ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
Προηγμένα Πληροφορικά Συστήματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρονοπρογραμματισμός Ροών Δεδομένων στο Νέφος

Herald Kllapi

Επιβλέποντες: **Γιάννης Ιωαννίδης,** Καθηγητής

ΑΘΗΝΑ

Νοέμβριος 2010

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρονοπρογραμματισμός Ροών Δεδομένων στο Νέφος

Herald Q. Kllapi
Α.Μ.: 945

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Γιάννης Ιωαννίδης,** Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Γιάννης Ιωαννίδης,** Καθηγητής
Αλέξης Δελής, Καθηγητής

Νοέμβριος 2010

ΠΕΡΙΛΗΨΗ

Ο χρονοπρογραμματισμός ροών δεδομένων στο νέφος είναι μια πολύπλοκη διαδικασία γεμάτη προκλήσεις. Ουσιαστικά, είναι ένα πρόβλημα βελτιστοποίησης, το οποίο διαφέρει σε δύο σημεία από την κλασσική βελτιστοποίηση στις βάσεις δεδομένων: ο χώρος αναζήτησης είναι πολύ πλούσιος δεδομένου του νέου περιβάλλοντος και το κριτήριο βελτιστοποίησης είναι τουλάχιστον δύο διαστάσεων, με το χρηματικό κόστος να είναι εξίσου σημαντικό με τον χρόνο ολοκλήρωσης. Σε αυτήν την εργασία μελετάμε τον χρονοπρογραμματισμό ροών δεδομένων που περιλαμβάνουν αυθαίρετους τελεστές που επεξεργάζονται δεδομένα στο πλαίσιο τριών προβλημάτων: 1) ελαχιστοποίηση του χρόνου εκτέλεσης δεδομένου ενός χρηματικού ορίου, 2) ελαχιστοποίηση χρηματικού κόστους δεδομένου χρονικού ορίου, και 3) εύρεση συμβιβασμών μεταξύ χρόνου και χρήματος χωρίς περιορισμούς. Διατυπώνουμε τα προβλήματα και παρουσιάζουμε ένα πλαίσιο βελτιστοποίησης το οποίο είναι προσεγγιστικό και εκμεταλλεύεται την ελαστικότητα των πόρων του νέφους. Για να διερευνήσουμε την αποτελεσματικότητα της προσέγγισής μας, υλοποιήσαμε το προτεινόμενο πλαίσιο σε ένα πρωτότυπο σύστημα και ενσωματώσαμε διάφορους άπληστους, πιθανοτικούς, και εξαντλητικής αναζήτησης αλγορίθμους. Τέλος, μέσα από διάφορα πειράματα που κάναμε με το πρωτότυπο ελαστικό βελτιστοποιητή χρησιμοποιώντας διάφορες επιστημονικές και συνθετικές ροές δεδομένων, έχουμε εντοπίσει αρκετά ενδιαφέροντα χαρακτηριστικά του χώρου των εναλλακτικών χρονοπρογραμμάτων καθώς και τα πλεονεκτήματα και μειονεκτήματα των διαφόρων αλγορίθμων αναζήτησης. Τα συνολικά αποτελέσματα είναι ελπιδοφόρα και δείχνουν την αποτελεσματικότητα της προσέγγισής μας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Επεξεργασία πολύπλοκων ροών δεδομένων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: αλγόριθμοι, σχεδίαση, οικονομικά μοντέλα, πειραματισμός, απόδοση

ABSTRACT

Scheduling data processing workflows (dataflows) on the cloud is a very complex and challenging task. It is essentially an optimization problem, very similar to query optimization, that is characteristically different from traditional problems in two aspects: Its space of alternative schedules is very rich, due to various optimization opportunities that cloud computing offers; its optimization criterion is at least two-dimensional, with monetary cost of using the cloud being at least as important as query completion time. In this paper, we study scheduling of dataflows that involve arbitrary data processing operators in the context of three different problems: 1) minimize completion time given a fixed budget, 2) minimize monetary cost given a deadline, and 3) find trade-offs between completion time and monetary cost without any a-priori constraints. We formulate these problems and present an approximate optimization framework to address them that uses resource elasticity in the cloud. To investigate the effectiveness of our approach, we incorporate the devised framework into a prototype system for dataflow evaluation and instantiate it with several greedy, probabilistic, and exhaustive search algorithms. Finally, through several experiments that we have conducted with the prototype elastic optimizer on numerous scientific and synthetic dataflows, we identify several interesting general characteristics of the space of alternative schedules as well as the advantages and disadvantages of the various search algorithms. The overall results are quite promising and indicate the effectiveness of our approach.

SUBJECT AREA: Complex dataflow processing

KEYWORDS: algorithms, design, economics, experimentation, performance

Στους γονείς μου Gjenovefa και Qako
στον αδερφό μου Lazart
και στην Blerina

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον καθηγητή Γιάννη Ιωαννίδη για την καθοδήγησή του και τις πολύτιμες συμβουλές του. Επίσης, θα ήθελα να ευχαριστήσω τον Μανώλη Τσαγκάρη ο οποίος είναι από τους καλύτερους ανθρώπους και επιστήμονες που έχω συνεργαστεί και την Εύα Σιταρίδη για την υπέροχη συνεργασία μας. Ευχαριστώ πολύ τον Κωνσταντίνο Τσακαλώζο για της συζητήσεις που κάναμε και της πολύτιμες παρατηρήσεις του. Ακόμα θα ήθελα να ευχαριστήσω τους φίλους μου, Άννα, Βιβή, Εύα, Νίσα, Ολυμπία, Βαγγέλη, Δημήτρη, Θωμά, Κωνσταντίνο, Νικόλα, Σωκράτη, και πολλούς άλλους. Τέλος, ευχαριστώ τους γονείς μου που με στήριξαν και με συμβούλεψαν όλα αυτά τα χρόνια και ακόμα συνεχίζουν να το κάνουν.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΑΛΓΟΡΙΘΜΩΝ

ΠΡΟΛΟΓΟΣ

Η εργασία αυτή έγινε στο Πανεπιστήμιο Αθηνών, στο τμήμα Πληροφορικής και Τηλεπικοινωνιών. Η αρχική ιδέα ξεκίνησε όταν εργαζόμουν στην ομάδα του κυρίου Ιωαννίδη σε ένα Ευρωπαϊκό έργο, το Health-e-Child, σε συνεργασία με την Εύα Σιταρίδη και τον Μανόλη Τσαγκάρη. Με τον Μανόλη και την Εύα είχαμε αναλάβει να φτιάξουμε το σύστημα κατανεμημένης επεξεργασίας (MPE – Medical Processing Engine). Την ίδια περίοδο περίπου, κάποιο υποσύνολο της ομάδας του MaDgIK (www.madgik.di.uoa.gr), υπό την επίβλεψη του κ. Ιωαννίδη, ξεκινήσαμε το ADP Project, σκοπός του οποίου είναι η κατανεμημένη επεξεργασία ερωτημάτων τα οποία περιέχουν κώδικα που δίνεται από τους χρήστες υπό την μορφή τελεστών (User Defined Code – Operators). Τα μέλη της ομάδας σε αλφαβητική σειρά είναι Γιάννης Ιωαννίδης, Γιώργος Κακαλέτρης, Herald Kllapi, Γιώργος Παπανίκος, Φραγκίσκος Πεντάρης, Παύλος Πολύδωρας, Εύα Σιταρίδη, Βασίλης Στούμπος, και Μανόλης Τσαγκάρης. Το ADP είναι ένα ενεργό έργο στην ομάδα MaDgIK και σκοπεύουμε να παραμένει ενεργό και να επεκταθεί και εκτός ομάδας στο μέλλον.

