Οικονομικό Πανεπιστήμιο Αθηνών
Τμήμα Διοικητικής Επιστήμης & Τεχνολογίας

Εξάμηνο 6ο

ΕΙΔΙΚΑ ΘΕΜΑΤΑ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΛΟΓΙΣΜΙΚΟΥ

Διδάσκων κος Διομήδης Σπινέλλης

ΣΥΝΕΙΣΦΟΡΑ ΣΤΗΝ ΕΦΑΡΜΟΓΗ

Presto

Autocompletion

Auto-correction

ΟΜΑΔΑ ΕΡΓΑΣΙΑΣ:

ΛΕΥΤΕΡΗΣ ΑΝΤΩΝΟΠΟΥΛΟΣ (810017)

Περιεχόμενα

Εισαγωγή

Το Presto είναι ένα Query Engine που αναπτύχθηκε από το Facebook και μπορεί να εκτελεί και να συνδυάζει queries από πολλαπλά data sources. Κάποια από τα datasources που υποστηρίζει είναι:

- HBase
- Hadoop/Hive
- Σχεσιακές Βάσεις Δεδομένων

Αντίθετα με άλλα query engines όπως το Hive που βασίζονται σε map reduce, το Presto λειτουργεί με δική του μηχανή εκτέλεσης που είναι βελτιστοποιημένη για low latency. Αυτό επιτρέπει την εκτέλεση διαδραστικών (interactive) queries και είναι πολύ πιο φιλικό προς τον χρήστη. Ταυτόχρονα είναι όμως συμβατό και με το Hive, καθώς μπορεί να διαβάσει απευθείας από HDFS (Hadoop Distributed File System).

Το Presto δεν αποθηκεύει καθόλου δεδομένα. Τα queries εκτελούνται απευθείας στα data sources όπου ανήκουν και το Presto αναλαμβάνει να συνδυάσει τα δεδομένα ώστε να επιστρέψει αποτελέσματα στον χρήστη. Έτσι αποφεύγεται η ανάγκη για ETL (Extract-Transform-Load) και επίσης τα queries γίνονται πάντα την τελευταία έκδοση των δεδομένων.

Οι συνεισφορές που επιλέξαμε για αυτό το project είναι δύο:

- Προσθήκη δυνατότητας auto-complete εντολών στο command line client αντίστοιχα με τον τρόπο που δουλεύει το Linux shell. Για παράδειγμα αν ο χρήστης γράψει SEL και πατήσει tab στο κέλυφος του Presto, θα συμπληρωθεί αυτόματα η λέξη SELECT.

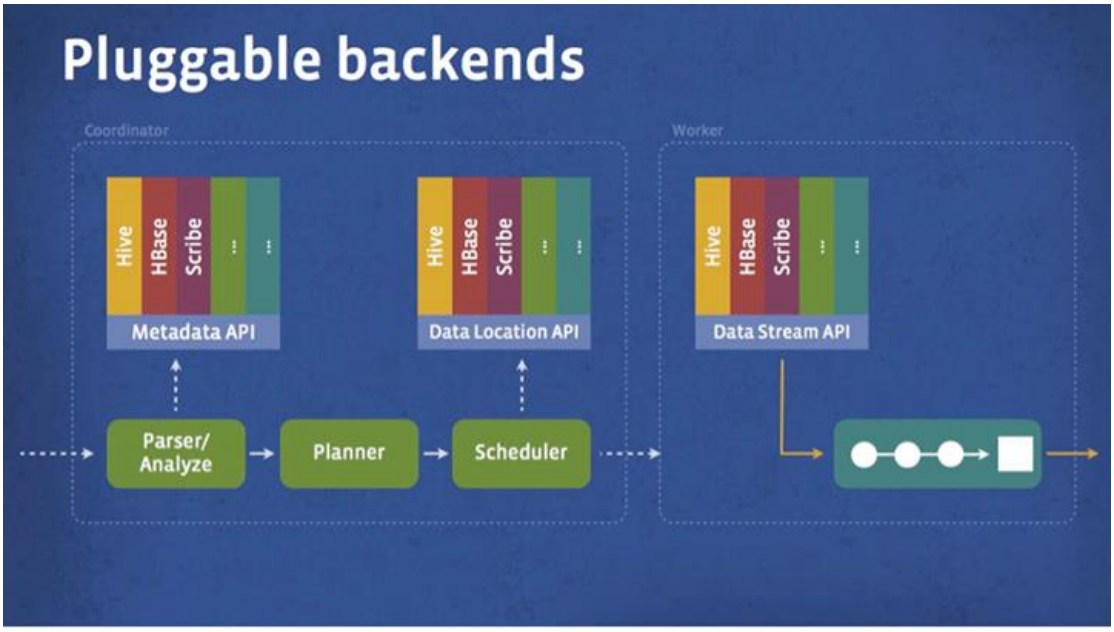
- Προσθήκη auto-suggestions όταν ο χρήστης εισάγει λανθασμένες εντολές. Για παράδειγμα αν ο χρήστης εισάγει "SELEC * FROM foo;" το Presto θα του εμφανίσει μετά το μήνυμα σφάλματος το εξής: "Perhaps you meant 'SELECT * FROM foo'"

