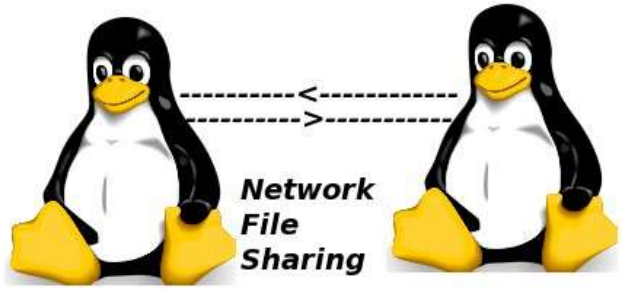
ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ ΣΥΣΤΑΣΕΩΝ SVD ΣΤΗΝ ΠΛΑΤΦΟΡΜΑ SPARK



Βασικά Συστήματα που χρησιμοποιήθηκαν

ΠΕΡΙΓΡΑΦΗ ΠΡΟΒΛΗΜΑΤΟΣ

Επέκταση έτοιμου αλγόριθμου που υλοποιήθηκε για την ανάδειξη του καλύτερου συνεργατικού αλγορίθμου για την πρόβλεψη των αξιολογήσεων χρηστών για ταινίες του Netflix, έτσι ώστε να παραλληλοποιηθεί με τα dataset του MovieLens στο Apache Spark.

1ο Βήμα Υλοποίησης

Μετατροπή του αρχικού προγράμματος σε C++ ώστε να δέχεται dataset στη μορφή του Movie Lens

Οι μετατροπές στο πρόγραμμα έγιναν στις δύο μεθόδους (ProcessFile() και ProcessTest() στις οποίες διαβάζονται τα 2 αρχεία training file και testing file.

- Στην ProcessFile() πλέον διαφέρει ο τρόπος διαβάσματος του training αρχείου.

- Στην ProcessTest() με αντίστοιχο τρόπο γίνεται το διάβασμα του testing αρχείου. Το γράψιμο των προβλέψεων στο αρχείο των αποτελεσμάτων γινόταν με τον ίδιο τρόπο.

- Η ReadNumber, ParseInt και ParseFloat αφαιρέθηκαν.

2ο Βήμα Υλοποίησης - Μετατροπή προγράμματος C++ σε Java ακολουθιακό πρόγραμμα

- Τα structs (Movie, Customer και Data) μετατράπηκαν σε κλάσεις.

- Η μέθοδος CalcMetrics() δεν χρειάζεται πλέον καθώς η λειτουργικότητά της απορροφίθηκε από την CalcFeatures() για την απλούστευση του προγράμματος και την καλύτερη κατανόησή του

    1. Η επεξεργασία κάθε γραμμής στο training set και η αύξηση των στατιστικών πλέον ενσωματώθηκε στη CalcFeatures() (καλώντας την CheckUpdateMovies(movieId, rating)).

    1. Στο αρχικό πρόγραμμα έγινε χρήση Maps έτσι ώστε να μπορεί να κάνει Map κάθε φορά που γινόταν μετάφραση των ids σε συμπαγή πίνακα ευρετηρίων (index array).

1. Η πρόσθεση των customers γινόταν χρησιμοποιώντας ένα map για την επαναρίθμηση των ids για τους δείκτες του array (index array)
    1. Πλέον γίνεται με την CheckUpdateCustomers(custId, rating)

Ορισμένες διαφορές στους τύπους δεδομένων όπως ορισμένα strings , με prefix το L, το οποίο δεν είναι τίποτα άλλο παρά wchar_t literal για extended set από χαρακτήρες.

3ο Βήμα Υλοποίησης

Εύρεση και στήσιμο κάποιου κοινόχρηστου file system έτσι ώστε να μπορεί να διαμοιραστεί σε όλους τους executors ένα κοινό directory για να διαβάζουν και να γράφουν παράλληλα.