1. ΕΙΣΑΓΩΓΗ

Η πολύπλοκη ανάκτηση και επεξεργασία δεδομένων είναι χαρακτηριστικό πολλών εφαρμογών και συνδυάζει την έννοια ερωτημάτων και αναζήτησης, ανάκτησης και φιλτραρίσματος δεδομένων, μετασχηματισμού και ανάλυσης δεδομένων, και άλλους χειρισμούς δεδομένων. Πλούσια ερωτήματα σαν και αυτά τυπικά περιγράφονται από γράφους επεξεργασίας δεδομένων οι οποίοι έχουν αυθαίρετους τελεστές στους κόμβους και ακμές οι οποίες περιγράφουν τις σχέσεις τελεστών παραγωγού καταναλωτή. Για παράδειγμα, υποθέστε ότι ένας χρήστης θέλει να βρει τα ονόματα και τις φωτογραφίες των συγγραφέων στο *ACM*. Αυτό θα μπορούσε να εκφραστεί με το παρακάτω *SQL* ερώτημα:

SELECT UNIQUE auth.name, img.image
FROM AuthorDB auth, ImageBD img
WHERE auth.pub="ACM" ***AND*** auth.name=*face*(img.image)

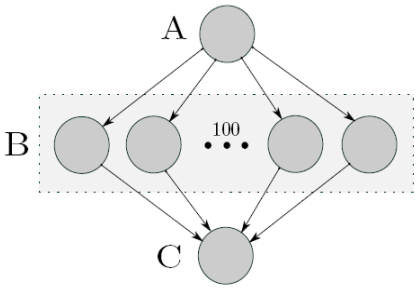
Η συνάρτηση *face()* ανιχνεύει το πρόσωπο ενός ατόμου σε μια συγκριμένη φωτογραφία την οποία δέχεται ως παράμετρο και επιστρέφει το όνομα του. Αυτό το *SQL* ερώτημα βελτιστοποιείτε [18] και μετατρέπετε σε ένα πλάνο εκτέλεσης, το οποίο ουσιαστικά είναι ένας γράφος ροής δεδομένων. Σε ένα κατανεμημένο περιβάλλων, εκτός των άλλων, ο βελτιστοποιητής του συστήματος θα πρέπει να αποφασίσει που θα εκτελέσει τους κόμβους του γράφου. Αυτό το στάδιο της βελτιστοποίησης ονομάζετε χρονοπρογραμματισμός (scheduling). Ο χρονοπρογραμματισμός ενός γράφου ροής δεδομένων σε ένα σύνολο από μηχανές είναι ένα γνωστό δύσκολο NP-πλήρες πρόβλημα ακόμα και στην πιο απλή μορφή του [13][25]. Παραδοσιακά, το μόνο κριτήριο βελτιστοποίησης είναι ο χρόνος τερματισμού του ερωτήματος και πολλοί ευριστικοί αλγόριθμοι έχουν προταθεί για να λύσουν το πρόβλημα αυτό [19].

Τα τελευταία χρόνια, ο υπολογισμός στο νέφος έχει προσελκύσει την προσοχή της επιστημονικής κοινότητας [1]. Χάρη στη εικονικοποίηση (virtualization), έχει εξελιχθεί από μια υποδομή με συστοιχίες υπολογιστών (cluster) η οποία έχει σχεδιαστεί για συγκεκριμένο σκοπό, στο πλέγμα (grid) [10], και τέλος στο νέφος. Το νέφος προσφέρει υπηρεσίες τριών κατηγοριών. Ανάλογα με τις απαιτήσεις της εφαρμογής, οι υποδομές (IaaS), οι πλατφόρμες (PaaS), ή το λογισμικό (SaaS) μπορούν να προσφερθούν σαν υπηρεσίες [12].

Μια από τις διαφορές ανάμεσα σε συστοιχίες υπολογιστών και τα νέφη είναι το μοντέλο κόστους των πόρων που προσφέρουν. Οι συστοιχίες υπολογιστών έχουν ένα σταθερό κόστος στην αρχή και σχετικά μικρό κόστος λειτουργίας. Σε αντίθεση, τα νέφη χαρακτηρίζονται από ελαστικότητα και προσφέρουν στους χρήστες την δυνατότητα να δανείζονται πόρους για όσο χρόνο τους χρειάζονται [2]. Μαζί με το γεγονός ότι δεν έχουν αρχικό κόστος, αυτή η δυνατότητα έχει πολλά πλεονεκτήματα συγκρινόμενη με τις συστοιχίες υπολογιστών. Η δυνατότητα να χρησιμοποιούνται πόροι οι οποίοι είναι διαθέσιμοι κατ απαίτηση, αλλάζει τη σχεδίαση των αλγορίθμων και των συστημάτων. Η εκτέλεση των ροών δεδομένων μπορεί πλέων να είναι ελαστική, προσφέροντας διάφορες επιλογές μεταξύ του χρόνου εκτέλεσης και του χρηματικού κόστους, κάνοντας το πρόβλημα της βελτιστοποίησης στο νέφος τουλάχιστον δύο διαστάσεων.

Για να δείξουμε την πολυπλοκότητα του προβλήματος, χρησιμοποιούμε τον γράφο του Σχήματος 1. Οι κόμβοι αναπαριστούν διαδικασίες επεξεργασίας δεδομένων (τελεστές) και οι ακμές αναπαριστούν σχέσεις παραγωγού-καταναλωτή. Υποθέστε ότι ο χρόνος εκτέλεσης του A και του C είναι 1 ώρα και οι χρόνοι εκτέλεσης καθενός από τα B είναι 10 λεπτά. Επίσης υποθέστε ότι το κβάντο με το οποίο χρεώνει το νέφος είναι 1 ώρα (ακόμα και αν χρησιμοποιήσουμε 10 λεπτά, θα χρεωθούμε 1 ώρα). Θεωρείστε δυο

διαφορετικά χρονοπρογράμματα για αυτή την ροή δεδομένων. ***Χρονοπρόγραμμα 1***: εκτέλεση όλων των τελεστών σε ίδιο εικονικό μηχάνημα (virtual machine). Στην αρχή εκτελείτε ο A, ύστερα τα 100 B, και στο τέλος ο C. Ο χρόνος που χρειάζεται για να εκτελεστεί όλος ο γράφος είναι το άθροισμα των χρόνων εκτέλεσης των τελεστών: 60+10·100+60=1120 λεπτά ή 18.6 ώρες. Μιας και χρησιμοποιείτε μόνο ένα μηχάνημα, το χρηματικό κόστος για αυτό το χρονοπρόγραμμα αντιστοιχεί σε 19 ώρες χρήσης των πόρων.



Εικόνα 1: Μια απλή ροή δεδομένων με τρείς τύπους τελεστών A, B, και C.

Χρονοπρόγραμμα 2: εκτέλεση όλων των τελεστών σε διαφορετικά μηχανήματα. Στην αρχή, ο A εκτελείτε και τα δεδομένα που παράγει μεταφέρονται στα μηχανήματα που θα εκτελέσουν τους B. Όλοι οι B εκτελούνται παράλληλα και όταν ολοκληρωθούν, τα δεδομένα μεταφέρονται σε ένα μηχάνημα, το οποίο εκτελεί τον C. Υποθέτοντας ότι το κόστος μεταφοράς των δεδομένων είναι αμελητέο, ο χρόνος ολοκλήρωσης είναι μόνο 60+10+60=130 λεπτά ή 2 ώρες και 10 λεπτά. Όμως, το κόστος εκτέλεσης είναι πολύ μεγάλο και αντιστοιχεί σε 102 ώρες χρήσης των πόρων. Όλες οι μηχανήματο-ώρες θα χρεωθούν ως ολόκληρες. Το χρονοπρόγραμμα 2 θα τρέξει περίπου 9 φορές πιο γρήγορα από το χρονοπρόγραμμα 1, αλλά θα κοστίσει 5 φορές πιο πολύ. ■

Το πλήθος καθώς και το χρονικό και χρηματικό κόστος των διαθέσιμων χρονοπρογραμμάτων εξαρτούνται από πολλές παραμέτρους, όπως τα χαρακτηριστικά του γράφου (χρόνος εκτέλεσης των τελεστών, μέγεθος των δεδομένων που παράγουν, κλπ.), την τιμολόγηση του νέφους (το μέγεθος και το κόστος του κβάντου), την ταχύτητα του δικτύου, και άλλα. Η επιλογή του βαθμού του παραλληλισμού που θα χρησιμοποιηθεί, ή ισοδύναμα, η επιλογή του βέλτιστου συμβιβασμού μεταξύ χρόνου και χρήματος, εξαρτάται από τις ανάγκες του εκάστοτε χρήστη. Για παράδειγμα, ένας χρήστης μπορεί να έχει χρηματικούς περιορισμούς αλλά να μην νοιάζεται για τον χρόνο ολοκλήρωσης, ένας άλλος χρήστης μπορεί να έχει χρονικούς περιορισμούς αλλά να μην νοιάζεται για το χρηματικό κόστος, και τέλος, ένας τρίτος χρήστης μπορεί να μην έχει εκ των προτέρων περιορισμούς αλλά να θέλει να επιλέξει μόνος του το κατάλληλο συμβιβασμό μεταξύ χρόνου και χρήματος, αφού πρώτα έχει δει όλες τις δυνατές επιλογές.

