στο ερώτημα. Στην περίπτωση που κανένα από αυτά τα δείγματα δεν υπάρχουν, επιλέγεται εκείνο που περιέχει τις περισσότερες στήλες του ερωτήματος, στο μεγαλύτερο διαθέσιμο μέγεθος.

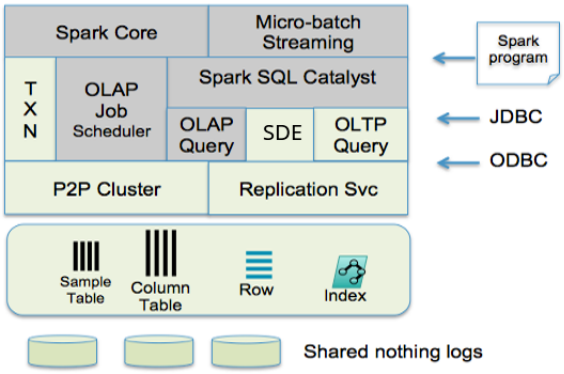
Ένα προσεγγιστικό ερώτημα είναι της μορφής:

SELECT ... FROM .. WHERE .. GROUP BY ...
WITH ERROR <fraction>[CONFIDENCE<fraction>] [BEHAVIOUR <string>]

όπου ο χρήστης ορίζει τη μέγιστη απόκλιση, την εμπιστοσύνη και προαιρετικά μία από τις διαθέσιμες ενέργειες ως BEHAVIOUR. Η τελευταία παράμετρος καθορίζει την συμπεριφορά της SnappyData αν η εκτέλεση πάνω στο δείγμα επιστρέψει απάντηση που δεν ικανοποιεί τους περιορισμούς. Οι επιλογές είναι η εμφάνιση ενός μηνύματος σφάλματος χωρίς κανένα αποτέλεσμα, η εμφάνιση των αποτελεσμάτων ακόμα και αν δεν ικανοποιούν τους περιορισμούς και η ολική ή μερική εκτέλεση, στην περίπτωση που δεν είναι όλες οι προσεγγίσεις εντός ορίων, του query πάνω στο συνολικό πίνακα.

Αρχιτεκτονική της SnappyData

Στο σχήμα 3.2 απεικονίζονται τα μέρη της SnappyData. Τα δεδομένα βρίσκονται κυρίως στη μνήμη και οι πίνακες που αποθηκεύονται μπορεί να έχουν format στηλών ή γραμμών. Για την αποθήκευση των πινάκων-στηλών χρησιμοποιούνται οι RDD δομές του Spark που επιτρέπουν τη συμπίεση των δεδομένων, ενώ για τους πίνακες-γραμμές, οι αντίστοιχες δομές του Gemfire, οι οποίες υποστηρίζουν τη δημιουργία δεικτών για την γρήγορη εκτέλεση αναγνώσεων και εγγραφών πάνω στα στοιχεία τους. Το format στηλών συμπιέζει τα δεδομένα και τα αποθηκεύει σε συνεχόμενες θέσεις μνήμης. Επομένως οι σύχρονοι επεξεργαστές εκτελούν ταχύτατα υπολογισμούς πάνω σε αυτά, για παράδειγμα άθροισμα και μέσο όρο. Οι



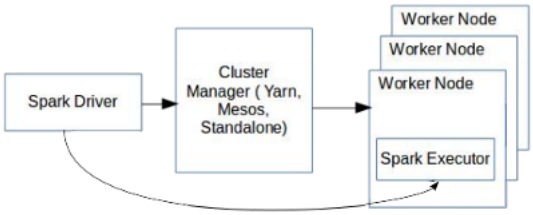
Σχήμα 3.2: Τα μέρη της SnappyData (Πηγή: [12])

πίνακες-γραμμές αποθηκεύονται σαν ολόκληρη γραμμή στη μνήμη. Η πρόσβαση σε αυτούς γίνεται με τη χρήση κλειδιών και hash συναρτήσεων, με αποτέλεσμα ταχείες ενημερώσεις και αναζητήσεις στοιχείων. Κατά τη δημιουργία του πίνακα, ο χρήστης μπορεί να επιλέξει το πλήθος των αντιγράφων, να διαμερίσει τον πίνακα σε κομμάτια ανάλογα με κάποιο κλειδί (primary key) και άλλα χαρακτηριστικά. Τέλος, στη μνήμη της SnappyData αποθηκεύονται και οι δομές που δημιουργήθηκαν για την προσεγγιστική επεξεργασία των ερωτημάτων.

Η επεξεργασία των δειγμάτων και η εκτέλεση των ερωτημάτων πάνω σε αυτά με σκοπό τη λήψη προσεγγιστικών και γρήγορων αποφάσεων, γίνεται από τη Synopsis Data Engine, όπως αναφέρθηκε παραπάνω. Οι συναλλαγές εκτελούνται στη μονάδα TXN και χρησιμοποιούν για το consensus ένα commit πρωτόκολλο δύο φάσεων με βάση το Paxos του Gemfire. Τέλος, οι ροές δεδομένων επεξεργάζονται από το Spark με χρήση της ειδικά διαμορφωμένης μονάδας του, το Spark Streaming.

Όταν καταφθάνει ένα ερώτημα στο σύστημα δημιουργείται ένα αρχικό πλάνο εκτέλεσης. Σύμφωνα με τα αποτελέσματα του, το σύστημα συμπεραίνει αν το προς εκτέλεση ερώτημα είναι υψηλού ή χαμηλού χρόνου απόκρισης (latency). Αν πρόκειται για OLTP εργασία και απαιτεί μικρό χρόνο εκτέλεσης, για να αποφευχθούν οι χρονικές δαπάνες εξαιτίας της χρονοδρομολόγησης, το στέλνει στο κατάλληλο κομμάτι δεδομένων. Διαφορετικά, το αναλαμβάνει ο OLAP χρονοδρομολογητής (scheduler), ο οποίος οργανώνει όλες τα αντίστοιχα ερωτήματα, όπως επίσης και τις εργασίες του Spark (Spark jobs).

Για τη συνοχή των αντιγράφων, τις γρήγορες ενημερώσεις τιμών και την άμεση ανίχνευση της κατάρρευσης κάποιου κόμβου του cluster, η SnappyData αξιοποιεί τα υπάρχοντα P2P (peer-to-peer) συστήματα του Gemfire. Σε αυτού του είδους τα δίκτυα κάθε μέλος του συστήματος μπορεί να επικοινωνεί με τα υπόλοιπα. Για την αντιγραφή των δεδομένων χρησιμοποιείται ένα SAN Volume Controller (SVC). Μέσω αυτού δεδομένα που βρίσκονται φυσικά σε ξεχωριστά συστήματα, φαίνονται σαν να υπάρχουν σε ένα κοινό σύστημα αποθήκευσης.



Σχήμα 3.3: Ένα Spark cluster (Πηγή: [21])

SnappyData Cluster

Η SnappyData χρησιμοποιεί το Spark σε μεγάλο βαθμό σαν βάση για τη λειτουργία της και επεκτείνει τα συστήματά του, ώστε να ταιριάζουν στις απαιτήσεις. Σε ένα Spark cluster, οι εφαρμογές τρέχουν ανεξάρτητα και συγχρονίζονται από το κύριο πρόγραμμα (driver program). Αυτές συνδέονται με τους διαχειριστές του cluster για να αποκτήσουν πρόσβαση σε κόμβους εκτέλεσης (executor nodes), όπου τρέχουν οι διεργασιές που εκτελούν υπολογισμούς και αποθηκεύουν δεδομένα για το τρέχον πρόγραμμα. Τέλος, το driver πρόγραμμα συνδέεται με τους executors μέσω ενός SparkContext. Μία αναπαράσταση των παραπάνω αποτελεί το σχήμα 3.3.

Για την επίτευξη των παραπάνω έγιναν αλλαγές στο μοντέλο του cluster. Τα μέλη ενός SnappyData cluster χωρίζονται σε τρεις κατηγορίες, όπως φαίνεται στο σχήμα 3.4. Οι locators γνωρίζουν τα μέλη που είναι συνδεδεμένα στο cluster ανά πάσα στιγμή και διαχειρίζονται τις αφίξεις και αναχώρησεις των κόμβων.

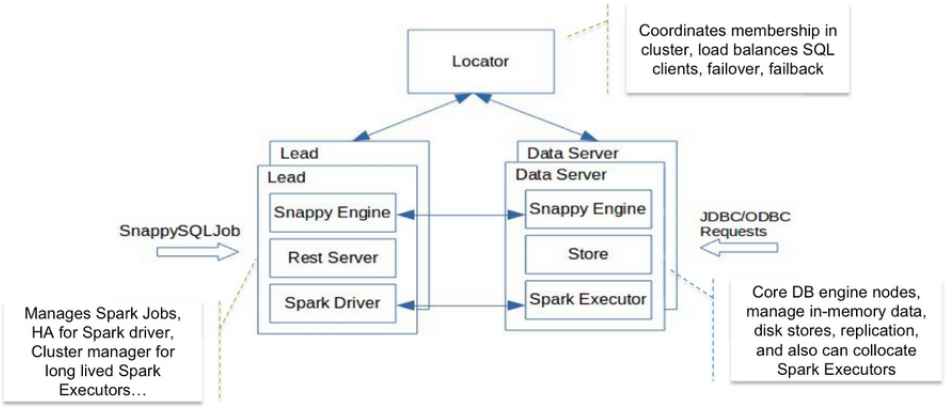
Μία άλλη κατηγορία μελών είναι οι lead κόμβοι, οι οποίοι λειτουργούν σαν Spark drivers διατηρώντας ένα SparkContext. Υπάρχει ένας κύριος lead κόμβος ενώ άλλοι βρίσκονται σε αναμονή, στην περίπτωση που αυτός καταρρεύσει. Εδώ γίνεται η χρονοδρομολόγηση και η εκτέλεση και των περισσότερων SQL ερωτημάτων που προωθούνται από την τελευταία κατηγορία μελών, τους data servers.

Οι data servers αποθηκεύουν τα δεδομένα και έχουν ενσωματωμένα ένα Spark executor και μία μηχανή SQL, για την εκτέλεση συγκεκριμένων ερωτημάτων ανεξάρτητα και αποδοτικότερα από το Spark. Όταν έρχεται ένα ερώτημα αυτό είτε εκτελείται τοπικά, είτε δρομολογείται προς τον lead κόμβο για εκτέλεση με χρήση της Spark SQL.