Αρχιτεκτονική και χαρακτηριστικά Presto

Το Presto είναι γραμμένο σε Java και αποτελείται από έναν αριθμό modules όπως: *presto-main, presto-tpch, presto-hive, presto-ml, presto-benchmark, presto-hive-cdh4, presto-parser, presto-cassandra, presto-hive-cdh5, presto-raptor, presto-cli, presto-hive-hadoop1, presto-server, presto-client, presto-hive-hadoop2, presto-spi*.

Σαν build tool καθώς και για τη διαχείριση όλων των dependencies έχει χρησιμοποιηθεί το Maven. Σαν γλώσσα ερωτημάτων υποστηρίζει την standard ANSI SQL, η διαχείριση της οποίας γίνεται με τη βοήθεια του Antlr στο module presto-parser.

Ένα άλλο βασικό χαρακτηριστικό του Presto είναι ότι μπορεί να επεκταθεί μέσω plugins και να προσφέρει SQL query δυνατότητες σε οποιουδήποτε είδους dataset. Το μόνο που χρειάζεται είναι το plugin να υλοποιήσει κάποιες διεπαφές για να αντλεί τα μετα-δεδομένα που χρειάζονται, να βρίσκει που βρίσκονται τα δεδομένα και να μπορεί να τα προσπελαύνει. Η αρχιτεκτονική αυτή αποτυπώνεται καλύτερα στην παρακάτω εικόνα:



Ροή εκτέλεσης ερωτημάτων

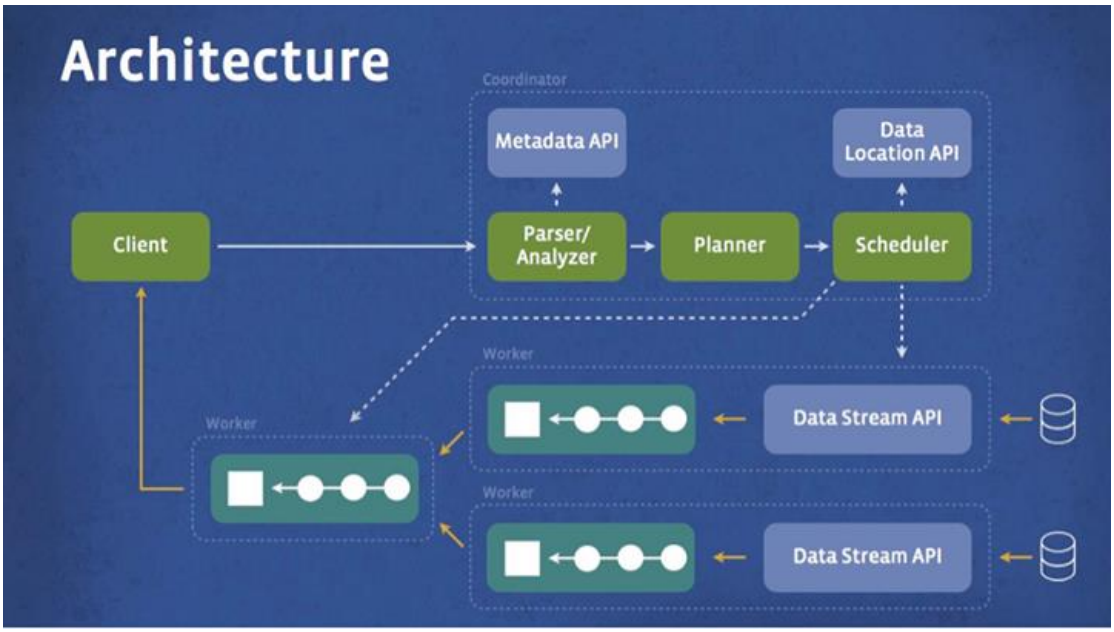
Για να εκτελεί ερωτήματα, το Presto περιέχει δύο βασικά υποσυστήματα: έναν coordinator και έναν ή περισσότερους workers. Ο coordinator αναλαμβάνει να:

- Διαβάζει/αναλύει τα queries (Parse/Analyze)
- Δημιουργεί το πλάνο εκτέλεσης (Plan)
- Αναθέτει δουλειά στους workers και παρακολουθεί την πρόοδο τους (Schedule)

Πιο συγκεκριμένα η ροή εκτέλεσης ενός ερωτήματος στο Presto είναι η εξής:

- Ο πελάτης στέλνει την SQL στον coordinator
- Ο coordinator διαβάζει, αναλύει την SQL και δημιουργεί το πλάνο εκτέλεσης
- Ο coordinator αναθέτει τη δουλειά στους workers βάσει του ποιοί είναι πιο κοντά στα δεδομένα
- Οι workers που τελειώνουν προωθούν τα αποτελέσματα στους workers του επόμενου σταδίου (pipelining)
- Ο πελάτης λαμβάνει το τελικό αποτέλεσμα

Αυτό αποτυπώνεται καλύτερα στην παρακάτω εικόνα:



Η συνεισφορά μας

Σχεδιασμός

Η συνεισφορά μας αφορά κυρίως τον command line client του Presto και ως έναν βαθμό τον Parser. Η αρχική ιδέα για την υλοποίηση και των δύο συνεισφορών ήταν να επαναχρησιμοποιήσουμε με κάποιον τρόπο την υπάρχουσα γραμματική που χρησιμοποιείται από τον parser του Presto. Αυτό τελικά το καταφέραμε μόνο για τη δεύτερη συνεισφορά, καθώς όπως αποδείχτηκε δεν είναι εύκολο να εξαχθούν autocompletion rules[1] από μια γραμματική του Antlr.

Αντί αυτού, επιλέξαμε να περιορίσουμε το autocomplete της πρώτης συνεισφοράς μόνο στην πρώτη εντολή και εξάγαμε τη λίστα εντολών από το σχετικό documentation[2] του Presto. Περισσότερες λεπτομέρειες ακολουθούν στο παρακάτω section.

Υλοποίηση

Ο command line client του Presto χρησιμοποιεί τη βιβλιοθήκη Jline. Μέσω αυτής της βιβλιοθήκης παρέχεται ένα εύρος βασικών δυνατοτήτων όπως ιστορικό εντολών, line editing, απόκρυψη χαρακτήρων (πχ. για passwords) και custom keybindings. Επιπλέον παρέχει ένα interface για tab auto-completition, που χρησιμοποιήσαμε για να προσθέσουμε το δικό μας auto-complete.