- **Χρησιμοποιήθηκαν 2 εικονικές μηχανές UBUNTU 14.04 με 4 GB RAM η κάθε μία .**

- **Το κάθε μηχάνημα εκτός από τον βασικό NAT adapter, έχει και από ένα bridged adapter και host-only adapter.**

  - **όταν είχα πρόσβαση στο διαδίκτυο -> bridged adapter**

  - **όταν δεν είχα πρόσβαση στο διαδίκτυο -> host only adapter (vboxnet0)**

4ο Βήμα Υλοποίησης

Εξοικείωση με τη πλατφόρμα του Apache Spark

- Διάβασμα σχετικών εγγράφων (Documentation) από site Apache Spark (
  - Spark Overview, Quick Start
  - Spark Programming Guide
  - API Docs (Java)
  - Spark Examples (Java)
  - Cluster Mode Overview
  - Spark Standalone Mode κτλπ

- Εξοικίωση με Scala Shell με εκτέλεση αλληλεπιδραστικών παραδειγμάτων.

- Διαφορές ανάμεσα στις δυνατότητες Scala και Java.

- Κατανόηση της παραλληλοποίησης μέσω Spark (RDD, Broadcast variables και Accumulator variables και tranformations/actions σε RDDs).

- Επιλογή κατάλληλων δομών δεδομένων σε Spark Java στις οποίες θα μετατραπούν οι δομές δεδομένων του ακολουθιακού κώδικα Java.

5ο Βήμα Υλοποίησης

Παραλληλοποίηση Java ακολουθιακού προγράμματος σε Java παράλληλο πρόγραμμα στο Spark

Σημεία αλγορίθμου που εντόπισα δυνατότητα να επιτευχθεί απευθείας παραλληλοποίηση ήταν τρία:

1. Διάβασμα του αρχείου δεδομένων. -> ProcessFile()

1.Ο υπολογισμός των prediction ratings για το test file με υπολογισμένους ήδη τους πίνακες customer και movie features. -> Αρχικοποίηση m_aCustFeatures_BRDCST και m_aMovieFeatures_BRDCST με πίνακες m_aCustFeatures και m_aMovieFeatures για να γίνουν διαθέσιμοι στους executors.

1.Το γράψιμο του αρχείου αποτελεσμάτων. -> ProcessTest()

6ο Βήμα Υλοποίησης

Σημεία αδυναμίας παραλληλοποίησης αλγορίθμου

- Δεν υπάρχει παραλληλοποιήσιμη R/W δομή που να υποστηρίζει δισδιάστατους πίνακες.

- Η CalcFeatures() δεν γινόταν να παραλληλοποιηθεί λόγω της φύσης του αλγορίθμου αλλά και των περιορισμών του Spark .

Π.χ. σε ένα RDD δεν μπορώ να κάνω access ένα συγκεκριμένο στοιχείο για να ενημερώσω μόνο αυτό.

Η CalcFeatures παρέμεινε σειριακή στο παράλληλο πρόγραμμα.
Ότι δεν περιλαμβάνεται στο SC εκτελείται μόνο από driver.

- Αποτελείται από 3 ένθετους βρόγχους, με ελάχιστο αριθμό εκτέλεσης των εντολών στον εσωτερικό βρόγχο 10^5 *120*64 = 768.000.000

- Στον 2ο βρόγχο με επαναληπτικά loops εκπαιδεύονται τα feature σε ολόκληρο το dataset και μόλις έχει σημειωθεί επαρκής πρόοδος προχωράει στο επόμενο, συνεπώς δεν είναι προκαθορισμένος ο αριθμός των εκτελέσεων πράγμα το οποίο δυσκολεύει την παραλληλοποίησή του.
- Ο υπολογισμός των feature χρειάζεται ανά πάσα στιγμή πρόσβαση στην cache του προηγούμενου feature για τον υπολογισμό του τωρινού.