Σε αντίθεση με το Apache Spark, το οποίο είναι κυρίως μία υπολογιστική μηχανή, τα μέλη του SnappyData cluster διατηρούν μία μεταβλητή κατάσταση στα JVMs και απαιτούν όλες οι Spark εργασίες να μοιράζονται την ίδια κατάσταση. Για την πραγματοποίηση των παραπάνω χρειάστηκαν δύο επεκτάσεις στο Spark:

- Άυξηση διάρκειας ζωής των executors. Στο Spark η υποβολή νέων εργασιών ενεργοποιεί την δημιουργία νέων executors. Στο συγκεκριμένο σύστημα, οι executors σχηματίζουν ένα P2P cluster και η διάρκεια ζωής τους δεν συνδέεται με το χρόνο εκτέλεσης των Spark εφαρμογών.

- Ο Spark driver όπως αναφέραμε αναθέτει εργασίες στους executors. Εάν ο driver στα-



Σχήμα 3.4: Μέλη του SnappyData cluster (Πηγή: [5])

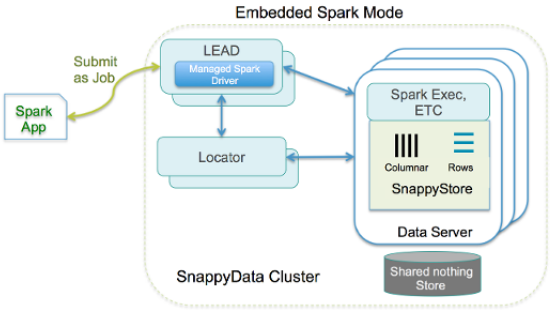
ματήσει απότομα τη λειτουργία του, το SparkContext κλείνει και μάζι με αυτό κλείνουν και οι executors χάνοντας όλα τα περιεχόμενα της κρυφής τους μνήμης. Η SnappyData χρησιμοποιεί το Spark JobServer για να διαχειρίζεται τις εργασίες και τα ερωτήματα από το lead κόμβο. Για την εξασφάλιση της διαθεσιμότητας, συνήθως χρησιμοποιούνται πάνω από ένας locator.

Χιλιάδες χρήστες μπορούν να συνδεθούν ταυτόχρονα στο SnappyData cluster χρησιμοποιώντας JDBC συνδέσεις. Το σύστημα για να ανταπεξέλθει σε αυτό τον όγκο εργασίας, κατηγοριοποιεί τα νέα αιτήματα ανάλογα με το χρόνο που απαιτούν. Με αυτό τον τρόπο δίνεται η δυνατότητα χειρισμού πολλαπλών λειτουργιών χαμηλού χρόνου απόκρισης, καθώς και ερωτημάτων πάνω σε μεγάλα σύνολα δεδομένων, ταυτόχρονα. Για γρήγορες εργασίες ο χρονοδρομολογητής παρακάμπτεται και η εκτέλεση οδηγείται κατευθείαν στα δεδομένα. Τα πολύπλοκα ερωτήματα διαχειρίζονται από τον μηχανισμό χρονοδρομολόγησης του Spark.

Το SnappyData cluster έχει σχεδιαστεί σαν μια βάση με μεγάλη διάρκεια λειτουργίας. Η κατάσταση διαχειρίζεται ως πίνακας, ο οποίος μπορεί να μοιραστεί σε οποιοδήποτε αριθμό συνδεδεμένων εφαρμογών. Τα δεδομένα αποθηκεύονται στη μνήμη και αντιγραφά τους δημιουργούνται σε τουλάχιστον ένα ακόμα κόμβο του συστήματος. Οι κόμβοι του συστήματος λειτουργούν για μεγάλο διάστημα και ο χρόνος ζωής τους δεν εξαρτάται από τη διάρκεια των εφαρμογών. Η SnappyData το πετυχαίνει αυτό αποσυνδέοντας τους μηχανισμούς εκκίνησης και τερματισμού που χρησιμοποιεί το Spark.

Ένας κατάλληλος τρόπος εκτέλεσης των υπολογισμών του Spark πάνω στα δεδομένα που βρίσκονται στη μνήμη, είναι να βρίσκονται τα τελευταία στο ίδιο JVM με τους Spark executors. Αυτός ο τρόπος σύνδεσης αποτελεί την πρωτότυπη ρύθμιση της SnappyData. Οι SnappyData servers ξεκινούν τη λειτουργία τους και συνδεόνται με το δίσκο ή εξωτερικά συστήματα αποθήκευσης φέρνοντας στη μνήμη δεδομένα. Οι Spark executors αρχίζουν τη λειτουργία τους όταν φτάνει η πρώτη Spark εργασία. Το σχήμα 3.5 αναπαριστά το SnappyData cluster για το συγκεκριμένο τρόπο σύνδεσης του Spark executor με τα υπόλοιπα μέλη.

Οι δύο άλλοι τρόποι σύνδεσης είναι όταν δεν χρησιμοποιείται cluster αλλά τα πάντα τρέχουν στο ίδιο JVM (Local Mode) [36] και όταν το Spark cluster και οι executors τρέχουν ξεχω-



Σχήμα 3.5: Απεικόνιση σύνδεσης Spark με SnappyData Data server (Πηγή: [23])

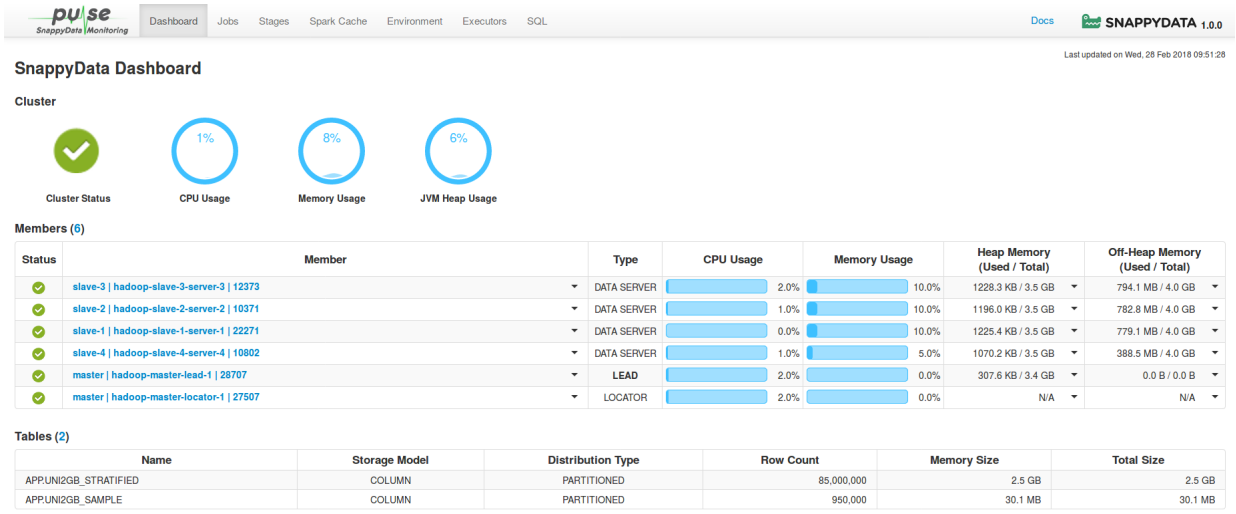
ριστά από τη SnappyData, η οποία λειτουργεί τώρα σαν σύστημα αποθήκευσης για το Spark (SnappyData Smart Connector Mode) [37].

Ο συγκεκριμένος τρόπος σύνδεσης διαθέτει κάποια πλεονεκτήματα έναντι των άλλων:

- Υψηλή απόδοση: Όλες οι Spark εφαρμογές έχουν πρόσβαση στον πίνακα, ο οποίος βρίσκεται τοπικά. Ακόμα, η μηχανή για τα ερωτήματα (query engine) αποφεύγει την αντιγραφή των δεδομένων, που απαιτεί πολύ χρόνο, ειδικά για μεγάλα μεγέθη δεδομένων.
- Μεγάλη διαθεσιμότητα: Όταν υποβάλλονται Spark εργασίες, ο lead κόμβος τις ανιχνεύει και εμποδίζει τον τερματισμό των executors ακόμα και αν ο Spark driver καταρρεύσει. Όσο λειτουργεί τουλάχιστον ένας lead κόμβος, οι Spark εργασίες συνεχίζουν να τρέχουν.
- Μικρή πολυπλοκότητα: Υπάρχει μόνο ένα cluster και έτσι ο χρήστης επιβλέπει, ρυθμίζει και κάνει αποσφαλμάτωση (debug) μόνο εκεί.

Η ομάδα της SnappyData έχει επεκτείνει τη διεπαφή χρήστη (User Interface-UI) που προσφέρει το Spark για να διευκολύνει την παρακολούθηση του cluster και των εργασιών. Το UI φαίνεται στο σχήμα 3.6. Όπως βλέπουμε από την εικόνα, στην αρχική σελίδα φαίνονται οι κόμβοι που είναι συνδεδεμένοι στο cluster. Συγκεκριμένα, αναγράφεται το όνομα του κόμβου, ο αριθμός της διεργασίας (Process ID-PID) και ο τύπος του κόμβου (server, locator ή leader). Επιπλέον, κάθε φορά που ανανεώνουμε τη σελίδα, δίνεται η κατάσταση του cluster και στατιστικά της χρήσης της μνήμης και της CPU για κάθε κόμβο ξεχωριστά, αλλά και συνολικά για το cluster. Τέλος, στο κάτω μέρος της αρχικής σελίδας εμφανίζονται οι πίνακες που έχουν δημιουργηθεί, ο τρόπος αποθήκευσής τους, δηλαδή στήλη ή γραμμή, το μέγεθος τους και κάποια ακόμη στοιχεία.

Στις υπόλοιπες καρτέλες ο χρήστης μπορεί να ενημερωθεί για τις SQL εργασίες που έχουν εκτελεστεί και τα επιμέρους στάδιά τους, τις παραμέτρους της SnappyData, όπως για παράδειγμα την έκδοση της Java που χρησιμοποιεί, κ.λ.π.



Σχήμα 3.6: Σελίδα παρακολούθησης του SnappyData cluster

Κεφάλαιο 4

XDB

Εισαγωγή

Όλο και περισσότερα ερωτήματα σε συσχετιστικές βάσεις δεδομένων (relational database systems) χρησιμοποιούν συναθροιστικές συναρτήσεις (aggregate functions). Καθώς τα δεδομένα πληθαίνουν, πέρα από την ανάγκη υπολογισμού συγκεκριμένων αποτελεσμάτων, εμφανίζεται και η ανάγκη εξαγωγής γενικών χαρακτηριστικών των δεδομένων. Οι χρήστες επιθυμούν αυτές τις πληροφορίες γρήγορα, παρά το γεγονός ότι για τις απαντήσεις απαιτείται πρόσβαση και εκτέλεση των ερωτημάτων σε μεγάλα μεγέθη δεδομένων. Η λειτουργία των συστημάτων προϋποθέτει την επεξεργασία ολόκληρων των δεδομένων για την παρουσίαση του αποτελέσματος. Κατά τη διάρκεια της εκτέλεσης, οι χρήστες περιμένουν χωρίς κάποια ανατροφοδότηση για την τελική απάντηση. Σε πολλές περιπτώσεις, είναι επιθυμητή μία προσεγγιστική λύση αλλά το σύστημα δεν τους παρέχει αυτή τη δυνατότητα.