Σε αυτήν την εργασία, κάνουμε τις εξής συνεισφορές:

1) Μοντελοποιούμε τον χρονοπρογραμματισμό ροών δεδομένων στο νέφος ως ένα δισδιάστατο πρόβλημα βελτιστοποίησης μεταξύ χρόνου και χρήματος (το οποίο κυρίως αγνοείτε μέχρι τώρα).
2) Μελετάμε τον χώρο των εναλλακτικών χρονοπρογραμμάτων το οποίο διαμορφώνετε από το μοντέλο και ερευνάμε τους συμβιβασμούς μεταξύ χρόνου και χρήματος για διαφορετικές ροές δεδομένων και νέφη.
3) Προτείνουμε διάφορους άπληστους, πιθανοτικούς, και εξαντλητικής αναζήτησης αλγορίθμους για την αναζήτηση του χώρου των διαφορετικών χρονοπρογραμμάτων.

Παρουσιάζουμε τα αποτελέσματα διάφορων πειραμάτων τα οποία έδειξαν ενδιαφέροντα χαρακτηριστικά των συμβιβασμών και των αλγορίθμων όσον αφορά την δυνατότητά τους να ανακαλύψουν αυτούς τους συμβιβασμούς. Τα υπόλοιπα κεφάλαια έχουν οργανωθεί ως εξής. Στα κεφάλαια 2, 3, 4, και 5, διατυπώνουμε το πρόβλημα του χρονοπρογραμματισμού ροών δεδομένων στο πλαίσιο του συστήματος ADP [30], το οποίο είναι ένα πρωτότυπο σύστημα για εκτέλεση ροών δεδομένων. Στο κεφάλαιο 6, παρουσιάζουμε την προσέγγιση μας μαζί με τους αλγορίθμους βελτιστοποίησης που χρησιμοποιούμε. Στο κεφάλαιο 7, παρουσιάζουμε τα αποτελέσματα της πειραματικής αξιολόγησης. Στο κεφάλαιο 8, παρουσιάζουμε την σχετική βιβλιογραφία και τέλος, στο κεφάλαιο 9, παρουσιάζουμε κάποιες κατευθύνσεις της μελλοντικής μας δουλειάς.

2. ΔΙΑΤΥΠΩΣΗ ΠΡΟΒΛΗΜΑΤΟΣ

Οι αιτήσεις των χρηστών μετατρέπονται σε ερωτήματα σε κάποια γλώσσα υψηλού επιπέδου (π.χ. SQL ή Hive [28]). Η βελτιστοποίηση ενός ερωτήματος θα μπορούσε να γίνει σε ένα γιγαντιαίο βήμα, εξετάζοντας όλα τα δυνατά πλάνα εκτέλεσης που θα μπορούσαν να απαντήσουν το αρχικό ερώτημα και επιλέγοντας αυτό που είναι βέλτιστο και πληροί τους απαιτούμενους περιορισμούς. Εναλλακτικά, δεδομένου του μεγέθους του χώρου των λύσεων, η βελτιστοποίηση θα μπορούσε να γίνει σε πολλαπλά μικρότερα βήματα, κάθε ένα από αυτά λειτουργώντας σε κάποιο επίπεδο και κάνοντας υποθέσεις για τα επίπεδα που βρίσκονται κάτω από αυτό. Αυτό είναι σε αναλογία με την κλασική βελτιστοποίηση και εκτέλεση ερωτημάτων στις βάσεις δεδομένων αλλά με τις εξής διαφορές. Οι τελεστές μπορούν να εκπροσωπούν αυθαίρετες ενέργειες σε δεδομένα έχοντας άγνωστη σημασιολογία, αλγεβρικές ιδιότητες, και χαρακτηριστικά επιδόσεις, και δεν περιορίζονται ώστε να προέρχεται από ένα γνωστό σταθερό σύνολο τελεστών (π.χ., από σχεσιακή άλγεβρα). Το τι είναι βέλτιστο μπορεί να υπόκειται στην ποιότητα της υπηρεσίας που προσφέρετε (QoS) ή άλλων περιορισμών και μπορεί να βασίζεται σε πολλαπλά διαφορετικά κριτήρια, π.χ., το χρηματικό κόστος των πόρων, φρεσκάδα των δεδομένων, κλπ., και όχι μόνο αποκλειστικά στην απόδοση. Οι πόροι που διατίθενται για την εκτέλεση ενός γράφου ροών δεδομένων είναι ευέλικτοι και διαθέσιμοι κατ απαίτηση και δεν είναι δεδομένοι εκ των προτέρων. Οι διαφορές αυτές καθιστούν την βελτιστοποίηση ροών δεδομένων ένα νέα πρόβλημα γεμάτο προκλήσεις. Επίσης αναδύεται η ανάγκη για νέους μηχανισμούς εκτέλεσης που δεν είναι συνήθως διαθέσιμοι. Η Διαδικασία βελτιστοποίησης που ορίζουμε, η οποία έχει παρουσιαστεί στο [30], έχει τρία διαφορετικά επίπεδα αφαίρεσης τα οποία περιγράφονται παρακάτω.

2.1 Γράφοι Τελεστών

Οι κόμβοι των γράφων αυτών είναι τελεστές που επεξεργάζονται δεδομένα και οι κατευθυνόμενες ακμές τους είναι αλληλεπιδράσεις τελεστών με τη μορφή παραγωγού-καταναλωτή. Οι τελεστές ενσωματώνουν αλγορίθμους επεξεργασίας δεδομένων και μπορεί να προέρχονται από τους τελικούς χρήστες. Σε αυτό το επίπεδο αφαίρεσης, μεγάλη σημασία έχουν οι αλγεβρικές ιδιότητες των τελεστών. Αυτές περιλαμβάνουν τους τυπικούς αλγεβρικούς μετασχηματισμούς, για παράδειγμα, ***αντιμεταθετικότητα***, ***προσεταιριστικότητα***, ***επιμεριστικότητα***, ***αποσύνθεση***, δηλαδή, οι τελεστές να μπορούν να είναι αφαιρέσεις ολόκληρων υπογράφων οι οποίοι περιλαμβάνουν συνθέσεις, συναθροίσεις, και άλλες αλληλεπιδράσεις πιο συγκεκριμένων τελεστών, και ***διαμερίσεις***, δηλαδή, τελεστές που επιδέχονται αντιγραφή και παράλληλη επεξεργασία, με κάθε αντίγραφο να επεξεργάζεται κάποιο τμήμα της αρχικής εισόδου, σε συνδυασμό με κάποια προ-και μετά-επεξεργασία.

