MapReduce: Simplified Data Processing on Large Clusters

Περίληψη. Αυτή η εργασία έχει ως σκοπό να κάνει μία περίληψη του άρθρου [DeGh04] και της νεότερης έκδοσής του [DeGh08]. Τα εν λόγω άρθρα παρουσιάζουν το *MapReduce*, ένα προγραμματιστικό μοντέλο που επιτρέπει την επεξεργασία και την παραγωγή μεγάλων ποσοτήτων δεδομένων. Οι χρήστες ορίζουν μία συνάρτηση *map* η οποία επεξεργάζεται ένα ζεύγος key/value ώστε να παράγει ένα ενδιάμεσο ζεύγος key/value, και μία συνάρτηση *reduce* η οποία συγχωνεύει όλα τα ενδιάμεσα values που σχετίζονται με τα ίδια ενδιάμεσα keys. Προγράμματα τα οποία γράφονται με αυτό τον τρόπο, εκτελούνται σε ένα μεγάλο cluster αυτόματα και παράλληλα.

1 Εισαγωγή

Η ανάγκη για επεξεργασία πάρα πολύ μεγάλων όγκων δεδομένων ταυτόχρονα σε κατανεμημένα συστήματα όπου υπάρχουν μαζικές παράλληλες μηχανές και ο τρόπος διαχείρισης των πιθανών σφαλμάτων, απασχόλησε για μεγάλο χρονικό διάστημα τους ερευνητές. Στόχος ήταν η εύρεση μίας εύκολης λύσης, όπου θα υπήρχε εκτέλεση υψηλού επιπέδου εφαρμογών και αποδέσμευση των προγραμματιστών από πολύπλοκες υλοποιήσεις.

Η περίπτωση της μέτρησης της συχνότητα πρόσβασης μιας URL, που είναι δυνατό προσπελαύνεται από χιλιάδες χρήστες του διαδικτύου, ή της αυτόματης ταξινόμησης εγγραφών σε κατανεμημένο σύστημα που δέχεται συνεχώς αιτήσεις από διάφορους clients, ή του ανεστραμμένου ευρετηρίου μιας μηχανής αναζήτησης, η οποία εμπλουτίζεται συνεχώς με νέα έγγραφα, είναι ζητήματα για τα οποία η λύση της σειριακής επεξεργασίας θα ήταν απαγορευτική.

Προηγούμενες προσεγγίσεις προσπάθησαν να παρουσιάσουν συστήματα για παράλληλο υπολογισμό, περιορισμένων όμως δυνατοτήτων και χωρίς ανοχή σφαλμάτων, αφού αφήναν τις λεπτομέρειες διαχείρισης των αποτυχιών της μηχανής στους προγραμματιστές. Από την άλλη πλευρά, οι Dean και Ghemawat στα [DeGh04] και [DeGh08] προτείνουν παράλληλες αναγνώσεις για την επεξεργασία όγκου δεδομένων σε clusters από υπολογιστές γενικής χρήσης, μέσω του *MapReduce*. Η ουσιαστική συμβολή των συγγραφέων αφορά:

(α) τη δημιουργία ενός framework που τρέχει σε μεγάλους clusters από εμπορικές μηχανές, υψηλής επεκτασιμότητας, και που δύναται να ανακάμψει από εσφαλμένες καταστάσεις,

(β) την ευκολία και ευελιξία της χρήσης του συστήματος από τους προγραμματιστές, αφού όλες οι πολύπλοκες διαδικασίες γίνονται στο παρασκήνιο και οι χρήστες προσαρμόζουν τις συναρτήσεις map() και reduce() σύμφωνα με τις εκάστοτε ανάγκες τους.