Η XDB αποτελεί μία βάση δεδομένων που αντιμετωπίζει τα παραπάνω προβλήματα υποστηρίζοντας διαδραστικά συναθροιστικά ερωτήματα (online aggregation queries). Η λειτουργία της παρέχει στο χρήστη τη δυνατότητα να παρακολουθεί την πρόοδο του ερωτήματος και να ελέγχει την εκτέλεσή του, μέσω του Apache Zeppelin [45], δίνοντας προσεγγιστικά αποτελέσματα και εκτιμώμενα διαστήματα εμπιστοσύνης. Οι παράμετροι που δίνονται στο σύστημα περιλαμβάνουν το συνολικό χρόνο εκτέλεσης του προσεγγιστικού ερωτήματος, την εμπιστοσύνη στα διαστήματα εμπιστοσύνης και το χρονικό διάστημα που μεσολαβεί ανάμεσα σε δύο αποτελέσματα. Σε αντίθεση με άλλα συστήματα, όπως οι OLAP βάσεις δεδομένων, ο χρήστης ενημερώνεται καθόλη τη διάρκεια εκτέλεσης του ερωτήματος και έχει τον έλεγχο του χρόνου που απαιτείται για τη λήψη μίας ικανοποιητικής απάντησης.

Επεκτάσεις για υποστήριξη διαδραστικών συναθροιστικών ερωτημάτων

Για την online επεξεργασία κρίνεται απαραίτητη η επέκταση των συναθροιστικών συναρτήσεων, έτσι ώστε να μπορεί το σύστημα να παρέχει προσεγγιστικά αποτελέσματα. Ο υπολο-

γισμός των SUM, COUNT και AVG είναι απλή διαδικασία, ενώ για τα VAR και STD DEV έχουν αναπτυχθεί αλγόριθμοι όπως των Chan, Golub και LeVeque [48]. Επιπλέον, πρέπει να κατασκευαστούν συναρτήσεις, οι οποίες θα επιστρέφουν τα διαστήματα εμπιστοσύνης των προσεγγίσεων. Τέλος, ο εκτελεστής (executor) των ερωτημάτων πρέπει να μετατραπεί έτσι ώστε να παρέχει τις τρέχουσες προσεγγιστικές τιμές. Η Postgres αποτελεί ένα επεκτάσιμο σύστημα που επιτρέπει αλλαγές. Για την πραγματοποιήση του στόχου ήταν απαραίτητη η υλοποίηση των παρακάτω μεταβολών.

Τυχαία πρόσβαση στα δεδομένα

Για τον σωστό υπολογισμό του αποτελέσματος και των στατιστικών μεγεθών είναι απαραίτητη η τυχαία πρόσβαση στα δεδομένα. Ορισμένες μέθοδοι πρόσβασης επηρεάζουν τον τρόπο που ανακτούνται τα δεδομένα. Ωστόσο, οι παρακάτω τεχνικές προσφέρουν τυχαία πρόσβαση σε αυτά και μπορούν να χρησιμοποιηθούν στο σύστημα:

- Σάρωση αρχείων: Οι παραδοσιακές μέθοδοι πρόσβασης των δεδομένων, που είναι αποθηκευμένα σε heap αρχεία, δεν ανακτούν τις γραμμές σε συγκεκριμένη σειρά, και επομένως μπορούν να χρησιμοποιηθούν στα διαδραστικά ερωτήματα. Όμως οι πληροφορίες που αποθηκεύονται σε heap αρχεία, παρουσιάζουν κάποια μοτίβα, ανάλογα με την εισαγωγή των δεδομένων ή την ομαδοποίηση τους. Σε περίπτωση που αυτά σχετίζονται με τις τιμές που χρειάζεται το ερώτημα, το σύστημα θα πρέπει να λαμβάνει υπόψη του αυτό το γεγονός στις στατιστικές τεχνικές που χρησιμοποιεί ή να επιλέγει μία άλλη μέθοδο πρόσβασης.

- Σάρωση δεικτών (indexes): Η σάρωση ενός δείκτη επιστρέφει γραμμές οι οποίες είναι ταξινομημένες βασιζόμενες είτε σε κάποια δομή, όπως ένα B-tree, είτε είναι οργανωμένες σε ομάδες, όπως οι δείκτες hash. Και στις δύο περιπτώσεις ερωτήματα πάνω στις τιμές που έχουν δεικτοδοτηθεί θα έδιναν λάθος αποτελέσματα. Για παράδειγμα, έστω ότι η στήλη πάνω στην οποία έχει φτιαχτεί ο δείκτης περιέχει 1000 μηδενικά και 1000 μονάδες. Εάν ζητούσαμε το μέσο όρο των τιμών της στήλης, μία τέτοια πρόσβαση θα μας έδινε ένα προσεγγιστικό ανακριβές αποτέλεσμα. Ωστόσο, αν το ερώτημα δεν συναθροίζει τις τιμές των δεικτών αλλά διαφορετικών στηλών, η μέθοδος αυτή προσφέρει τυχαία πρόσβαση στα δεδομένα.

- Δειγματοληψία δεικτών: Η τελευταία τεχνική πρόσβασης χρησιμοποιεί ψευδοτυχαία δειγματοληψία από τις δομές δεδομένων των δεικτών [54]. Για αυτή τη διαδικασία επιλέγονται τυχαία κόμβοι του B-tree και με αυτόν τον τρόπο επιτυγχάνεται ο υπολογισμός χρήσιμων διαστημάτων εμπιστοσύνης. Από την άλλη, η αποτελεσματικότητα της μπορεί να είναι χαμηλότερη από τις σαρώσεις αρχείων ή δεικτών, εφόσον απαιτείται η δοκιμή διαφόρων μονοπατιών του δέντρου και έτσι η προσκόμιση δεδομένων (prefetch) συχνά δεν ωφελεί.

Υποστήριξη GROUP BY ερωτημάτων

Ένα σύστημα που εκτελεί online ερωτήματα συνάθροισης καλείται να παρέχει απαντήσεις στο χρήστη το συντομότερο δυνατόν. Επιπλέον, στις περιπτώσεις όπου τα ερωτήματα έχουν κάποια δήλωση GROUP BY, τα αποτελέσματα κάθε ομάδας πρέπει να ενημερώνονται με ίδιο ρυθμό. Μία συνηθισμένη μέθοδος οργάνωσης των αποτελεσμάτων για αυτά τα ερωτήματα είναι η ταξινόμηση τους ανάλογα με τα αντίστοιχα πεδία. Ωστόσο, ο αλγόριθμος της ταξινόμησης είναι ένας blocking αλγόριθμος, δηλαδή το τελικό αποτέλεσμά του είναι διαθέσιμο μόνο αφού έχουν επεξεργαστεί ολόκληρα τα δεδομένα. Επιπλέον, με αυτό τον τρόπο τα αποτελέσματα για κάθε ομάδα τιμών υπολογίζονται ξεχωριστά και σειριακά. Επομένως, αυτή η τεχνική δεν μπορεί να εφαρμοστεί για online συναθροιστικά ερωτήματα.

Μία εναλλακτική μέθοδος είναι η χρήση μίας hash συνάρτησης πάνω στις στήλες που υπάρχουν στο GROUP BY. Αυτός ο αλγόριθμος είναι non-blocking και προσφέρει καινούργια ακριβέστερα αποτελέσματα σε κάθε ομάδα τιμών αμέσως μόλις διαβαστούν οι νέες γραμμές που αντιστοιχούν σε αυτή. Ένα μειονέκτημα της μεθόδου είναι οι απαιτήσεις της σε μνήμη. Όσο μεγαλώνουν οι ομάδες τιμών και ξεπερνούν το διαθέσιμο χώρο στη προσωρινή μνήμη (buffer), ο απόδοση του αλγορίθμου δεν κλιμακώνεται καλά. Για την επίλυση αυτού του προβλήματος χρησιμοποιείται η μέθοδος Hybrid Cache [46], η οποία όταν ο hash πίνακας χωράει στη μνήμη αποδίδει σαν την απλή προσέγγιση, ενώ κλιμακώνει καλά όταν αυτός την ξεπερνάει σε μέγεθος.

Με βάση τα παραπάνω, ενημέρωση κάθε αποτελέσματος θα πραγματοποιείται μετά την επεξεργασία των γραμμών της κάθε ομάδας. Με τη hash μέθοδο, τα δεδομένα φτάνουν τυχαία, και έτσι είναι πιθανό τα αποτελέσματα ομάδων τιμών με λίγες γραμμές να ενημερώνονται πολύ σπάνια. Για να γίνεται ενημέρωση όλων των αποτελεσμάτων με τον ίδιο ρυθμό, μία λύση είναι να γίνεται κυκλική (round-robin) ανάγνωση των δεδομένων, δηλαδή μία γραμμή από την πρώτη ομάδα, μία από τη δεύτερη και ούτω καθεξής. Για να παρέχει διαστήματα εμπιστοσύνης με παρόμοιο εύρος, το σύστημα θα μπορούσε να χρησιμοποιεί ένα σταθμισμένο αλγόριθμο κυκλικής ανάγνωσης δεδομένων.

Για την επίτευξη των παραπάνω έχει αναπτυχθεί ένας αλγόριθμος ανάγνωσης δεδομένων που βασίζεται στους δείκτες του σχήματος [47]. Για την εκτέλεση του, είναι απαραίτητη η ύπαρξη ενός B-δέντρου (B-tree) με δείκτες των δεδομένων. Με το πρώτο αίτημα για δεδομένα, γίνεται μία ανάγνωση στην αριστερότερη άκρη του δέντρου όπου βρίσκεται ο δείκτης k_1. Αφού επιστραφεί το πρώτο πακέτο δεδομένων με δείκτη k_1, στο επόμενο αίτημα για δεδομένα γίνεται αναζήτηση για ένα δείκτη k_2 λίγο μεγαλύτερο από τον k_1. Αυτή η διαδικασία τερματίζεται όταν έχει βρεθεί δείκτης k_n, τέτοιος ώστε να μην υπάρχει δείκτης μεγαλύτερος του. Τα επόμενα αιτήματα εξυπηρετούνται όπως είπαμε με έναν σταθμισμένο κυκλικό αλγόριθμο.