2.2 Συγκεκριμένοι Γράφοι Τελεστών

Αυτοί οι γράφοι είναι παρόμοιοι με τους γράφους τελεστών, αλλά οι κόμβοι τους είναι συγκεκριμένοι τελεστές (concrete operator), δηλαδή, κομμάτια λογισμικού που υλοποιούν τους αφαιρετικούς τελεστές με κάποιο συγκεκριμένο τρόπο και φέρουν όλες τις απαραίτητες πληροφορίες για την εκτέλεσή τους. Σε αυτό το επίπεδο, ενδιαφέρων έχουν οι διαθέσιμες υλοποιήσεις ενός τελεστή. Τυπικά, μπορεί να υπάρχουν πολλαπλές υλοποιήσεις ενός τελεστή. Για παράδειγμα, μια υλοποίηση που χρησιμοποίει λίγη μνήμη αλλά είναι αργή και μια άλλη που χρειάζεται πολύ μνήμη αλλά είναι πολύ γρήγορη, μια πολυνηματική ή μονονηματική έκδοση, ή δύο εντελώς διαφορετικές, αλλά λογικά ισοδύναμες υλοποιήσεις του ιδίου τελεστή. Ένα παράδειγμα από τις βάσεις δεδομένων είναι ο τελεστής ζεύξης (join) ο οποίος έχει πολλαπλές υλοποιήσεις: ο

αλγόριθμος εμφωλευμένων βρόχων έχει χαμηλή κατανάλωση μνήμης αλλά και μεγάλο χρόνο εκτέλεσης ενώ ο αλγόριθμος ζεύξης κατακερματισμού (hash-join) χρησιμοποιεί περισσότερη μνήμη (για να δημιουργήσει το ευρετήριο κατακερματισμού) αλλά έχει μικρό χρόνο εκτέλεσης.

2.3 Πλάνα Εκτέλεσης

Αυτοί είναι παρόμοιοι με τους συγκεκριμένους γράφους τελεστών, αλλά οι κόμβοι τους είναι συγκεκριμένοι τελεστές των οποίων οι πόροι έχουν δεσμευτεί και όλες οι παράμετροι έχουν τεθεί. Σε αυτό το επίπεδο, γίνετε η ανάθεση των τελεστών σε μηχανές. Η κρίσιμη πληροφορία είναι οι πόροι που απαιτούνται για την εκτέλεση των τελεστών, π.χ., η χρήση του επεξεργαστή και της μνήμης. Σε αυτή την εργασία παρουσιάζουμε τη μοντελοποίηση και μια μεθοδολογία για να λυθεί το πρόβλημα για αυτό το στάδιο της βελτιστοποίησης. Τα στατιστικά στοιχεία που απαιτούνται υποθέτουμε ότι είναι γνωστά, και υπολογίζονται είτε με μαθηματικούς τύπους ή με βάση ιστορικά δεδομένα που έχουν κρατηθεί από προηγούμενες εκτελέσεις.

Μια ροή δεδομένων αναπαρίσταται ως ένας ακυκλικός κατευθυνόμενος γράφος (Directed Acyclic Graph - DAG) ***γράφος(τελεστές, ροές)***. Οι κόμβοι (τελεστές) αντιστοιχούν σε αυθαίρετους συγκεκριμένους τελεστές και οι ακμές (ροές) αντιστοιχούν στα δεδομένα τα οποία μεταφέρονται μεταξύ τους. Ένας τελεστής μοντελοποιείται ως ***τελεστής(χρόνος, επεξεργαστής (CPU), μνήμη, συμπεριφορά)***, δηλαδή, με τον χρόνο εκτέλεσης του, την μέση χρήση του επεξεργαστή του μηχανήματος που μετράται ως ποσοστό της χρήσης του επεξεργαστή όταν εκτελείτε σε απομόνωση (χωρίς την παρουσία άλλων τελεστών), την μέγιστη χρήση της μνήμης που απαιτείται για την αποτελεσματική εκτέλεση του, και τέλος, τη συμπεριφορά του που είναι μια τιμή που είναι ίση με *σωλήνωση* (*pipeline*) ή *αποθήκευση-και-προώθηση* (*store-and-forward*). Αν η συμπεριφορά είναι αποθήκευση-και-προώθηση (Α&Π), όλες οι είσοδοι του τελεστή πρέπει να είναι διαθέσιμες πριν την εκτέλεσή του. Αν είναι σωλήνωση (ΣΛ), η εκτέλεση του μπορεί να αρχίσει όταν κάποιο μέρος της εισόδου είναι διαθέσιμη. Δύο χαρακτηριστικά παραδείγματα από τις βάσεις δεδομένων είναι ο τελεστής *ταξινόμησης* και ο τελεστής *επιλογής*: ο τελεστής ταξινόμησης είναι Α&Π και της επιλογής είναι ΣΛ. Υποθέτουμε ότι οι τελεστές έχουν ομοιόμορφη κατανάλωση πόρων κατά τη διάρκεια της εκτέλεσής τους (η χρήση του επεξεργαστή, της μνήμης, και η συμπεριφορά δεν αλλάζουν). Μια ροή ανάμεσα σε δύο τελεστές, παραγωγού και καταναλωτή, μοντελοποιείτε ως ***ροή (παραγωγός, καταναλωτή, δεδομένα)***, δηλαδή, οι δύο τελεστές που συνδέονται και το μέγεθος των δεδομένων που παράγει ο παραγωγός.

Ο υποδοχέας (container) είναι η αφαίρεση της μηχανής, η οποία μπορεί να είναι εικονική ή φυσική, αποκρύπτοντας τους πόρους που προσφέρονται από την υφιστάμενη υποδομή. Οι υποδοχείς είναι υπεύθυνοι για την εποπτεία των τελεστών και την παροχή του απαραίτητου περιβάλλοντος για την εκτέλεσή τους. Ένας υποδοχέας περιγράφεται από την επεξεργαστική ισχύ του, την μνήμη του, και την ταχύτητα του δίκτυο: ***υποδοχέας(επεξεργαστής, μνήμη, δίκτυο)***. Ένα χρονοπρόγραμμα S_G του γράφου ροής δεδομένων G είναι μια ανάθεση της τελεστών στους υποδοχείς ***χρονοπρόγραμμα(αναθέσεις)***. Μια μεμονωμένη ανάθεση τελεστή μοντελοποιείται ως εξής: ***ανάθεση(τελεστής, υποδοχέας, αρχή, τέλος)***, όπου *αρχή* και *τέλος* είναι η χρονική στιγμή έναρξης και λήξης του τελεστή όταν εκτελείται παρουσία άλλων τελεστών. Με $t(S_G)$ και $m(S_G)$ συμβολίζουμε τον χρόνο εκτέλεσης και το χρηματικό κόστος αντίστοιχα του χρονοπρογράμματος S_G του γράφου ροής δεδομένων G. Οι δύο επόμενες ενότητες καθορίζουν τον τρόπο με τον οποίο μοντελοποιούμε τον χρόνο και το χρηματικό κόστος ένας χρονοπρογράμματος. Το νέφος είναι ένας πάροχος εικονικών μηχανών. Μοντελοποιούμε μόνο το υπολογιστικό κομμάτι του νέφους και όχι την αποθήκευση των δεδομένων. Αυτό το κάνουμε γιατί μια συγκεκριμένη ροή δεδομένων

G θα διαβάσει και θα γράψει τα ίδια δεδομένα ανεξαρτήτως του χρονοπρογράμματος που θα επιλέγει για να εκτελεστεί. Έτσι, το κόστος του διαβάσματος και του γραψίματος των δεδομένων είναι το ίδιο. Στο υπόλοιπο κομμάτι της εργασίας, χρησιμοποιούμε την '.' για να υποδηλώσουμε ένα χαρακτηριστικό ενός αντικειμένου, π.χ., ο χρόνος εκτέλεσης ενός τελεστή Α συμβολίζεται με Α.χρόνος. Οι αδιάφορες τιμές σημειώνονται με "_", π.χ., ανάθεση(τελεστής, υποδοχέας, _, _).

3. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΧΡΟΝΟΥ

Για να υπολογιστεί ο χρόνος ολοκλήρωσης ενός χρονοπρογράμματος το οποίο εκτελείτε σε ένα σύνολο από μηχανές, διάφορες πτυχές της εκτέλεσης πρέπει να μοντελοποιηθούν. Αυτό είναι ζητούμενο για κάθε κατανεμημένο σύστημα και η συγκεκριμένη προσέγγισή μας δεν εξαρτάτε από την πλατφόρμα η οποία θα επιλεγεί για να εκτελεστούν οι γράφοι ροών δεδομένων. Σε αυτό το κεφάλαιο παρουσιάζουμε την προσέγγισή που ακολουθήσαμε στην εργασία μας, το οποίο είναι εμπνευσμένο από παλιότερες εργασίες.

Υπάρχουν δύο τύποι περιορισμών, αυτοί που συνάγονται από τον γράφο ροής δεδομένων και εκείνοι που επιβάλλονται από το περιβάλλον εκτέλεσης. Οι περιορισμοί που συνάγονται από τον γράφο είναι λόγο της εξάρτησης μεταξύ των τελεστών αλλά και από τη φύση των ίδιων των τελεστών. Ένας τελεστής Α&Π δεν μπορεί να εκτελεστεί μέχρι όλες οι είσοδοί του να γίνουν διαθέσιμες, ενώ ένας τελεστής ΣΛ πρέπει να περιμένει για όλες τις ροές που παράγεται από τελεστές Α&Π. Περιγράφουμε την προσέγγισή μας στο κεφάλαιο 3.1.

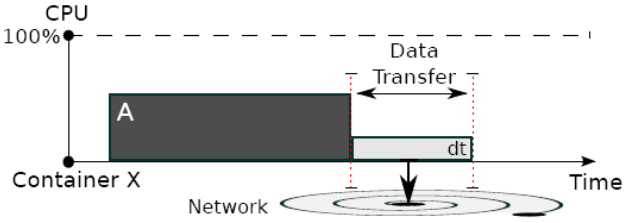
Οι περιορισμοί που επιβάλει το περιβάλλον εκτέλεσης οφείλεται στους περιορισμένους πόρους του συστήματος. Από αυτήν την άποψη, κατηγοριοποιούμε τους πόρους σε διαμοιραζόμενους (time-shared) και αδιαίρετους (space-shared)[11]. Οι διαμοιραζόμενοι πόροι μπορούν να χρησιμοποιηθούν από πολλούς τελεστές ταυτόχρονα χωρίς επιβάρυνση της λειτουργίας τους. Ωστόσο, η ταυτόχρονη χρήση των αδιαίρετων πόρων πέραν των ορίων του υποδοχέα συνεπάγεται υψηλό κόστος. Για παράδειγμα, ο δίσκος θα μπορούσε να χρησιμοποιηθεί ως επέκταση της μνήμης, αλλά αυτό συνεπάγεται ότι το διάβασμα και το γράψιμο σε αυτή την «μνήμη» θα είναι τουλάχιστον μια τάξη μεγέθους πιο αργή. Θεωρούμε την μνήμη ως τον μόνο αδιαίρετο πόρο, ενώ ο επεξεργαστής και το δίκτυο είναι διαμοιραζόμενοι. Οι περιορισμοί επιβάλλονται μόνο από τους αδιαίρετους πόρους. Δηλαδή, σε κάθε υποδοχέα, η μνήμη πρέπει να είναι επαρκής για την εκτέλεση των τελεστών που του έχουν ανατεθεί. Από την άλλη πλευρά, ο επεξεργαστής και το δίκτυο απαιτούν ιδιαίτερη μεταχείριση. Περιγράφουμε την προσέγγισή μας για την ταυτόχρονη χρήση αυτών των πόρων στην παράγραφο 3.2.

3.1 Κόστος Δικτύου

Δύο θέματα πρέπει να αντιμετωπιστούν όσον αφορά το κόστος του δικτύου. Το χειρισμό των τελεστών Α&Π και το χειρισμό των τελεστών ΣΛ. Η μεταφορά μέσω του δικτύου μοντελοποιείτε με ειδικούς τελεστές που μεταφέρουν τα δεδομένα (dt). Αυτοί οι τελεστές εισάγονται μεταξύ των τελεστών της ροής *ροή(παραγωγός, καταναλωτής, δεδομένα)*, αν ο παραγωγός και ο καταναλωτής έχουν ανατεθεί σε διαφορετικούς υποδοχείς. Πάντα εισάγονται δύο τελεστές dt, ένας μετά τον παραγωγό και ένας πριν τον καταναλωτή. Εάν ο τελεστής είναι Α&Π (Σχήμα 2), ο τελεστής dt δημιουργεί ένα νέο κόμβο στο γράφο ροής δεδομένων. Από την άλλη πλευρά, εάν ο τελεστής είναι ΣΛ (Εικόνα 3), η λειτουργία του τελεστή dt ενσωματώνετε στον ίδιο το τελεστή χωρίς να αλλάξει ο γράφος. Η εικόνα 3Α απεικονίζει την πραγματική εκτέλεση του τελεστή Α η οποία είναι αναμειγμένη με τον τελεστή dt και το σχήμα 3Β, απεικονίζει τη μοντελοποίηση του Α. Ο χρόνος εκτέλεσης αυξάνεται και η χρήση της CPU μειώνετε. Για απλότητα, υποθέτουμε ότι η αρχική καθυστέρηση είναι 0. Στο τελικό γράφημα, όλη η επικοινωνία μέσω του δικτύου γίνεται από τελεστές ΣΛ, ανεξάρτητα από τις ιδιότητες που είχαν οι τελεστές στον αρχικό γράφο.

3.1.1 Τελεστής Αποθήκευση και προώθηση

Έστω A ένας τελεστής A&Π που ανήκει στον γράφο G το οποίο ορίζετε ως *A(_, _, _, A&Π)* με *ανάθεση(A, X, _, _)*. Για κάθε τελεστή B με *ροή(A, B, $D_{A \to B}$)* και *ανάθεση(B, Y, _, _)*, αν x ≠ y, γίνετε εισαγωγή στο G ενός τελεστή μεταφοράς δεδομένων dt μεταξύ των A και B, ως εξής: αφαίρεση της *ροή(A , B, $D_{A \to B}$)*, εισαγωγή της *ροή(A, dt, $D_{A \to B}$)*, και *ροή(dt, B, $D_{A \to B}$)*, και *ανάθεση(dt, X, _, _)*.



Εικόνα 2: Τελεστής αποθήκευση και προώθηση.

Ο χρόνος εκτέλεσης του dt ορίζεται ως εξής:

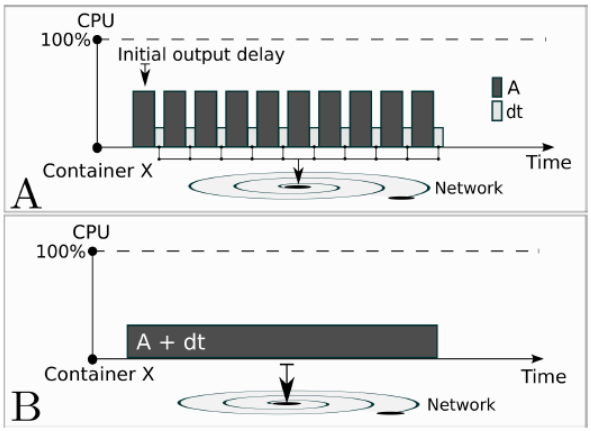
$$dt.time = \frac{D_{A \to B}}{\min(X.network, Y.network)} \qquad (1)$$

$$dt(time, DT_{CPU}, DT_{MEM}, PL)$$

όπου DT_{CPU} και DT_{MEM} να σταθερές τιμές για την χρήση του επεξεργαστή και τις απαιτήσεις σε μνήμη των τελεστών που μεταφέρουν δεδομένα. Η ίδια μεθοδολογία ακολουθείται για όλους τους τελεστές C με *ροή(C, A, $D_{C \to A}$)*, των οποίων οι έξοδοι καταναλώνονται από τον A.