Ο command line client του Presto υλοποιείται αρχικά μέσω της κλάσης com.facebook.presto.cli.Console . Η συγκεκριμένη κλάση είναι υπεύθυνη για να εμφανίσει τη γραμμή εντολών, να κάνει initialize το Jline και να εκτελέσει τις εντολές του χρήση αφού πατήσει ENTER. Επιπλέον, το Presto κάνει override την κλάση jline.console.ConsoleReader του Jline με μια δική του κλάση την com.facebook.presto.cli.LineReader .

Η συνεισφορά μας εδώ ήταν να προσθέσουμε τη δυνατότητα εισαγωγής ενός ή περισσότερων autocompleters στον constructor της LineReader, η οποία με τη σειρά της τους περνάει στην ConsoleReader. Επίσης δημιουργήσαμε την κλάση CommandCompleter που υλοποιούσε το interface jline.console.Completor και αναλάμβανε να φέρει τα κατάλληλα suggestions. Μετά όμως από επικοινωνία με την ομάδα του Presto, μας πρότειναν να χρησιμοποιήσουμε μια από τις ήδη υπάρχουσες κλάσεις, την jline.console.completer.StringsCompleter πετυχαίνοντας το ίδιο αποτέλεσμα αλλά χωρίς το code duplication, οπότε ο CommandCompleter αφαιρέθηκε.

[1] http://www.antlr3.org/pipermail/antlr-interest/2008-November/031576.html
[2] http://prestodb.io/docs/current/sql.html

Η δεύτερη συνεισφορά ήταν περισσότερο πολύπλοκη. Για την υλοποίηση της δημιουργήθηκε η κλάση QueryAutoSuggester, η οποία δέχεται ένα query στον constructor και έχει τη μέθοδο getSuggestions που επιστρέφει το κοντινότερο suggestion για αυτό το query. Αυτό γίνεται με τον εξής τρόπο:

1. Το query γίνεται tokenize, δημιουργείται δηλαδή ένας πίνακας με strings που περιέχει κάθε στοιχείο του query
2. Δημιουργείται επίσης ένας πίνακας με όλα τα πιθανά tokens, ο οποίος εξάγεται από το αρχείο Statement.g (που περιλαμβάνει τη γραμματική) με τη βοήθεια των κλάσεων του Antlr. Γι' αυτό το λόγο η QueryAutoSuggester προστέθηκε στο πακέτο com.facebook.presto.parser αντί για το com.facebook.presto.cli .
3. Υπολογίζεται η απόσταση Levenshtein του κάθε token του query από τα υπόλοιπα πιθανά tokens και όσες επιλογές έχουν <2 ταξινομούνται κατά αύξουσα σειρά σε έναν πίνακα που ονομάζεται matches.
4. Για κάθε token του query επιλέγεται το πρώτο token του πίνακα matches και το αντικαθιστά. Έτσι για παράδειγμα το SELEC μπορεί να γίνει SELECT, το FRAM να γίνει FROM κτλ. Για τα tokens που είναι σωστά η απόσταση είναι 0 οπότε ουσιαστικά αντικαθίστανται με τον εαυτό τους.

Για τη συνολική υλοποίηση της συνεισφοράς απαιτήθηκαν οι εξής αλλαγές:

- Προστέθηκε η μέθοδος renderAutoSuggestions στην κλάση com.facebook.presto.cli.Query, η οποία καλείται μετά από την υπάρχουσα renderErrorLocation. Αυτή η μέθοδος δημιουργεί ένα στιγμιότυπο της QueryAutoSuggester και εμφανίζει το suggestion που θα της επιστραφεί.
- Προστέθηκε η κλάση QueryAutoSuggester που αναφέραται παραπάνω.
- Προστέθηκε η κλάση TestQueryAutoSuggester η οποία περιέχει τα κατάλληλα **Unit Tests** για την QueryAutoSuggester. Τα σενάρια που ελέγχει αυτό το Unit Test είναι: εισαγωγή ενός σωστού query (όπου το output θα πρέπει να συμπίπτει με το input), εισαγωγή ενός query με ορθογραιφκό (selec * FROM foo το οποίο θα πρέπει να έχει διορθωθεί στο output) και ένα κενο query (όπου το output θα πρέπει πάλι να είναι κενό).