2. Προσέγγιση και αξιολόγηση

2.1 Παρουσίαση του MapReduce

Τρόπος λειτουργίας

Η κεντρική ιδέα των άρθρων είναι η παρουσίαση ενός συστήματος που θα μπορεί να επεξεργάζεται παράλληλα και αποτελεσματικά πάρα πολύ μεγάλο όγκο δεδομένων που ολοένα και αυξάνεται, προσπαθώντας να αυξήσει την *τοπικότητα* της όλης διαδικασίας. Ο τρόπος λειτουργίας του MapReduce έτσι όπως παρουσιάζεται μέσα από τα [DeGh04] και [DeGh08] είναι ο ακόλουθος:

(i) Η MapReduce βιβλιοθήκη στο πρόγραμμα του χρήστη, χωρίζει τα δεδομένα εισόδου σε *M* κομμάτια των 64MB, και στη συνέχεια, αντίγραφα του προγράμματος επεξεργασίας αρχίζουν να τρέχουν σε μία συστάδα υπολογιστών.

(ii) Ένα από αυτά τα αντίγραφα ονομάζεται *αφέντης*, ενώ τα υπόλοιπα *εργάτες*. Ο αφέντης είναι υπεύθυνος σε πρώτη φάση για την απόδοση των *M* (*map*) και *R* (*reduce*) εργασιών σε καθένα από τους εργάτες.

(iii) Ένας εργάτης που του έχει ανατεθεί μία map εργασία, διαβάζει τα δεδομένα από το τμήμα που του αναλογεί και εξάγει ενδιάμεσα ζεύγη του τύπου *key/value*, τα οποία αποθηκεύονται στη μνήμη.

(iv) Περιοδικά τα ενδιάμεσα ζεύγη key/value αποθηκεύονται στον τοπικό δίσκο, όπου μέσω συνάρτησης διαχωρισμού μοιράζονται σε R τμήματα. Οι τοποθεσίες αυτών των τμημάτων δίνονται στον αφέντη και αυτός με τη σειρά του τις δίνει στους R reduce εργάτες.

(v) Ο reduce εργάτης διαβάζει επαναληπτικά την ταξινομημένη λίστα ζευγών key/value και για κάθε ενδιάμεσο key περνάει στη συνάρτηση reduce το key και τις ενδιάμεσες τιμές που σχετίζονται με αυτό. Η έξοδος της συνάρτησης reduce επισυνάπτεται σε ένα αρχείο εξόδου.

(vi) Όταν οι παραπάνω διαδικασίες ολοκληρωθούν, ο αφέντης αφυπνίζει/ενημερώνει το πρόγραμμα του χρήστη.

Από την παραπάνω περιγραφή, παρατηρούμε ότι ο αφέντης χρειάζεται να πάρει *O(M+R)* προγραμματιστικές αποφάσεις και διατηρεί *O(M*R)* καταστάσεις στη μνήμη.

Ανοχή σφαλμάτων

Σε έναν σύστημα όπου υπάρχουν εκατοντάδες ή χιλιάδες μηχανές να τρέχουν παράλληλα, οι πιθανότητες να προκύψουν σφάλματα κατά την εκτέλεση είναι ένα λογικό σενάριο. Στα [DeGh04] και [DeGh08] αναλύονται οι περιπτώσεις αποτυχιών και αναφέρεται ο τρόπος που το MapReduce τις διαχειρίζεται.

Αποτυχία εργάτη: Ο αφέντης επικοινωνεί περιοδικά με κάθε εργάτη. Αν ο εργάτης δε απαντήσει μέσα σε συγκεκριμένο χρόνο θεωρεί ότι ο εργάτης απέτυχε. Σε αυτό το σημείο διακρίνουμε τρεις περιπτώσεις: (α)ο εργάτης προτού αποτύχει είχε ολοκληρώσει μία map εργασία, τότε η εργασία ακυρώνεται και επαναλαμβάνεται από κάποιον άλλο εργάτη(τα αποτελέσματά της βρίσκονταν αποθηκευμένα στον τοπικό δίσκο του εργάτη που απέτυχε), (β)ο εργάτης προτού αποτύχει είχε ολοκληρώσει την reduce εργασία, τότε η εργασία δεν εκτελείται ξανά διότι τα αποτελέσματά της έχουν ήδη αποθηκευτεί στο global σύστημα, (γ)ο εργάτης τη στιγμή που απέτυχε εκτελούσε map ή reduce εργασία, τότε η εργασία ακυρώνεται και επαναλαμβάνεται από την αρχή από κάποιον άλλο εργάτη. Οι reduce εργασίες που βρίσκονται σε εξέλιξη ενημερώνονται κατάλληλα σε περίπτωση που η map εργασία από την οποία προμηθεύονται τα δεδομένα εκτελεσθεί από την αρχή από κάποιον άλλο εργάτη.