- **Υπάρχει ανάγκη να γίνεται Index κάθε φορά συγκεκριμένο στοιχείο σε ένα RDD.**

- **Η Calcfeature καλεί τη PredictRating και χρειάζεται την Τρέχουσα cached value για τους υπολογισμούς.**

- **Το Spark με τις δομές που υποστηρίζει (RDD) δεν επιτρέπουν να γίνεται index κάποιο συγκεκριμένο στοιχείο.**

- **Μετά την PredictRating και τον υπολογισμό του error γίνονται cache off οι τιμές από τα παλιά features.**

- **Αυτό δημιουργεί ένα άλλο εμπόδιο στο τέλος του 2ου βρόγχου υπάρχει ένας επιλέον βρόγχος.**

- **Κάνει cache off τα παλιά αποτελέσματα και για να το πετύχει αυτό χρειάζεται πρόσβαση στη παλιά τιμή της cache που πλέον δεν είναι διαθέσιμη.**

7ο Βήμα Υλοποίησης
Απόπειρες παραλληλοποίησης

Πακέτο amplab:spark-indexedrdd:0.3
IndexedRDD αποτελεί μια δομή αποθηκευτικού χώρου key-value για το Spark.
Όμως είναι συμβατό μόνο με πολύ παλαιά έκδοση, 1.5.0 του Spark και χρησιμοποιεί απαρχαιωμένα features εκείνης της έκδοσης τα οποία δεν υποστηρίζονται πλέον από το 1.6.1 .

- **βιβλιοθήκη Mlib**

Παρέχει έτοιμες συναρτήσεις για Dimensionality Reduction όπως SVD,
Παρέχει 4 εκδοχές distributed matrix με σημαντικούς περιορισμούς κάθε μια.

- **DataFrames και SQL**

τα οποία επίσης έχουν πρόβλημα στην ενημέρωση των δεδομένων λόγω του ότι βασίζονται σε immutable RDDs.

Kryo Registrator
για βελτιστοποίηση, κατά την διάρκεια της ανάπτυξης λειτουργούσε αλλά στο τέλος δεν πέτυχε.

- **anonymous function**

η οποία όμως δυστυχώς δεν επιτρέπει την εκτέλεση transformation σε RDD μέσα από τη κλάση.

•Δοκίμασα να φτιάξω δικιά μου MatrixRow.

•Έφτιαξα την κλάση MatrixRowAccumulatorParam η οποία θα καθορίζει τον τρόπο που θα συσσωρεύονται οι τιμές ενός MatrixRow.

•Με την inner κλάση InnerFunctionClass3 χρησιμοποιώντας τις 2 προηγούμενες κλάσεις MatrixRow και MatrixRowAccumulatorParam παίρνω από τους πίνακες movie και customer την αντίστοιχη γραμμή και τα αποθηκεύω σε MatrixRow .

Παίρνοντας τα αντίστοιχα στοιχεία που έχουν τα feature του customer και του movie υπολογίζω την βαθμολογία για όλα τα feature σε επαναληπτικό βρόγχο ο οποίος εκτελεί απευθείας τον κώδικα της PredictRating καθώς δεν μπορούσε να κάνει αλλιώς πρόσβαση στα πεδία της μεθόδου παρόλο που βρισκόταν στην enclosing κλάση.

Τελικά επιστρέφεται όλη η πληροφορία με τα αποτελέσματα και τα 2 MatrixRow (που έχουν τα feature) σε ένα tuple 3 το οποίο θα αποτελεί value για ένα tuple 2 του οποίου το key θα είναι το πολυπόθητο index .

❖ Ουσιαστικά Έγινε απόπειρα δημιουργίας αυτόσχεδιας δομής ικανή να παραλληλοποιηθεί από το Spark αλλά ταυτόχρονα να μπορεί γίνει INDEX.

Στην CalculateSQ_InnerClass γίνεται προσπάθεια να ξεπεραστούν το πρόβλημα με την προηγούμενη Inner κλάση.

Δεν γίνεται να ενημερώσω μια τιμή σε ένα MatrixRow Accumulator
Κατασκευάζεται ένα άδειο MatrixRow με μηδενικά έτσι ώστε να ενημερωθεί ένα item τη φορά.

Προσθέτω αυτό το MatrixRow στον MatrixRow Accumulator.

Το οποίο όμως εκτός από το γεγονός ότι είναι βαριά σαν διαδικασία επίσης δεν έβγαζε σωστά αποτελέσματα.

- **Spark_CalcFeatures**

Αρχικά γίνεται η δημιουργία του RDD από Data και τα 2 MatrixRows με τα feature των ταινιών και πελατών με τα οποία τροφοδοτούνται οι inner Κλάσεις.