Με την κατάλληλη προετοιμασία της μνήμης, αυτή η μέθοδος είναι τουλάχιστον τόσο αποτελεσματική όσο η σαρωσή δεδομένων μέσω δεικτών που δεν έχουν ομαδοποιηθεί. Κάθε γραμμή θα ερωτηθεί ακριβώς μία φορά, και για αυτό μπορεί να είναι απαραίτητη κάποια εργασία Ε/Ε. Η απόδοση του αλγορίθμου βελτιώνεται αν ο δείκτης είναι η κύρια μέθοδος πρόσβασης του πίνακα, αν αυτός έχει ομαδοποιηθεί στις στήλες του GROUP BY ή αν οι δείκτες περιέχουν δεδομένα και των στηλών του GROUP BY και των συναθροιστικών συναρτήσεων, με

πρόθεμα την πρώτη. Σε κάθε περίπτωση η αποδοτικότητα του αλγορίθμου είναι τόσο καλή όσο η σάρωση των δευτερευόντων (secondary) δεικτών. Ένα πλεονέκτημα της μεθόδου είναι ότι δίνεται η δυνατότητα στο χρήστη να μεταβάλλει την ταχύτητα ενημέρωσης των αποτελεσμάτων για κάθε ομάδα ξεχωριστά. Τέλος, όταν κάποια ομάδα φτάσει στο τελικό αποτέλεσμα, οι υπόλοιπες συνεχίζουν την προσεγγιστική επεξεργασία με αυξημένο ρυθμό.

Υπολογισμός διαστημάτων εμπιστοσύνης

Τα διαστήματα εμπιστοσύνης είναι ένας τρόπος αξιολόγησης των προσεγγιστικών αποτελεσμάτων. Όταν αυτό έχει μεγάλο εύρος, προειδοποιεί το χρήστη ότι οι γραμμές, που έχουν επεξεργαστεί, δεν αρκούν για να υπολογιστεί μια χρήσιμη απάντηση. Επιπλέον, όπως αναφέρθηκε ο χρήστης μπορεί να σταματήσει την εκτέλεση του ερωτήματος όταν κρίνει ότι το μέγιστο σφάλμα της προσέγγισης είναι αρκετά μικρό. Το διάστημα εμπιστοσύνης έχει σημασία όταν τα δεδομένα επιλέγονται με τυχαία σειρά, και επομένως τα δεδομένα που έχουν επιλεγεί, αποτελούν ένα τυχαίο δείγμα χωρίς επανατοποθέτηση του συνολικού πληθυσμού.

Υπάρχουν αρκετές κατηγορίες διαστημάτων εμπιστοσύνης που μπορούν να υπολογιστούν μετά από n δεδομένα:

- Συντηρητικά διαστήματα εμπιστοσύνης (conservative confidence intervals): Περιέχουν την τελική απάντηση με μία πιθανότητα μεγαλύτερη ή ίση με p. Ο υπολογισμός τους βασίζεται στην ανισότητα του Hoeffding [49], ή σε επεκτάσεις της [50].

- Διαστήματα εμπιστοσύνης μεγάλου δείγματος (large-sample confidence intervals): Εδώ, όπως και πριν η τελική απάντηση βρίσκεται μέσα στο διάστημα με πιθανότητα p, και ο προσδιορισμός των διαστημάτων βασίζεται στο Κεντρικό οριακό θεώρημα [52]. Τέτοια διαστήματα μπορούν να υπολογιστούν όταν το πλήθος των δεδομένων, που έχει επεξεργαστεί το σύστημα, είναι αρκετά μικρό, έτσι ώστε αυτά να μπορούν να θεωρηθούν ένα δείγμα με επανατοποθέτηση από το συνολικό πληθυσμό, ενώ ταυτόχρονα αρκετά μεγάλο, για να μπορεί να εφαρμοστεί με ακρίβεια το θεώρημα. Αυτού του είδους τα διαστήματα έχουν το μειονέκτημα ότι η πραγματική πιθανότητα να ανήκει το τελικό αποτέλεσμα στο διάστημα εμπιστοσύνης είναι πολλές φορές αρκετά μικρότερη από την εικονική p. Το πλεονέκτημα τους απέναντι στα συντηρητικά διαστήματα εμπιστοσύνης είναι ότι στις περισσότερες περιπτώσεις, τα διαστήματα μεγάλου δείγματος έχουν μικρότερο εύρος.

- Αιτιοκρατικά διαστήματα εμπιστοσύνης (deterministic confidence intervals): Περιέχουν το ακριβές αποτέλεσμα με πιθανότητα 1 [52]. Για την υλοποίηση τέτοιων διαστημάτων υποθέτουμε την ύπαρξη αριθμών a και b, τέτοιες ώστε $a \leq v(i) \leq b$. Τέτοιες τιμές μπορούν να προκύψουν από τον κατάλογο συστήματος (system catalog) της βάσης, όπου συχνά αποθηκεύονται οι οριακές τιμές από τον βελτιστοποιητή ερωτημάτων. Η χρήση τους είναι κατάλληλη όταν το πλήθος των επεξεργασμένων δεδομένων είναι πολύ μεγάλο. Σε αντίθεση με άλλες μορφές διαστημάτων, σε αυτή την προσέγγιση το τωρινό προσεγγιστικό αποτέλεσμα δεν βρίσκεται απαραίτητα στο κέντρο του διαστήματος.

time (ms)	nsamples	nrejected	sum	rel. CI
207	72	0	41905823233.75000000	0.111494
416	133	0	38878433808.49624060	0.091990
602	222	0	40858994536.62162162	0.067455
800	291	0	39568406437.73195876	0.061471
1008	373	0	40391909152.35924933	0.053441
1212	460	0	40572575127.19565217	0.047149
1404	534	0	40253381301.23595506	0.043751
1600	607	0	40114491490.37891269	0.040810
1800	658	0	40406139800.69908815	0.038810
2005	711	0	40197610157.46835443	0.037365
2205	767	0	40439019754.77183833	0.035734
2400	836	0	40561136858.32535885	0.034065
2609	921	0	40620790902.01954397	0.032465
2805	984	0	40823572513.17073171	0.031181
3000	1051	0	41149858772.22645100	0.029993
3200	1130	0	41281069248.63716814	0.028989
3404	1220	0	41321008548.00000000	0.027983
3602	1278	0	41242479548.87323944	0.027338
3805	1365	0	40910385414.65934066	0.026799

Σχήμα 4.1: Παράδειγμα εξόδου της XDB

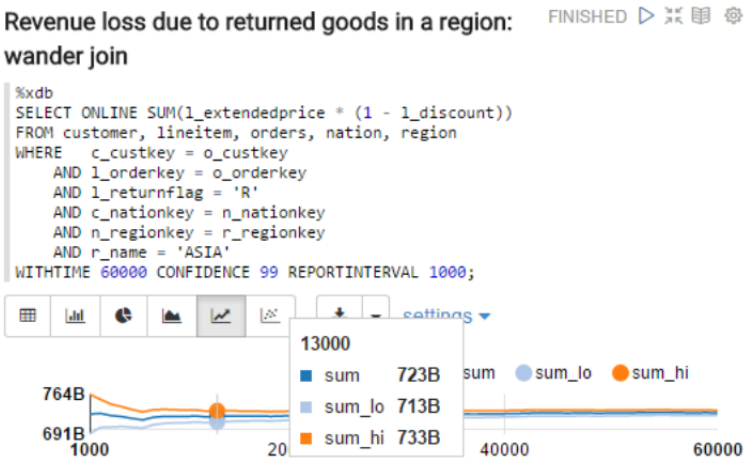
Στην πραγματικότητα, η καλύτερη λύση είναι ο συνδιασμός των παραπάνω μεθόδων για τον υπολογισμό των διαστημάτων ανάλογα με το μέγεθος των δεδομένων που έχουν επεξεργαστεί. Στην XDB, οι μέθοδοι που χρησιμοποιούνται περισσότερο είναι των διαστημάτων μεγάλου δείγματος και των αιτιοκρατικών διαστημάτων. Η αναλυτική περιγραφή της διαδικασίας υπολογισμού για κάθε περίπτωση ερωτήματος γίνεται στο [52].

Υλοποίηση συστήματος

Η XDB ενσωματώνει τις τεχνικές της online συνάθροισης σε μια πρόσφατη έκδοση της PostgreSQL, συγκεκριμένα την έκδοση 9.4.2. Οι επεκτάσεις αυτές καλύπτουν όλα τα επίπεδα εκτέλεσης ερωτημάτων (pipeline). Για τις στήλες που εμφανίζονται στα JOIN και στα WHERE, κατασκευάζονται δευτερεύοντες δομές δεικτών, όπως B-trees. Επίσης, η XDB μεταβάλει τον SQL αναλυτή (parser), τον βελτιστοποιητή (optimizer) των ερωτημάτων και τον executor για να υποστηρίζουν λέξεις-κλειδιά όπως CONFIDENCE, ONLINE, WITHTIME και REPORTINTERVAL. Ο optimizer αντικαθιστά τις απλές συναθροιστικές συναρτήσεις με online συναθροιστικές και με τους τελεστές (operators) των σχετικών διαστημάτων εμπιστοσύνης. Τέλος, έγιναν οι απαραίτητες αλλαγές με σκοπό την υποστήριξη μιάς αποδοτικής λύσης για JOIN ερωτήματα, που βασίζεται στο wander join αλγόριθμο [51].

Ο χρήστης σε κάθε διαδραστικό ερώτημα δίνει σαν παραμέτρους το συνολικό χρόνο εκτέλεσης σε milliseconds, την εμπιστοσύνη επί τοις εκατό και το διάστημα ανάμεσα στα αποτελέσματα πάλι σε milliseconds. Επιπλέον, αντί για SELECT, χρησιμοποιεί τη δήλωση SELECT ONLINE. Ένα παράδειγμα ερωτήματος με online άθροισμα πάνω σε TPC-H δεδομένα είναι το παρακάτω:

SELECT ONLINE SUM ($l\_extendedprice * (1 - l\_discount)$), COUNT(*)
FROM customer, orders, lineitem WHERE $c\_mktsegment$ ='BUILDING'
AND $c\_custkey = o\_custkey$ AND $l\_orderkey = o\_orderkey$
WITHTIME 20000 CONFIDENCE 95 REPORTINTERVAL 1000;



Σχήμα 4.2: Διεπαφή της XDB μέσω Apache Zeppelin (Πήγη: [55])

Αυτό το ερώτημα καλεί το σύστημα να εκτελέσει ένα online ερώτημα με JOIN 3 πινάκων, να δώσει προσεγγιστικές απαντήσεις για τα SUM και COUNT, και τα αντίστοιχα διαστήματα με 95% εμπιστοσύνη. Ο συνολικός χρόνος εκτέλεσης είναι 20 δευτερόλεπτα και η ενημέρωση των προσεγγίσεων θα γίνεται κάθε ένα δευτερόλεπτο.