Αποτυχία αφέντη: Σε αυτή την περίπτωση όλες οι υπολογιστικές διαδικασίες του MapReduce ακυρώνονται - αν και η αποτυχία του κόμβου αφέντη είναι σπάνιο φαινόμενο.

Επιπλέον, η σημασία της παρουσίας σφαλμάτων έχει να κάνει και με τον τρόπο καθορισμού των map και reduce συναρτήσεων από το χρήστη, δηλαδή αν αυτές θα είναι *ντετερμινιστικές* ως προς τις εισαγόμενες τιμές τους ή *μη-ντετερμινιστικές*. Στην πρώτη περίπτωση, η κατανεμημένη εφαρμογή παράγει συνεχώς το ίδιο αποτέλεσμα όπως θα παραγόταν από μία μη-λανθασμένη σειριακή εκτέλεση του συνολικού προγράμματος. Αντίθετα, στη δεύτερη περίπτωση, το αποτέλεσμα εξαρτάται από κάθε φορά από τα δεδομένα.

Εργασίες Backup

Μία συνηθισμένη αιτία που προκαλεί αύξηση του συνολικού χρόνου εκτέλεσης είναι ο “straggler”: μία μηχανή που χρειάζεται ασυνήθιστα μεγάλο χρόνο για να ολοκληρώσει μία από τις λίγες τελευταίες map ή reduce εργασίες υπολογισμού της. Η λύση που προτείνεται στα [DeGh04] και [DeGh08] είναι η εξής: όταν μια λειτουργία MapReduce κοντεύει στο τέλος της, ο αφέντης προγραμματίζει *backup* εκτελέσεις των υπολοίπων εργασιών που βρίσκονται σε εξέλιξη. Η εργασία μαρκάρεται ως ολοκληρωμένη όταν η αρχική ή η backup εκτέλεση ολοκληρωθεί. Μετρήσεις κατέγραψαν έως και 44% μείωση του χρόνου ολοκλήρωσης της εργασίας χρησιμοποιώντας την backup τεχνική.

2.2 Επεκτάσεις

Οι επεκτάσεις στο ήδη υπάρχον σύστημα προσφέρονται ως επιπλέον λύσεις στις ανάγκες των προγραμματιστών, και τους δίνουν τις δυνατότητες:

- τροποποίησης της συνάρτησης διαχωρισμού,
- εύκολης δημιουργίας ταξινομημένου εξερχόμενου αρχείου ανά τμήμα,
- προσδιορισμού προαιρετικής *Combiner* συνάρτησης που κάνει μερική συγχώνευση των δεδομένων πριν την αποστολή τους μέσω δικτύου,
- υποστήριξης για ανάγνωση δεδομένων διαφόρων μορφών,
- παραγωγής βοηθητικών αρχείων ως επιπρόσθετα outputs των map ή/και reduce λειτουργιών,
- παράλειψης εγγραφών που μπορεί να προκαλέσουν προβλήματα στη λειτουργία του συστήματος,
- τοπικής εκτέλεσης όλων των εργασιών μιας MapReduce λειτουργία για ευκολία debugging, profiling και μικρής έκτασης testing,
- αναφοράς κατάστασης συστήματος μέσω status pages,
- καταμέτρησης γεγονότων προκληθέντων από διάφορα συμβάντα.

2.3 Πειράματα και αποτελέσματα

Για την πειραματική αξιολόγηση του συστήματος έγιναν δύο υπολογισμοί (*αναζήτησης* και *ταξινόμησης*) που εκτελέστηκαν σε ένα μεγάλο cluster μηχανών με περίπου 1TB δεδομένα.