3.1.2 Τελεστής Σωλήνωση

Έστω A και B δύο συνδεδεμένοι τελεστές που ανήκουν στον γράφο ροής δεδομένων G με *ροή(A, B, DA → B)*. Έστω ότι οι αναθέσεις των A και B είναι ανάθεση(A, X, _, _) και ανάθεση(B, Y, _, _) αντίστοιχα, με X ≠ Y.



Εικόνα 3: Τελεστής σωλήνωσης

Υποθέτουμε ότι οι χρόνοι εκτέλεσης των δύο τελεστών έχουν πλήρη επικάλυψη. Χρησιμοποιώντας την εξίσωση 1, ο χρόνος εκτέλεσης είναι:

$$T = \max(A.time + dt.time, B.time + dt.time)$$

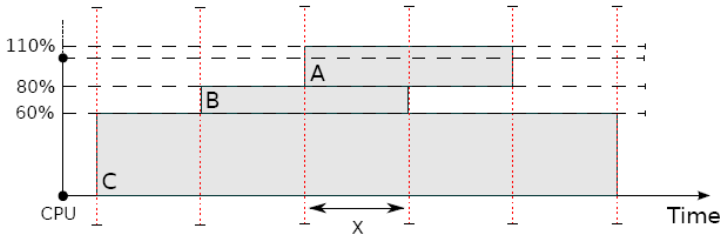
Οι νέες ιδιότητες του τελεστή A είναι:

$$A(T, \frac{A.time * A.cpu + dt.time * DT_{CPU}}{T}, -, -)$$

Η μνήμη και η συμπεριφορά του δεν αλλάζουν. Οι νέες ιδιότητες του B υπολογίζονται με παρόμοιο τρόπο. Η ίδια τεχνική εφαρμόζεται σε όλους τους συνδεδεμένους τελεστές.

3.2 Επικάλυψη Τελεστών

Δύο ή περισσότεροι τελεστές οι οποίοι ανατίθενται στον ίδιο υποδοχέα μοιράζονται τους ίδιους πόρους. Το πρόβλημα εμφανίζεται όταν οι τελεστές που είναι ενεργοί απαιτούν πάνω από το 100% χρήσης του επεξεργαστή. Για παράδειγμα, θεωρήστε τους τελεστές A, B, και C της εικόνας 4. Με βάση την παραδοχή της ομοιόμορφης κατανάλωσης πόρων, οι περιοχές ανάμεσα στα κατώφλια έχουν ομοιόμορφη χρήση του επεξεργαστή. Το πρόβλημα εμφανίζεται στην περιοχή X, στην οποία το σύνολο των αναγκών του επεξεργαστή είναι πάνω από 100% του διαθέσιμου. Αυτό το πρόβλημα αντιμετωπίζεται επεκτείνοντας το μήκους της περιοχής X και, ως εκ τούτου, τον συνολικό χρόνο εκτέλεσης όλων των τελεστών, μειώνοντας τη χρήση του επεξεργαστή που χρησιμοποιούν στην περιοχή X.



Εικόνα 4: Επικάλυψη τελεστών

Πιο συγκεκριμένα, έστω G ένα διάγραμμα ροής δεδομένων και C ένας υποδοχέας. Έστω OP = {opj} το σύνολο των τελεστών που έχουν ανατεθεί στο C και R = {ri} το σύνολο των χρονικών περιόδων που σχηματίζονται από την αρχή ή το τέλος των τελεστών. Έστω time(ri) και cpu(ri) είναι η διάρκεια και η χρήση του επεξεργαστή στην περιοχή ri. Με βάση τα παραπάνω, ισχύει το ακόλουθο:

$$cpu(r_i) = \sum_{j=1}^{|OP|} (op_j).cpu * \delta(op_j, r_i, C)$$

όπου,

$$\delta(op_j, r_i, C) = \begin{cases} 1, & \text{if } op_j \text{ active in } r_i \text{ in } C \\ 0, & \text{otherwise} \end{cases}$$

Η διάρκεια της ri υπολογίζεται ως εξής:

$$time(r_i) = \begin{cases} time(t_i), & \text{if } cpu(t_i) \leq 1 \\ time(t_i) * cpu(t_i), & \text{otherwise} \end{cases}$$

η επέκταση της περιοχής ri επηρεάζει τη διάρκεια όλων των τελεστών που είναι ενεργοί στην ri. Η ίδια τεχνική που χρησιμοποιείται για την ταυτόχρονη χρήση του επεξεργαστή εφαρμόζεται και για την ταυτόχρονη χρήση του δικτύου.

4. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΧΡΗΜΑΤΟΣ

Οι πάροχοι υπηρεσιών νέφους κβαντίζουν τον χρόνο που μισθώνουν τους υπολογιστικούς πόρους. Το κβάντο αυτό είναι συνήθως μία ώρα αλλά αρχίζουν να εμφανίζονται και άλλες εναλλακτικές μορφές [2]. Το ελάχιστο χρηματικό κόστος $m(SG)_{min}$ που απαιτείτε την εκτέλεση ενός γράφου ροής δεδομένων G στο νέφος, δεδομένου ενός χρονοπρογράμματος SG και της τιμολόγησης (το μέγεθος του κβάντου Q_t και το κόστος της μίσθωσης Q_m ενός υποδοχέα για Q_t διάρκεια) είναι δύσκολο να υπολογιστεί. Σε αυτό την εργασία κάνουμε έναν υπολογισμό κατά προσέγγιση. Σε κάθε υποδοχέα, μοιράζουμε τον χρόνο σε παράθυρα μήκους Q_t ξεκινώντας από τον πρώτο τελεστή του χρονοπρογράμματος. Το κόστος είναι τότε το άθροισμα των παραθύρων που έχουν τουλάχιστον έναν τελεστή, πολλαπλασιαζόμενο με το Q_m.

$$m(S_G) = Q_m * (\sum_{i=1}^{|C|} \sum_{j=1}^{|W|} \epsilon(c_i, w_j))$$

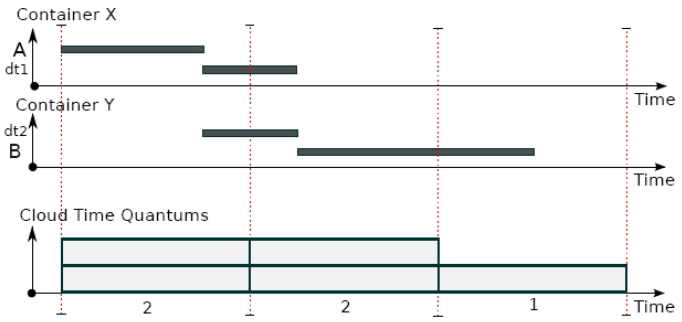
όπου C = {c_i} είναι το σύνολο των υποδοχέων, W = {W_j} είναι το σύνολο των χρονικών παράθυρων, και

$$\epsilon(c_i, w_j) = \begin{cases} 1, & \text{if at least one operator is active in } w_j \text{ in } c_i \\ 0, & \text{otherwise} \end{cases}$$

Ο κατακερματισμός f(SG) του χρονοπρογράμματος SG είναι ο χρόνος κατά τον οποίο οι πόροι που δεσμεύτηκαν δεν χρησιμοποιούνται αλλά χρεώνονται.

$$f(S_G) = Q_t * \sum_{i=1}^{|C|} \sum_{j=1}^{|W|} (\epsilon(c_i, w_j) - \tau(c_i, w_j))$$

όπου τ(c, w) είναι το ποσοστό του χρόνου που οι τελεστές είναι ενεργοί στον υποδοχέα c_i στο χρονικό παράθυρο W_j. Είναι εύκολο να αποδειχθεί ότι $\lim_{Qt \to 0} f(SG) = 0$ για κάθε γράφο ροής δεδομένων. Αυτό σημαίνει ότι αν οι πόροι χρεώνονται για ακριβώς όσο χρόνο χρησιμοποιούνται, δεν υπάρχει κατακερματισμός.