Τρόπος εργασίας/Συνεισφορά μελών

Για την υλοποίηση της εργασίας δημιουργήσαμε ένα fork[3] του Presto repository στον δικό μας Github λογαριασμό. Στο fork που δημιουργήσαμε υπάρχουν δύο branches:

- master: Εδώ υλοποιήσαμε τη συνεισφορά που αφορούσε την πρώτη συνεισφορά (tab-autocomplete εντολών).
- autosuggest: Εδώ υλοποιήσαμε την δεύτερη συνεισφορά (autosuggestions όταν εισάγεται λανθασμένη εντολή).

Και στις δύο περιπτώσεις δουλέψαμε με τα local copies του repository, ενώ όταν τελειώσαμε κάναμε squash τα τοπικά commits και ανεβάσαμε μόνο ένα-δύο commits με όλες τις αλλαγές σε κάθε branch. Αυτό έγινε γιατί τα guidelines του Presto αναφέρουν ότι κάθε pull request θα πρέπει να περιέχει μόνο ένα commit, ώστε να μην μολύνεται το ιστορικό του repository όταν αυτό γίνει δεκτό.

Σχετικά τώρα με την συνεισφορά του κάθε μέλους της ομάδας, δυστυχώς το δεύτερο μέλος δεν μπόρεσε να ανταποκριθεί φέτος στις απαιτήσεις του μαθήματος. Εγώ όμως επειδή είμαι τελειόφοιτος δεν είχα την ευκαιρία να αφήσω το μάθημα για του χρόνου, με αποτέλεσμα να πάρω την απόφαση να υλοποιήσω την εργασία μόνος μου. Προφανώς επειδή δεν είμαι πολύ καλός γνώσης του προγραμματισμού μίλησα με φίλους και συμφοιτητές που είναι καλύτεροι σε αυτό και με συμβούλεψαν σε διάφορα σημεία στην πορεία της υλοποίησης.

Επικοινωνία με την ομάδα του Presto

Η επικοινωνία με την ομάδα του Presto έγινε κυρίως μέσα από τα pull requests που υποβάλαμε. Για να υποβάλουμε τα pull requests απαιτήθηκε να υπογράψουμε το Contributor License Agreement[4] του Facebook. Συνολικά υποβάλαμε δύο ξεχωριστά pull requests, ένα για κάθε συνεισφορά και τα χειριστήκαμε εντελώς ανεξάρτητα.

Για την πρώτη συνεισφορά μας έγιναν δύο προτάσεις:

1. Να μη δημιουργούμε πολλαπλούς constructors στην κλάση LineReader για να δέχεται διαφορετικό αριθμό από completers αλλά να χρησιμοποιήσουμε το varargs feature της Java. Έτσι αντικαταστήσαμε τον constructor με: LineReader(History history, Completer... completers)
2. Να αξιοποιήσουμε την κλάση StringsCompleter αντί να υλοποιούμε δικό μας Completer. Με αυτό τον τρόπο αποφεύχθηκε ένα αρκετά μεγάλο code duplication.

[3] https://github.com/eantonopoulos/presto
[4] https://code.facebook.com/cla

Για την δεύτερη συνεισφορά αν και κάναμε το σχετικό pull request η ομάδα του Presto δεν μας έχει απαντήσει ακόμα. Όταν απαντήσουν θα συνεχίσουμε υλοποιώντας το feedback ώστε το pull request να γίνει αποδεκτό.

Τα URLs των pull requests είναι τα εξής:

1η συνεισφορά: https://github.com/facebook/presto/pull/1343

2η συνεισφορά: https://github.com/facebook/presto/pull/1357