Αναζήτηση: Το *grep* πρόγραμμα σαρώνει 10^{10} 100-byte εγγραφές, αναζητώντας ένα σχετικά σπάνιο τριών-χαρακτήρων pattern. Το input χωρίζεται σχεδόν σε 64MB κομμάτια και ολόκληρο το output τοποθετείται σε ένα αρχείο. Ο ρυθμός μεταφοράς δεδομένων σε σχέση με το χρόνο *αυξάνεται σταδιακά* κατά τη διάρκεια των map εργασιών και μέχρι την ολοκλήρωσή τους, ενώ έπειτα *μειώνεται γρήγορα* μέχρι που μηδενίζεται. Η ολοκλήρωση του υπολογισμού απαιτεί περίπου 150sec.

Ταξινόμηση: Το *sort* πρόγραμμα ταξινομεί 10^{10} 100-byte εγγραφές. Μία map συνάρτηση εξάγει ένα 10-byte ταξινομημένο key και εκπέμπει το πραγματικό κείμενο ως το ενδιάμεσο ζεύγος key/value. Γίνεται χρήση μίας *Identity* συνάρτησης κατά τη reduce λειτουργία, η οποία περνάει τα ενδιάμεσα μη-αλλαγμένα ζεύγη key/values ως output ζεύγη key/values. Το τελικό ταξινομημένο output γράφεται σε ένα σετ 2-τρόπων κατασκευασμένα GFS αρχεία. Ο input ρυθμός είναι υψηλότερος από τον ρυθμό ανακατάταξης και τον output ρυθμό λόγω της βελτιστοποίησης της τοπικότητας, ενώ ο ρυθμός ανακατάταξης είναι υψηλότερος από τον output ρυθμό γιατί η output φάση γράφει δύο αντίγραφα των ταξινομημένων δεδομένων.

3. Κριτική επισκόπηση

3.1 Δυνατά σημεία

Τα δύο άρθρα αποτελούν μία καινοτομία στο χώρο της παράλληλης επεξεργασίας πολύ μεγάλων όγκων δεδομένων, με πιο δυνατά σημεία τα εξής:

1. Επιτρέπεται η εκτέλεση λειτουργιών map/reduce με χρήση κατανεμημένης επεξεργασίας, δεδομένου ότι η κάθε λειτουργία map που γίνεται είναι ανεξάρτητη από άλλες, όλο το mapping μπορεί να γίνει παράλληλα.
2. Μπορεί να γίνεται επεξεργασία πολύ μεγαλύτερου όγκου δεδομένων από αυτόν που ένας commodity εξυπηρετητής μπορεί να επεξεργαστεί.
3. Σε περίπτωση που ένας εξυπηρετητής είναι αργός, ή αδυνατεί να επεξεργαστεί τα δεδομένα που του ανατέθηκαν, γίνεται rescheduling της εργασίας του αυτόματα.

3.2 Αδύνατα σημεία

Εν συνεχεία, αναφέρουμε κάποια αδύνατα σημεία που εντοπίσαμε στα δύο άρθρα:

1. Το MapReduce θα μπορούσε να χαρακτηριστεί λιγότερο αποδοτικό σε σχέση με άλλους αλγορίθμους που είναι περισσότερο γραμμικοί.
2. Η χρησιμότητα του συστήματος μοιάζει να εξαρτάται από ένα cluster όπως αυτός της Google, και δεν είναι σαφές το πόσο εύκολα μεταφέρσιμη είναι η βιβλιοθήκη ώστε να μπορέσει να συνυπάρξει με άλλους τύπους multi-processor συστημάτων.
3. Στην αξιολόγηση του συστήματος δίνονται μετρήσεις χρόνου εκτέλεσης για πολλούς διαφορετικούς αλγόριθμους που έχουν γραφτεί χρησιμοποιώντας το MapReduce, αλλά δεν υπάρχει τιμή αναφοράς που να συγκρίνει αυτές τις μετρήσεις.

Αναφορές

[DeGh04] Jeffrey Dean, Sanjay Ghemawat: MapReduce: Simplified Data Processing on Large Clusters *OSDI*2004

[DeGh08] Jeffrey Dean, Sanjay Ghemawat: MapReduce: Simplified Data Processing on Large Clusters *Commun.ACM*51(1),2008