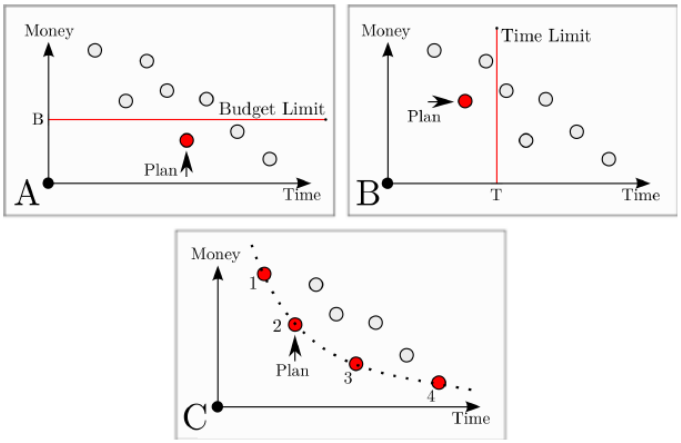


Εικόνα 5: Το χρηματικό κόστος ενός χρονοπρογράμματος

Στο Σχήμα. 5 δείχνουμε ένα πλάνο εκτέλεσης με τέσσερις τελεστές. Το κόστος είναι 5*Q_m. Υπάρχουν τρία κβάντα που δεν χρησιμοποιούνται πλήρως. Ο κατακερματισμός του είναι 5 * Q_t - (1 + 1 4 + 1 4 + 1 + 1 2) * Q_t = 2 * Q_t.

5. ΠΡΟΒΛΗΜΑΤΑ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Όπως αναφέρθηκε νωρίτερα, ο χώρος των λύσεων είναι δύο διαστάσεων. Οι λύσεις που ανήκουν στην κορυφογραμμή (καμπύλη Pareto) [24] αποτελούν συμβιβασμούς μεταξύ χρόνου και χρήματος. Ορίζουμε δύο είδη προβλημάτων βελτιστοποίησης: κάτω από περιορισμούς και κορυφογραμμής. Όλα τα προβλήματα παρουσιάζονται στην εικόνα 6. Τα προβλήματα περιορισμών είναι (Π1) βρες το γρηγορότερο πλάνο δεδομένου ενός οικονομικού προϋπολογισμού (εικόνα 6Α) και (Π2) βρες το φτηνότερο πλάνο δεδομένης μίας χρονικής προθεσμίας (εικόνα 6Β). Τα προβλήματα αυτά είναι συμμετρικά και είναι μονοδιάστατα προβλήματα βελτιστοποίησης, μόνο κάτω από περιορισμούς την άλλη διάσταση. Στο πρόβλημα κορυφογραμμής, δεν υπάρχουν περιορισμοί εκ των προτέρων. Οι διάφορες λύσεις που βρίσκονται στην κορυφογραμμή του χώρου χρόνος/χρήμα προτείνονται από το σύστημα και αυτό με τον καλύτερο συμβιβασμό μεταξύ χρόνου και χρήματος επιλέγεται μετά από τη βελτιστοποίηση. Στην εικόνα 6C, ο βελτιστοποίησης θα επιστρέψει όλα τις Pareto βέλτιστες λύσεις, και ο ενδιαφερόμενος χρήστης θα επιλέξει αυτό που εκείνος θεωρεί βέλτιστο.



Εικόνα 6: Τα τρία προβλήματα βελτιστοποίησης

Το σχήμα της κορυφογραμμής μπορεί να συλληφθεί από μια μετρική που ονομάζεται ελαστικότητα και ορίζεται ως εξής:

$$E = \frac{(T_{max} - T_{min})/T_{max}}{(M_{max} - M_{min})/M_{max}}$$

όπου T_{min} και T_{max} είναι η ελάχιστη και μέγιστη τιμή στην διάσταση του χρόνου των χρονοπρογραμμάτων που βρίσκονται στην κορυφογραμμή. Ομοίως, τα M_{min} και M_{max} είναι το ελάχιστο και το μέγιστο στην διάσταση του χρήματος. Αυτή η μετρική εκφράζει την επιτάχυνση του χρόνου ολοκλήρωσης αν περισσότερα χρήματα είναι διαθέσιμα [31]. Για ροές δεδομένων με υψηλή ελαστικότητα, ένα μικρό χρηματικό ποσό συνεισφέρει πολύ στην ελαχιστοποίηση του χρόνου εκτέλεσης (δηλαδή, αξίζει να πληρώσει κανείς για να επιταχύνει τον χρόνο εκτέλεσης). Από την άλλη πλευρά, δίνοντας περισσότερα χρήματα, δεν έχει σημαντικές επιπτώσεις στο χρόνο ολοκλήρωσης για ροές δεδομένων με χαμηλή ελαστικότητα. Για αυτές τις ροές, αξίζει τον κόπο, για παράδειγμα, να πληρώσει κάποιος 50% λιγότερο για να χάσει μόνο το 10% από τον χρόνο ολοκλήρωσης!

6. ΑΛΓΟΡΙΘΜΟΙ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Ανεξάρτητα από το συγκεκριμένο είδος προβλήματος βελτιστοποίησης, ουσιαστικά οι ίδιο αλγόριθμοι μπορούν να χρησιμοποιηθούν για να γίνει η αναζήτηση στο χώρο των εναλλακτικών λύσεων και να βρεθεί η βέλτιστη ή οι βέλτιστες. Σε αυτή την εργασία, προτείνουμε μια οικογένεια από αλγόριθμους που ακολουθούν μια προσέγγιση εμφωλευμένων βρόχων. Ο εξωτερικός βρόχος καλεί τον εσωτερικό βελτιστοποιητή αρκετές φορές, με διαφορετικό άνω όριο στον αριθμό των παράλληλων υποδοχέων που μπορούν να χρησιμοποιηθούν στο χρονοπρόγραμμα. Ο αλγόριθμος αυτός σταματάει όταν το κριτήριο βελτιστοποίησης δεν βελτιωθεί πολύ μετά από ένα συγκεκριμένο αριθμό διαδοχικών επαναλήψεων.

Ο βαθμός παραλληλισμού P_G της ροής δεδομένων G είναι ένα σημαντικός προγνωστικός παράγοντας της απόδοσης. Ουσιαστικά είναι ο μέγιστος αριθμός των τελεστών που μπορούν να είναι ενεργοί ταυτόχρονα όταν ο αριθμός των υποδοχέων είναι απεριόριστος. Υποθέστε ότι ο αριθμός των υποδοχέων και η ταχύτητα του δικτύου είναι άπειρη. Ικανοποιώντας μόνο τους περιορισμούς που συνεπάγονται από τον γράφο ροής δεδομένων, μια προσέγγιση του PG είναι ο μέγιστος αριθμός των τελεστών που έχουν χρονική επικάλυψη στην εκτέλεση τους.

Ο εσωτερικός βρόχος, βελτιστοποιεί ως προς τον χρόνο ολοκλήρωσης ή το χρηματικό κόστος με περιορισμό στην άλλη διάσταση. Χρησιμοποιούμε μια σουίτα από άπληστους και πιθανοτικούς αλγόριθμους για το εσωτερικό βρόχο. Οι αλγόριθμοι αυτοί παρουσιάζονται στα επόμενα κεφάλαια.