Η έξοδος του συστήματος γίνεται αυτόματα στο terminal. Ένα παράδειγμα αποτελεί η εικόνα 4.1. Για τη διευκόλυνση του χρήστη στο χειρισμό του συστήματος, στο πλαίσιο της XDB αναπτύχθηκε μία διεπαφή μέσω του Apache Zeppelin [45]. Εκεί αναπαριστώνται τα ενημερωμένα αποτελέσματα και τα διαστήματα εμπιστοσύνης τους. Ένα στιγμιότυπο της διεπαφής φαίνεται στο σχήμα 4.2

Κεφάλαιο 5

Πειραματική Αξιολόγηση

Παράμετροι αξιολόγησης

Στο πειραματικό μέρος της διπλωματικής μελετήθηκε η απόδοση μόνο των SnappyData και XDB. Η BlinkDB δεν αξιολογήθηκε μαζί με τα άλλα δύο συστήματα επειδή οι αλγόριθμοι που περιγράφονται από την ομάδα που την υλοποίησε, δεν είναι open-source. Επίσης, η SnappyData αποτελεί ένα πιο σύγχρονο μοντέλο της BlinkDB, οπότε θεωρήσαμε ότι δεν έχει νοήμα μία σύγκριση μεταξύ των δύο. Σαν παραμέτρους αξιολόγησης θεωρήσαμε το χρόνο απόκρισης και την ακρίβεια του εκάστοτε αποτελέσματος, ενώ στόχος είναι η εξαγωγή συμπερασμάτων για τη συμπεριφορά των συστημάτων ανάλογα με τα ερωτήματα και την κατανομή που ακολουθούν τα δεδομένα.

Σύστημα αξιολόγησης

Για την διεξαγωγή των πειραμάτων χρησιμοποιήθηκε ένα σύστημα πέντε εικονικών μηχανημάτων (VMs), τα οποία δημιουργήθηκαν σε ένα private Openstack [53] cloud του εργαστηρίου Υπολογιστικών Συστημάτων (CSlab). Το cluster αποτελείται από ένα master και τέσσερις slaves. Κάθε VM είχε 8 GB RAM, 4 VCPUs και 80 GB σκληρό δίσκο. Το λειτουργικό σύστημα όλων των μηχανημάτων είναι Ubuntu 16.04 x64. Στο cluster στήθηκε το Hadoop 2.6.4, από το οποίο χρησιμοποιήθηκε το κατανεμημένο σύστημα αποθήκευσης HDFS. Η έκδοση της SnappyData που χρησιμοποιήθηκε είναι η 1.0 enterprise, η οποία εγκαταστάθηκε σε όλα τα VMs. Η XDB στήθηκε μόνο σε ένα VM, το οποίο είχε ίδιες προδιαγραφές με τα υπόλοιπα.

Οργάνωση πειραμάτων

Για τα πειράματα κατασκευάσαμε τρία σύνολα δεδομένων (datasets) διαφόρων μεγεθών και κατανομών, όπως φαίνεται στον πίνακα 5.1. Αυτά φορτώθηκαν στις βάσεις δεδομένων με

μορφή πίνακα T επτά στηλών: $c1, c2, .., c7$. Τα μεγέθη των δεδομένων επιλέχτηκαν έτσι ώστε να εξάγουμε συμπεράσματα για τη λειτουργία της SnappyData σε σύγκριση με την XDB, όταν τα δεδομένα χωράνε στη μνήμη του cluster και όταν την ξεπερνούν σαν μέγεθος. Για το προσεγγιστικό μέρος δημιουργήθηκαν ένα ομοιόμορφο δείγμα, ένα στρωματοποιημένο με QCS μία τυχαία στήλη του πίνακα και ένα στρωματοποιημένο με QCS δύο στήλες του πίνακα. Όλα τα δείγματα κατασκευάστηκαν με λόγο δειγματοληψίας 0.01. Το ομοιόμορφο δείγμα κατασκευάστηκε με χρήση του Spark, πάνω στο οποίο τα ερωτήματα εκτελέστηκαν μέσω του snappy-shell.

Μέγεθος δεδομένων (GB)	2, 15, 50
Κατανομή δεδομένων	uniform, zipfian $s = 1.5$
Εύρος τιμών	[0-1000]
Λόγος δειγματοληψίας	0.01
QCS	$\{c3\}$, $\{c2, c3\}$

Πίνακας 5.1: Παράμετροι δεδομένων

Αποτελέσματα της μελέτης

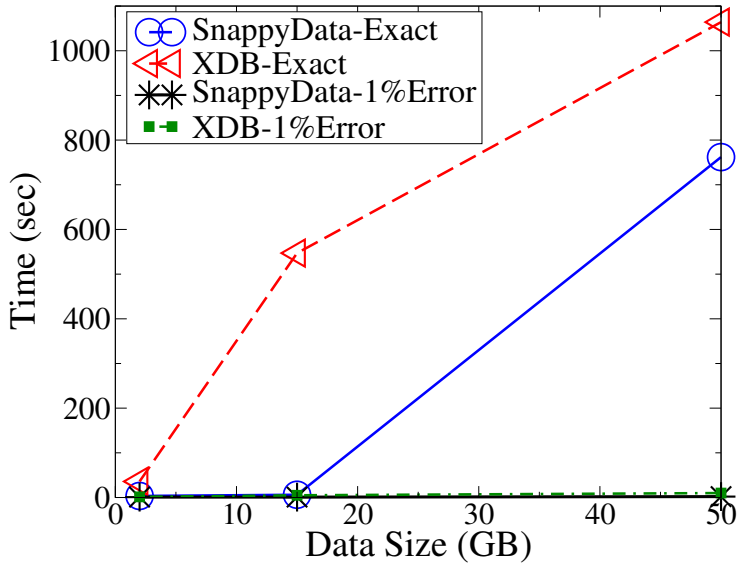
Τυχαία ερωτήματα

Πάνω στα δεδομένα που παράξαμε, εκτελέστηκαν αρχικά δύο workloads. Το πρώτο σύνολο ερωτημάτων, έστω w1, αποτελείται από έντεκα ερωτήματα όπου υπολόγιζουν SUM ή COUNT σε τυχαίες στήλες του πίνακα και δεν είχαν επιλεκτικότητα. Το δεύτερο, w2, αποτελείται από είκοσι ερωτήματα με τις ίδιες συναθροιστικές συναρτήσεις αλλά περιείχε και δηλώσεις WHERE σε διάφορες στήλες του πίνακα. Για τη δειγματοληψία που έγινε στη SnappyData επιλέχτηκε σαν QCS μία τυχαία στήλη του πίνακα.

Κλιμακωσιμότητα με το μέγεθος των δεδομένων

Από το πρώτο σύνολο ερωτημάτων μετρήσαμε το μέσο χρόνο εκτέλεσης των ερωτημάτων του w1 είτε για εκτέλεση πάνω σε ολόκληρο τον πίνακα είτε για προσεγγιστική εκτέλεση. Από τα δύο συστήματα ζητήσαμε σαν σφάλμα 1%. Το διάγραμμα του χρόνου απόκρισης σε σύγκριση με το μέγεθος του dataset είναι το 5.1. Όπως φαίνεται, η SnappyData απαντάει σε πολύ μικρό χρόνο όταν ο πίνακας είναι μικρότερος ή ίσος με τα 15 GB, ενώ στην περίπτωση των 50 GB η αύξηση είναι πολύ μεγάλη. Αυτή η απότομη διόγκωση του χρόνου οφείλεται στο γεγονός ότι ο πίνακας πλέον δεν χωράει στη μνήμη του cluster και είναι απαραίτητη η πρόσβαση στο δίσκο.

Αντίστοιχα, η XDB δίνει γρήγορες απαντήσεις πάνω σε όλα τα δεδομένα όταν αυτά έχουν μέγεθος 2 GB, ενώ ο χρόνος απόκρισης αυξάνεται ανάλογα με το μέγεθος του πίνακα. Αντίστοιχα με τη SnappyData, στην XDB που έχει στηθεί σε ένα μόνο VM, ο χρόνος εκτέλεσης μεγαλώνει απότομα όταν τα δεδομένα ξεπεράσουν τη μνήμη του μηχανήματος.



Σχήμα 5.1: Χρόνος εκτέλεσης για διάφορα μεγέθη δεδομένων

Αντίθετα, βλέπουμε ότι οι προσεγγιστικές λύσεις παρέχουν απαντήσεις σε πολύ μικρό χρονικό διάστημα, ανεξάρτητα από το μέγεθος του αρχικού συνόλου δεδομένων. Με αυτά τα αποτελέσματα αναδεικνύεται η σημασία της προσεγγιστικής επεξεργασίας.

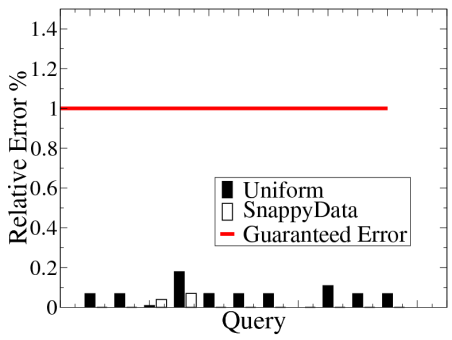
Ακρίβεια ομοιόμορφου δείγματος και SnappyData

Τα σχήματα 5.2 προέκυψαν από τα w1 και w2 πάνω στα δεδομένα με μέγεθος 2 GB και ομοιόμορφη κατανομή. Τα δύο διαγράμματα συγκρίνουν το σφάλμα του ομοιόμορφου δείγματος και της SnappyData. Οι μαύρες στήλες αντιστοιχούν στο ομοιόμορφο δείγμα και οι λευκές στη SnappyData με QCS μία τυχαία στήλη του πίνακα. Για κάθε ερώτημα ξεκινήσαμε ζητώντας σφάλμα 1% και εάν η SnappyData δεν επέστρεφε αποτέλεσμα, μεγαλώναμε σταδιακά το σφάλμα. Το σφάλμα που δώσαμε τελικά σαν παράμετρο σε αυτή έτσι ώστε να μας επιστρέψει κάποιο αποτέλεσμα, αναπαρίσταται με κόκκινη γραμμή.

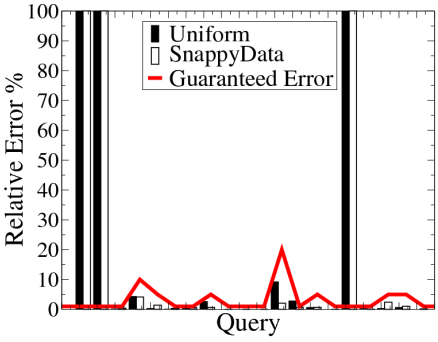
Από το 5.2α΄ παρατηρούμε ότι οι διαφορές στα πραγματικά σφάλματα των δειγμάτων είναι μικρές. Η SnappyData μας δίνει απαντήσεις οι οποίες πληρούν σε κάθε ερώτημα το σφάλμα που δώσαμε σαν παράμετρο.