Η δημιουργία 2 broadcast variables(read only, cached σε κάθε executor) και 2 accumulator(only added παράλληλη εκτέλεση) ώστε τα MatrixRows να έχουν πρόσβαση στις προηγούμενες τιμές και να υπολογίζουν τα αποτελέσματα παράλληλα.

Τελικά γίνεται ενημέρωση των maps με τα features των accumulated MatrixRows.

- **Δυστυχώς απέτυχε. Οι τιμές ήταν λάθος.**

- **Τέλος στην spark_PredictRating αυτό που αλλάζει είναι οι πίνακες με τα features για τον υπολογισμό των βαθμολογιών πλέον γίνεται χρήση broadcast πινάκων με τα features έτσι ώστε να γίνεται παράλληλα ο υπολογισμός.**

MOVIE LENS 100K			
ENGINE CONSTRUCTION	0 ms	0 ms	5 136 ms
LOAD HISTORY	1 083 ms	0 ms	3 021 ms
CALCULATION FEATURE	41 246 ms	6 018 ms	18 281 ms
PROCESSING TEST	3 069 ms	0 ms	1 018 ms
TOTAL	**45 398 ms**	**6 018 ms**	**27 456 ms**
MOVIE LENS 1M			
ENGINE CONSTRUCTION	0 ms	0 ms	4 006 ms
LOAD HISTORY	2 033 ms	3 087 ms	8 012 ms
CALCULATION FEATURE	468 103 ms	170 324 ms	146 043 ms
PROCESSING TEST	3 008 ms	1 073 ms	7 105 ms
TOTAL	**473 144 ms**	**174 484 ms**	**165 166 ms**
MOVIE LENS 3M			
ENGINE CONSTRUCTION	0 ms	0 ms	8 007 ms
LOAD HISTORY	10 051 ms	7 083 ms	25 087 ms
CALCULATION FEATURE	1 297 037 ms	355 141 ms	374 032 ms
PROCESSING TEST	138 021 ms	3 067 ms	22 062 ms
TOTAL	**1 445 109 ms**	**365 291 ms**	**429 012 ms**
MOVIE LENS 5M			
ENGINE CONSTRUCTION	1 004 ms	0 ms	6 032 ms
LOAD HISTORY	15 043 ms	35 057 ms	51 071 ms
CALCULATION FEATURE	2 128 068 ms	1 173 314 ms	645 056 ms
PROCESSING TEST	12 057 ms	15 137 ms	48 053 ms
TOTAL	**2 156 172 ms**	**1 223 508 ms**	**750 212 ms**

8ο Βήμα Υλοποίησης

Συμπεράσματα χρόνων εκτέλεσης προγραμμάτων

- ✓ Καταρχήν τα εξαγόμενα αποτελέσματα στα prediction files είναι ακριβώς ίδια.

Η εκτέλεση των προγραμμάτων με το μικρό dataset των 100K έδειξε ότι το σειριακό πρόγραμμα σε Java εκτελείται πιο γρήγορα από το παράλληλο.

Ακολούθησαν πολλά πειράματα με μεγαλύτερα dataset έτσι ώστε να γίνει κλιμάκωση των προγραμμάτων.

Με το dataset του 1 εκατομμυρίου φάνηκε ότι το παράλληλο πρόγραμμα εκτελείται πιο γρήγορα .

Το ίδιο παρατηρήθηκε και με το μεγαλύτερο dataset που κατάφερα να τρέξω αυτό των 5 εκατομμυρίων καθώς με μεγαλύτερα dataset δεν φτάνει η RAM στον driver για να το εκτελέσει, εκεί βέβαια η διαφορά στον συνολικό χρόνο εκτέλεσης αυξήθηκε αρκετά.

Όμως δοκιμάζοντας dataset ενδιάμεσο αυτό των 3 εκατομμυρίων διαπιστώνω ότι το σειριακό πρόγραμμα σε αυτή την περίπτωση εκτελείται γρηγορότερα, από το οποίο προκύπτει ότι δεν υπάρχει consistency.

Μελλοντικές Επεκτάσεις

Επέκταση των δύο προγραμμάτων (σειριακό και παράλληλο) σε Java έτσι ώστε να διαβάζουν dataset στην μορφή του Netflix.