Στο 5.2β΄ τα αποτελέσματα του ομοιόμορφου δείγματος και της SnappyData δίνουν παρόμοια σφάλματα. Ωστόσο παρατηρούμε ότι συγκριτικά με το σχήμα 5.2α΄ οι αποκλίσεις είναι μεγαλύτερες όταν τα ερωτήματα έχουν επιλεκτικότητα, ακόμα και αν τα δεδομένα ακολουθούν ομοιόμορφη κατανομή. Σε μερικά ερωτήματα κανένα από τα δύο συστήματα δεν κατάφερε να προσφέρει απαντήσεις μέσα στα όρια που θέσαμε.

Στο διάγραμμα 5.3 παρουσιάζονται τα σφάλματα των αντίστοιχων προσεγγιστικών απαντήσεων όπου όμως τα δεδομένα ακολουθούν τώρα zipfian κατανομή με $s = 1.5$. Τα πρώτα 6 ερωτήματα είναι COUNT, ενώ τα τελευταία 5 για SUM. Στη συγκεκριμένη περίπτωση ο φόρτος εργασίας είναι χωρίς επιλεκτικότητα. Βλέπουμε στην εικόνα ότι το στρωματοποιημένο



(α′) Σύνολο ερωτημάτων w1



(β′) Σύνολο ερωτημάτων w2

Σχήμα 5.2: Ομοιόμορφο δείγμα και SnappyData στα δεδομένα με μέγεθος 2GB και ομοιόμορφη κατανομή

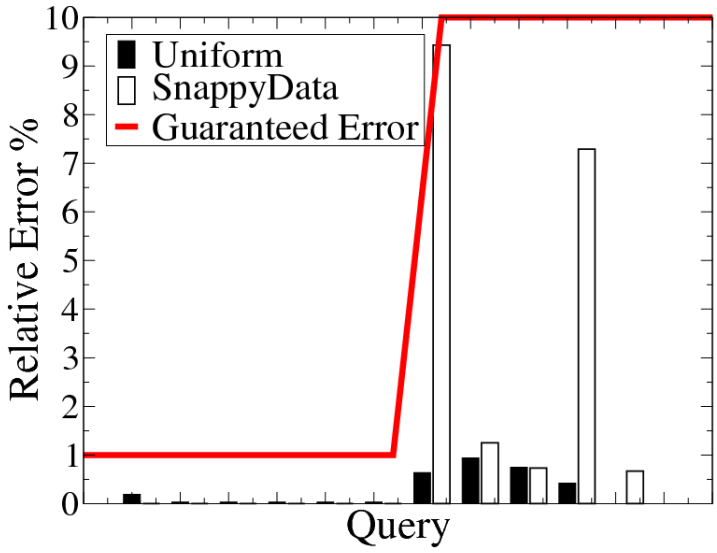
δείγμα δίνει απαντήσεις παρόμοιες με το πραγματικό αποτέλεσμα για τα COUNT ερωτήματα, ενώ για τα SUM τα αποτελέσματα έχουν μεγάλο σφάλμα. Αυτό οφείλεται στο γεγονός ότι τα πρώτα είναι πιο εύκολο να προσεγγιστούν καθώς τα προσεγγιστικά SUM επηρεάζονται όχι μόνο από το μέγεθος των δεδομένων αλλά και από τις τιμές τους.

Το ομοιόμορφο δείγμα κατάφερε να φέρει σε κάθε περίπτωση αποτελέσματα που είναι αρκετά ακριβή, σε αντίθεση με τη SnappyData η οποία έδωσε σφάλματα που πλησίασαν το 10%. Η δειγματοληψία που έγινε κατά τη δημιουργία του δείγματος επηρέασε αυτά τα αποτελέσματα. Το τυχαίο QCS οδήγησε στην κατασκευή ενός δείγματος που αναπαριστά καλά την τυχαία στήλη ενώ δεν δίνει τόση σημασία στις υπόλοιπες, με αποτέλεσμα όταν γίνονται ερωτήματα πάνω σε αυτές να έχει μειωμένη απόδοση. Έπειτα θα δείξουμε ότι όταν τα ερωτήματα είναι πάνω στο QCS, η απόδοση της SnappyData είναι καλύτερη.

Χρονική σύγκριση SnappyData και XDB

Όπως περιγράψαμε στα αντίστοιχα κεφάλαια, η SnappyData δέχεται σαν παράμετρο στο ερώτημα το επιθυμητό σφάλμα και την εμπιστοσύνη σε αυτό. Αντίθετα, η XDB παίρνει σαν όρισμα το συνολικό χρόνο εκτέλεσης του ερωτήματος και την εμπιστοσύνη. Όταν θελήσαμε να συγκρίνουμε τη SnappyData με την XDB, επειδή τα ερωτήματα είναι διαφορετικά, ακολουθήσαμε την εξής διαδικασία: αρχικά, εκτελέσαμε το ερώτημα στη SnappyData μετρώντας το χρόνο απόκρισης του συστήματος και σημειώνοντας το σφάλμα που δώσαμε σαν παράμετρο. Έπειτα, τρέξαμε το ερώτημα στη XDB και καταγράψαμε το χρόνο που χρειάστηκε για να φτάσει αυτό το σφάλμα. Στο σχήμα 5.4 παρουσιάζονται οι χρόνοι εκτέλεσης των ερωτημάτων του συνόλου w1. Τα δεδομένα πάνω στα οποία έγιναν οι μετρήσεις είναι μεγέθους 2 GB. Στο αριστερό διάγραμμα παρουσιάζονται τα αποτελέσματα των δεδομένων με ομοιόμορφη κατανομή, ενώ στο δεξιά τα δεδομένα με zipfian. Και στα δύο σχήματα τα πρώτα 6 ζεύγη στηλών είναι για ερωτήματα COUNT, ενώ τα τελευταία 5 για SUM.

Όταν η κατανομή των δεδομένων είναι ομοιόμορφη, παρατηρούμε στο 5.4α′ ότι για ερω-

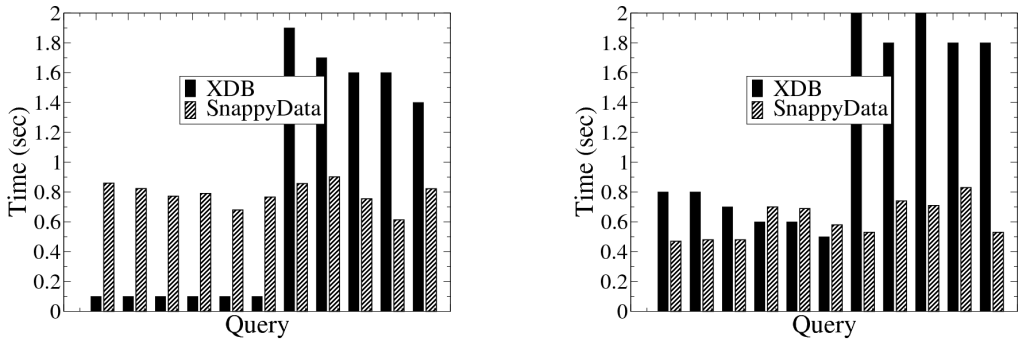


Σχήμα 5.3: Ομοιόμορφο δείγμα και SnappyData στα δεδομένα με μέγεθος 2GB και zipfian κατανομή για w1 workload

τήματα με COUNT, προσεγγίζει το τελικό αποτέλεσμα πιο γρήγορα η XDB από τη SnappyData. Η XDB παρέχει αποτελέσματα από τα πρώτα milliseconds εκτέλεσης, καθώς βασίζεται σε online aggregation. Επιπλέον, στην περίπτωση της ομοιόμορφης κατανομής, η XDB προσεγγίζει την κατανομή γρήγορα και είναι ικανή να δώσει ικανοποιητικά αποτελέσματα αμέσως. Στην περίπτωση της SnappyData, το ερώτημα υποχρεωτικά εκτελείται σε όλο το δείγμα, το οποίο εισάγει χρονική καθυστέρηση.

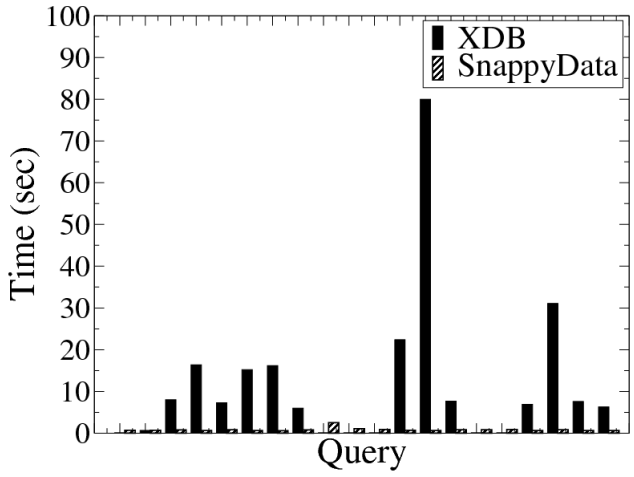
Στο σχήμα 5.4β΄, είναι εμφανής η αύξηση του χρόνου που απαιτεί η XDB για την παροχή του επιθυμητού σφάλματος. Η δυσκολία αυτή έγγυται στο ότι η διαδραστική συνάθροιση δεν είναι εξίσου αποτελεσματική για δεδομένα που δεν ακολουθούν ομοιόμορφη κατανομή. Σε κάθε περίπτωση βλέπουμε ότι τα SUM ερωτήματα είναι πίο δύσκολο για τη XBB να προσεγγιστούν από τα COUNT.

Στο διάγραμμα 5.5 παρουσιάζονται οι χρόνοι απόκρισης των δύο συστημάτων για την εκτέλεση του w2 σε δεδομένα 2GB που ακολουθούν ομοιόμορφη κατανομή. Εδώ φαίνεται ότι το online aggregation αδυνατεί να υποστηρίξει ικανοποιητικά τα ερωτήματα με επιλεκτικότητα, σε αντίθεση με τη SnappyData. Σε μερικά ερωτήματα η XDB χρειάστηκε περισσότερο χρόνο να απαντήσει προσεγγιστικά από το χρόνο εκτέλεσης του ερωτήματος στο ολικό dataset. Το γεγονός αυτό δικαιολογείται αν λάβουμε υπόψη μας ότι κάθε ανάγνωση γραμμής δεδομένων πρέπει να είναι τυχαία, ενώ στην εκτέλεση σε όλα τα δεδομένα, αυτά διαβάζονται σειριακά. Επιπλέον, παρατηρήθηκε ότι το σφάλμα μειώνεται εκθετικά με το μέγεθος των δεδομένων που έχουν επεξεργαστεί. Επομένως, η μεταβολή του σφάλματος, από κάποιο σημείο και έπειτα, είναι μικρή. Ένα παράδειγμα από τα πειράματα μας είναι ότι μόλις σε πέντε δευτερόλεπτα η XDB μας έδωσε σφάλμα 5% και χρειάστηκε συνολικά τριάντα δευτερόλεπτα για να μειώσει το σφάλμα στο 1%.



(α′) Ομοιόμορφη κατανομή (β′) Zipfian κατανομή

Σχήμα 5.4: Χρόνος απόκρισης SnappyData και XDB στα δεδομένα μεγέθους 2GB για w1 workload



Σχήμα 5.5: Χρόνος απόκρισης για το w2 workload των SnappyData και XDB στα δεδομένα μεγέθους 2GB

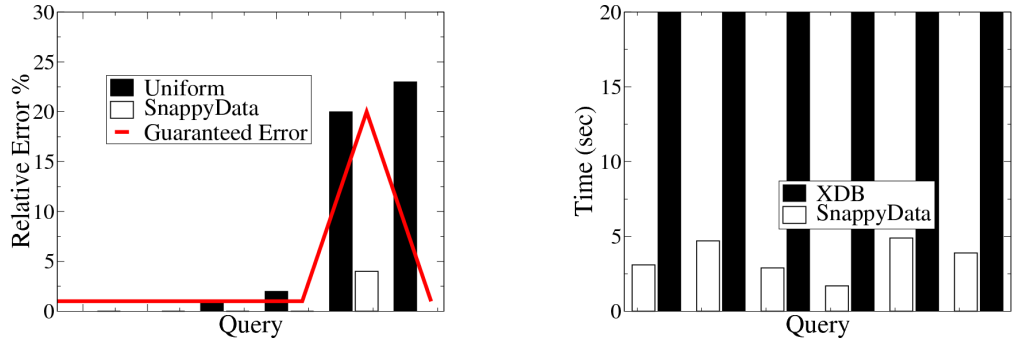
Απάντηση σε γνωστά ερωτήματα

Στα datasets των 2 GB έγιναν δύο ακόμη σύνολα ερωτημάτων για την περίπτωση που τα μελλοντικά ερωτήματα είναι γνωστά και η SnappyData έχει δημιουργήσει δείγματα με κατάλληλο QCS. Το ένα από τα δύο σύνολα απαιτούσε ομαδοποίηση των δεδομένων μέσω της δήλωσης GROUP BY $c2$. Αυτά αποτελούνται από έξι ερωτήματα το κάθε ένα, εκ των οποίων τα τέσσερα είχαν δήλωση WHERE. Συγκεκριμένα τα ερωτήματα φαίνονται στον πίνακα 5.2. Συγκρίναμε το ομοιόμορφο δείγμα και τη SnappyData ως προς την ακρίβεια, καθώς και τις SnappyData και XDB ως προς το χρόνο εκτέλεσης. Η SnappyData χρησιμοποίησε στρωματοποιημένο δείγμα με QCS τις στήλες {c2,c3}.

Στην εικόνα 5.6 παρουσιάζονται τα αποτελέσματα που πήραμε όταν εκτελέσαμε τα ερωτήματα του πίνακα 5.2 πάνω στα δεδομένα που ακολουθούσαν zipfian κατανομή. Διαλέξαμε τη συγκεκριμένη κατανομή διότι στο σχήμα 5.3 παρατηρήσαμε μεγαλύτερες αποκλίσεις από τη

1	$select\ sum(c2)\ from\ T;$
2	$select\ sum(c3)\ from\ T;$
3	$select\ sum(c2)\ from\ T\ where\ c3\ >\ 200\ and\ c3\ <\ 210;$
4	$select\ sum(c2)\ from\ T\ where\ c2\ >\ 500\ and\ c2\ <\ 510;$
5	$select\ sum(c2)\ from\ T\ where\ c3\ >\ 200\ and\ c3\ <\ 210\ and\ c2\ >\ 500\ and\ c2\ <\ 510;$
6	$select\ sum(c2)\ from\ T\ where\ c4\ >\ 700\ and\ c4\ <\ 710\ and\ c5\ >\ 400\ and\ c5\ <\ 410;$

Πίνακας 5.2: Σύνολο γνωστών ερωτημάτων



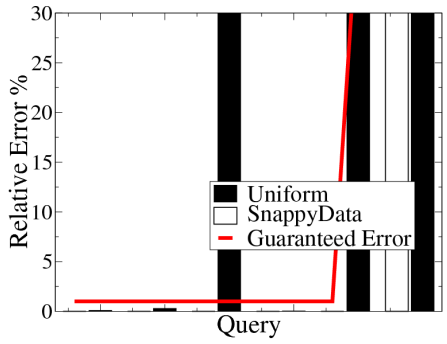
(α′) Ακρίβεια (β′) Χρόνος εκτέλεσης

Σχήμα 5.6: Ερωτήματα πάνω στα δεδομένα με μέγεθος 2GB και κατανομή zipfian

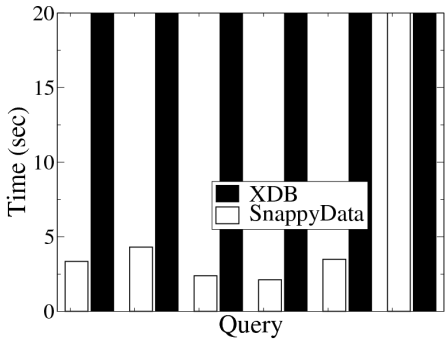
SnappyData σε σύγκριση με την ομοιόμορφη κατανομή. Στο διάγραμμα 5.6α′ συγκρίνουμε την ακρίβεια του ομοιόμορφου δείγματος και της SnappyData. Η τελευταία έδωσε μηδενικά σφάλματα όταν τα ερωτήματα είτε δεν είχαν επιλεκτικότητα, είτε ζητούσαν συγκεκριμένες περιοχές δεδομένων ανάλογα με τις τιμές των c2, c3, πάνω στις οποίες έγινε και η δειγματοληψία. Το τελευταίο ερώτημα, στο οποίο έδωσε σφάλμα περίπου 5%, ήταν πάνω σε δύο τυχαίες στήλες του πίνακα και όχι στις c2, c3. Αντίθετα, το ομοιόμορφο δείγμα έδινε μεγαλύτερα σφάλματα όσο αυξανόταν η επιλεκτικότητα των ερωτημάτων.

Το διάγραμμα 5.6β′ δίνει πληροφορίες για το χρόνο εκτέλεσης που χρειάστηκαν τα δύο συστήματα για τα ερωτήματα του πίνακα 5.2. Ενώ η SnappyData απάντησε γρήγορα σε κάθε ερώτημα, η XDB χρειάστηκε πάνω από 100 δευτερόλεπτα για να δώσει το επιθυμητό σφάλμα.

Στο τελευταίο πείραμα θελήσαμε να αναλύσουμε την απόδοση των δύο συστημάτων όταν τα ερωτήματα έχουν δήλωση GROUP BY. Τα αποτελέσματα που πήραμε φαίνονται στα σχήματα 5.7 και 5.8. Στο διάγραμμα 5.7α′ βλέπουμε ότι η SnappyData συνεχίζει να δίνει μηδενικό σφάλμα όταν γνωρίζει τις στήλες του ερωτήματος. Ωστόσο, στην περίπτωση που το ερώτημα χρησιμοποιεί δύο τυχαίες στήλες στο WHERE, δεν κατάφερε να προσεγγίσει το αποτέλεσμα όσο ελαστικό και αν είναι το σφάλμα που ζητάει ο χρήστης. Το ομοιόμορφο δείγμα από την άλλη δίνει καλά αποτελέσματα όταν τα ερωτήματα δεν έχουν επιλεκτικότητα. Όπως βλέπουμε και στο σχήμα 5.8, ανιχνεύει κάθε ομάδα τιμών. Στα ερωτήματα που είχαμε επιλεκτικότητα μόνο σε μία στήλη, πήραμε διαφορετικά αποτελέσματα από το ομοιόμορφο δείγμα. Στην μία περίπτωση, το σύστημα ανίχνευσε μόνο τις μισές ομάδες τιμών και το αποτέλεσμα δεν



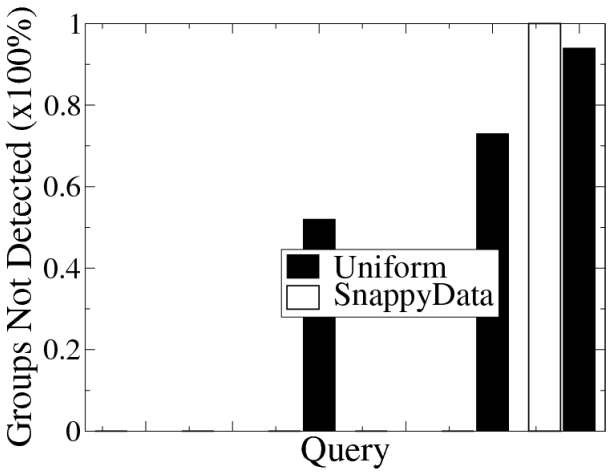
(α′) Ακρίβεια



(β′) Χρόνος εκτέλεσης

Σχήμα 5.7: Ερωτήματα με ομαδοποίηση πάνω στα δεδομένα με μέγεθος 2GB και κατανομή zipfian

είχε μεγάλη ακρίβεια, ενώ στην άλλη ανίχνευσε όλες τις ομάδες τιμών και το προσεγγιστικό αποτέλεσμα ήταν ίσο με το πραγματικό. Όταν η επιλεκτικότητα ήταν πάνω σε δύο στήλες, το ομοιόμορφο δείγμα δεν επέστεψε ικανοποιητικά αποτελέσματα.



Σχήμα 5.8: Ικανότητα των διαφορετικών ομάδων τιμών

Στο σχήμα 5.7β′ συγκρίνουμε την XDB και τη SnappyData ως προς το χρόνο απόκρισης για συγκεκριμένο σφάλμα. Η XDB σε αυτό το σύνολο ερωτημάτων, όπως και στου σχήματος 5.6β′, δεν κατάφερε να επιστρέψει αποτελέσματα σε μικρό χρονικό διάστημα.

Κεφάλαιο 6

Επίλογος

Σύνοψη και συμπεράσματα

Στην εργασία αυτή μελετήσαμε την συμπεριφορά τριών προσεγγιστικών βάσεων δεδομένων και συγκεκριμένα των BlinkDB, SnappyData και XDB. Με τον όρο συμπεριφορά αναφερόμαστε στον χρόνο απόκρισης και στην ακρίβεια με την οποία απαντάν συναθροιστικά SQL ερωτήματα. Η εκτενής πειραματική μας αξιολόγηση κατέληξε στα παρακάτω συμπεράσματα:

Όταν οι ερωτήσεις δεν έχουν φίλτρα και δεν υπάρχει επιλεκτικότητα, όλα τα συστήματα που εξετάσαμε μπορούν να παρέχουν ακριβείς και διαδραστικές απαντήσεις. Επίσης, στην περίπτωση αυτή, η XDB επιτυγχάνει τα καλύτερα αποτελέσματα καθώς μπορεί να επιτύχει αντίστοιχο σφάλμα με την SnappyData σε ίσο ή και μικρότερο χρονικό διάστημα. Επίσης, καθώς η XDB τρέχει σε έναν μόνο κόμβο, αποτελεί πιο συμφέρουσα επιλογή από άποψη κόστους.

Στην περίπτωση που τα ερωτήματα ζητάν αρκετά επιλεκτικές περιοχές του συνόλου δεδομένων ή στην περίπτωση που τα δεδομένα ακολουθούν κάποια αρκετά πολωμένη κατανομή, η ομοιόμορφη δειγματοληψία δεν είναι αρκετή για να παρέχει μια αξιόπιστη προσέγγιση. Για την περίπτωση αυτή πρέπει να επιστρατεύσουμε πιο "έξυπνα" στρωματοποιημένα δείγματα. Όπως είδαμε στα τελευταία σετ πειραμάτων του Κεφαλαίου 5, σε αντίθεση με την XDB, η SnappyData επιτυγχάνει μικρό χρόνο απόκρισης και σφάλμα και για αυτήν την περίπτωση, υπό την προυπόθεση όμως ότι γνωρίζει τις κολώνες του συνόλου δεδομένων που πρόκειται να ερωτηθούν.

Βιβλιογραφία

[1] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, 29–42, 2013.

[2] C. Engle, A. Lupher, R. Xin, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica. Shark: Fast Data Analysis Using Coarse-grained Distributed Memory. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 689–692, 2012.

[3] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. Jordan, S. Madden, B. Mozafari, and I. Stoica. Knowing When You're Wrong: Building Fast and Reliable Approximate Query Processing Systems. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 481–492, 2014.

[4] A. Kleiner, A. Talwalkar, S. Agarwal, I. Stoica, and M. I. Jordan. A general bootstrap performance diagnostic. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 419–427, 2013.

[5] SnappyData. Cluster Architecture. `http://github.com/SnappyDataInc/snappydata/blob/master/docs/architecture/cluster_architecture.md`, 2017..

[6] S. Chaudhuri, G. Das, and V. Narasayya. Optimized stratified sampling for approximate query processing. In *TODS*, 2007.

[7] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 539–550, 2003.

[8] S. Acharya, P. B. Gibbons, and V. Poosala. Congressional samples for approximate answering of group-by queries. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 487–498, 2000.

[9] A. Thusoo, J. S. Sarma, and N. Jain, et al. Hive: a warehousing solution over a map-reduce framework. In *Proceedings of the VLDB Endowment*, 2(2): 1626–1629, 2009.

[10] Apache Hadoop. Mapreduce Project. `http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html`, 2013..

[11] Apache Spark. Lightning-fast cluster computing. `http://spark.apache.org`, 2017..

[12] B. Mozafari, J. Ramnarayan, S. Menon, Y. Mahajan, S. Chakraborty, H. Bhanawat, and K. Bachhav. SnappyData: A Unified Cluster for Streaming, Transactions, and Interactive Analytics. In *CIDR*, 2017.

[13] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. In *Foundations and Trends in Databases*, 4, 2012.

[14] K. Zeng, S. Gao, J. Gu, B. Mozafari, and C. Zaniolo. Abs: a system for scalable approximate queries with accuracy guarantees. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 1067–1070, 2014.

[15] K. Zeng, S. Gao, B. Mozafari, and C. Zaniolo. The analytical bootstrap: a new method for fast error estimation in approximate query processing. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 277–288, 2014.

[16] CliffGuard. A General Framework for Robust and Efficient Database Optimization. `http://www.cliffguard.org`.

[17] K. Zeng, S. Agarwal, A. Dave, M. Armbrust, and I. Stoica. G-OLA: Generalized online aggregation for interactive analysis on big data. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 913–918, 2015.

[18] M. Al-Kateb and B. S. Lee. Stratified Reservoir Sampling over Heterogeneous Data Streams. In *Proceedings of the 22nd international conference on Scientific and statistical database management*, 621–639, 2010.

[19] J. S. Vitter. Random sampling with a reservoir. In *ACM Transactions on Mathematical Software (TOMS)*, 11(1): 37–57, 1985.

[20] E. Liarou et al. MonetDB/DataCell: online analytics in a streaming column-store. In *Proceedings of the VLDB Endowment VLDB Endowment*, 5(12): 1910–1913, 2012.

[21] SnappyData Documentation. Hybrid Cluster Manager. `http://snappydatainc.github.io/snappydata/architecture/hybrid_cluster_manager`.

[22] SnappyData Documentation. Overview of Synopsis Data Engine (SDE). `http://snappydatainc.github.io/snappydata/aqp`.

[23] SnappyData Documentation. Embedded SnappyData Store Mode. `http://snappydatainc.github.io/snappydata/affinity_modes/embedded_mode`.

[24] B. Mozafari, E. Z. Y. Goh, and D. Y. Yoon. CliffGuard: A Principled Framework for Finding Robust Database Designs. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1167–1182, 2015.

[25] Pivotal. Pivotal GemFire. `http://pivotal.io/pivotal-gemfire`.

[26] M. Kornacker, A. Behm, V. Bittorf, T. Bobrovytsky, C. Ching, A. Choi, J. Erickson, M. Grund, D. Hecht, M. Jacobs, I. Joshi, L. Kuff, D. Kumar, A. Leblang, N. Li, I. Pandis, H. Robinson, D. Rorke, S. Rus, J. Russell, D. Tsirogiannis, S. Wanderman-Milne, and M. Yoder. Impala: A modern, open-source SQL engine for Hadoop. In *CIDR*, 2015.

[27] M. Armbrust et al. Spark SQL: Relational data processing in Spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1383–1394, 2015.

[28] MemSQL. `http://www.memsql.com`.

[29] Apache Samza. `http://samza.apache.org`.

[30] R. C. Fernandez, P. R. Pietzuch, J. Kreps, N. Narkhede, J. Rao, J. Koshy, D. Lin, C. Riccomini, G. Wang. Liquid: Unifying Nearline and Offline Big Data Integration. In *CIDR*, 2015.

[31] J. Meehan, N. Tatbul, S. Zdonik, C. Aslantas, U. Cetintemel, J. Du, T. Kraska, S. Madden, D. Maier, A. Pavlo, M. Stonebraker, K. Tufte, H. Wang. S-store: streaming meets transaction processing. In *Proceedings of the VLDB Endowment - Proceedings of the 41st International Conference on Very Large Data Bases, Kohala Coast, Hawaii* 8(13): 2134–2145, 2015.

[32] Amazon. Amazon S3. `http://aws.amazon.com/s3`.

[33] B. Mozafari and N. Niu. A Handbook for Building an Approximate Query Engine. In *IEEE Data Eng. Bull.*, 38: 3–29. 2015.

[34] SnappyData Documentation. Data Ingestion Pipeline. `http://snappydatainc.github.io/snappydata/architecture/data_ingestion_pipeline`.

[35] Wikipedia. Delta encoding. `http://en.wikipedia.org/wiki/Delta_encoding`.

[36] SnappyData Documentation. Local Mode. `http://snappydatainc.github.io/snappydata/affinity_modes/local_mode`.

[37] SnappyData Documentation. SnappyData Smart Connector Mode. `http://snappydatainc.github.io/snappydata/affinity_modes/connector_mode`.

[38] Bradley Efron. Bootstrap Methods: Another Look at the Jackknife. In *The Annals of Statistics*, 7(1), 1979.

[39] Apache Spark. Spark Streaming Programming Guide. `http://spark.apache.org/docs/latest/streaming-programming-guide.html`.

[40] Apache Kafka. A distributed streaming platform. `http://kafka.apache.org`.

[41] Apache Flume. Flume. `http://flume.apache.org`.

[42] Twitter. `http://twitter.com`.

[43] ZeroMQ. Distributed Messaging. `http://zeromq.org`.

[44] Amazon. Amazon Kinesis. `http://aws.amazon.com/kinesis`.

[45] Apache. Apache Zeppelin. `http://zeppelin.apache.org`.

[46] K. Bratbergsengen. Hashing Methods and Relational Algebra Operations. In *VLDB '84 Proceedings of the 10th International Conference on Very Large Data Bases*, 323–333, 1984.

[47] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online Aggregation. In *SIGMOD '97 Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, 171–182, 1997.

[48] T. F. Chan, G. H. Golub, and R. J. LeVeque. Algorithms for computing the sample variance: Analysis and recommendations. In *The American statistician*, 37(3):242–247, 1983.

[49] Probability Inequalities for Sums of Bounded Random Variables. W. Hoeffding. In *Journal of the American Statistical Association*, 58(301): 13–30, 1963.

[50] P. J. Haas. *Hoeffding Inequalities for Join-Selectivity Estimation and Online Aggregation*. IBM Research Report RJ 10040, IBM Almaden Research, 1996.

[51] F. Li, B. Wu, K. Yi, and Z. Zhao. Wander Join: Online Aggregation via Random Walks. In *SIGMOD '16 Proceedings of the 2016 International Conference on Management of Data*, 615–629, 2016.

[52] P. J. Haas. Large-Sample and Deterministic Confidence Intervals for Online Aggregation. In *SSDBM '97 Proceedings of the Ninth International Conference on Scientific and Statistical Database Management*, 51–63, 1997.

[53] Openstack. `http://www.openstack.org`.

[54] F. Olken, D. Rotem. Random sampling from databases: a survey. In *Statistics and Computing*, 5(1): 25–42, 1995

[55] XDB. ApproXimate DB. `http://initialdlab.github.io/XDB`.

Γλωσσάριο

Ελληνικός όρος	Αγγλικός όρος
αφέντης	master
δειγματοληψία	sampling
δεικτοδότηση	indexing
ένωση	join
πακέτο	batch
παράθυρο	window
ροή δεδομένων	data stream
σκλάβος	slave
συνάθροιση	aggregation
σύνολο δεδομένων	dataset
συνεχόμενα ερωτήματα	continuous queries
συστάδα υπολογιστών	cluster
χρησιμοποιημένο λιγότερο πρόσφατα	least recently used
χρονοσφραγίδα	timestamp