6.1 Αλγόριθμος Εξωτερικού Βρόχου

Έχουμε υλοποιήσει έναν γενικό βελτιστοποιητή εμφωλευμένων βρόγχων (Αλγόριθμος 1) το οποίο τα λύνει προβλήματα περιορισμών ή κορυφογραμμής, ανάλογα με τις τιμές των παραμέτρων του. Οι παράμετροι του αλγόριθμου 1 είναι: 1) ο γράφος ροής δεδομένων G όπως ορίζεται στο κεφάλαιο 2. 2) CONST είναι μια ρουτίνα που επιστρέφει ‘αληθές’ αν ένα χρονοπρόγραμμα ικανοποιεί τους περιορισμούς. Αν η CONST επιστρέφει πάντα ‘αληθές’ (χρόνος < ∞ & χρήμα < ∞) είναι αρκετό για να εκφραστεί η έλλειψη περιορισμών (χρησιμοποιείται στο πρόβλημα κορυφογραμμής). 3) η FILTER είναι μια ρουτίνα που εφαρμόζεται σε ένα σύνολο χρονοπρογραμμάτων και επιστρέφει ένα περιορισμένο σύνολο, αφαιρώντας τα χρονοδιαγράμματα που κυριαρχούνται από άλλα, ανάλογα με το χρόνο, το χρήμα, ή και τα δύο (κορυφογραμμή). 4) η LIMIT είναι μια γεννήτρια ορίων υποδοχέων. Στην απλούστερη μορφή της, γενάει όρια μέχρι το συνολικό αριθμό των τελεστών της ροής δεδομένων. Ωστόσο, στις περισσότερες περιπτώσεις, αυτό είναι ένα πολύ χαλαρό άνω όριο. 5) η STOP είναι μια ρουτίνα που καθορίζει το αν πρέπει ή όχι να σταματήσει η εξερεύνηση βασιζομένη σε κάποια από τα πολλά πιθανά κριτήρια. Ο απλούστερος τρόπος είναι να καλεστεί το χαμηλότερο επίπεδο διαδοχικά για όλα τα όρια που γεννιούνται από την ρουτίνα LIMIT. Μια άλλη δυνατότητα είναι να τερματιστεί η εξερεύνηση όταν η διαφορά μεταξύ των τιμών της παραμέτρου OPT για ένα συγκεκριμένο αριθμό διαδοχικών χρονοπρογραμμάτων να είναι κάτω από ένα όριο. Τέλος, το OPT είναι ένας μονοδιάστατος βελτιστοποιητής που προσπαθεί να αναθέσει τους τελεστές σε υποδοχείς, είτε ελαχιστοποιώντας το χρόνο ή το χρήμα. Παρακάτω περιγράφουμε τις παραμέτρους του αλγορίθμου 1 όπως αρχικοποιούνται για καθένα από τα τρία προβλήματα βελτιστοποίησης.

Input:
G: The dataflow graph
CONST: Solution constraints
FILTER: Solution space filter
LIMIT: Container limit sequence generator
STOP: Stopping condition
OPT: Lower level optimizer
Output:
$space$: The space of solutions

1: $space \leftarrow \oslash$
2: **while** LIMIT.$hasNext()$ and STOP.$continue()$ **do**
3: $limit \leftarrow$ LIMIT.$getNext()$
4: $next \leftarrow$ OPT$(G, limit,$ CONST$)$
5: $space \leftarrow$ FILTER$(space \cup \{next\})$
6: STOP.$addFeedback(next)$
7: LIMIT.$addFeedback(next)$
8: **end while**
9: **return** $space$

Αλγόριθμος 1: Γενικός Αλγόριθμος Εμφωλευμένων Βρόγχων

Πρόβλημα Π1: Η CONST ελέγχει κατά πόσον ένα χρονοπρόγραμμα καλύπτεται από το χρηματικό προϋπολογισμό. Η FILTER επιστρέφει μόνο το γρηγορότερο χρονοπρόγραμμα. Η LIMIT παράγει το πολύ 20 όρια υποδοχέων χωρίζοντας σε ίσα τμήματα το διάστημα [1, PG]. Η STOP ελέγχει αν τα τελευταία πέντε χρονοπρογράμματα διαφέρουν σημαντικά ως προς το χρόνο ολοκλήρωσης τους. Χρησιμοποιώντας γραμμική παλινδρόμηση (linear regression), υπολογίζουμε την γραμμή στο χώρο χρόνος/επανάληψη χρησιμοποιώντας τις πέντε τελευταίες επαναλήψεις. Εάν η κλίση της γραμμής είναι λιγότερο από 0,1, η εξερεύνηση σταματάει. Η OPT ελαχιστοποιεί το χρόνο ολοκλήρωσης με τον χρηματικό προϋπολογισμού ως περιορισμό.

Πρόβλημα Π2: Η CONST ελέγχει κατά πόσον ένα πρόγραμμα βρίσκεται εντός του χρονικού ορίου. Η FILTER επιστρέψει μόνο το φθηνότερο χρονοπρόγραμμα. Η LIMIT είναι το ίδιο με το πρόβλημα Π1. Η STOP υπολογίζεται με τον ίδιο τρόπο όπως και στην Π1, αλλά η γραμμή υπολογίζεται στο χώρο χρήματα/επανάληψης. Η OPT ελαχιστοποιεί το χρηματικό κόστος έχοντας το χρονικό όριο ως περιορισμό.

Πρόβλημα Κορυφογραμμής: Η CONST δέχεται οποιοδήποτε χρονοδιάγραμμα. Η FILTER επιστρέφει την κορυφογραμμή των λύσεων. Η LIMIT είναι το ίδιο με το πρόβλημα Π1 και η κατάσταση διακοπής είναι πάντα ψευδής. Η OPT μπορεί να είναι οποιοδήποτε αλγόριθμος που περιγράφετε στις επόμενες επιμέρους ενότητες. Το μέγεθος της κορυφογραμμής είναι το πολύ ο αριθμός των ορίων που παράγεται από την LIMIT. Στις παρακάτω ενότητες παρουσιάζουμε τους διάφορους αλγόριθμοι που έχουμε διερευνήσει.

6.2 Άπληστοι Αλγόριθμοι Χρονοπρογραμματισμού

Έχουμε υλοποιήσει διάφορους άπληστους αλγόριθμους χρονοπρογραμματισμού [17] χρησιμοποιώντας διάφορα ευριστικά. Ο αλγόριθμος 2, παρουσιάζει τον γενικό άπληστο αλγόριθμο. Μόνο δύο ρουτίνες πρέπει να οριστούν. Η NEXT επιστρέφει τον επόμενο τελεστή που θα ανατεθεί επιλέγοντας από ένα σύνολο τελεστών που είναι έτοιμοι γι 'αυτό και η ASSIGN επιστρέφει τον υποδοχέα στον οποίο θα ανατεθεί ο επόμενος τελεστής. Ένας τελεστής είναι υποψήφιος για ανάθεση από τη στιγμή που δεν έχει εξαρτήσεις από άλλους τελεστές. Αρχικά, οι τελεστές που δεν έχουν εισόδους, δεν

έχουν εξαρτήσεις. Οι τελεστές Α&Π είναι υποψήφιοι όταν όλες οι είσοδοι τους είναι διαθέσιμές. Οι τελεστές ΣΛ είναι υποψήφιοι όταν όλες οι είσοδοι προέρχονται είτε από τελεστές ΣΛ ή από τελεστές Α&Π που έχουν τερματίσει.

Input:
G: The dataflow graph
C: The maximum number of parallel containers to use
CONST: Solution constraints
NEXT: Next operator to assign
ASSIGN: Container the next operator is assigned to
Output:
S_G: The schedule of G with at most C containers

1: $S_G.assigns \leftarrow \oslash$
2: $ready \leftarrow$ {operators in G that has no dependencies}
3: **while** $ready \neq \oslash$ **do**
4: $n \leftarrow$ NEXT$(ready)$
5: $candidates \leftarrow$ {containers that assignment of n satisfy CONST}
6: **if** $candidates = \oslash$ **then**
7: **return** $ERROR$
8: **end if**
9: $c \leftarrow$ ASSIGN$(n, candidates)$
10: $ready \leftarrow ready - \{n\}$
11: $ready \leftarrow ready +$ {operators in G that constraints no longer exist}
12: $S_G.assigns \leftarrow S_G.assigns + \{assign(n, c, -, -)\}$
13: **end while**
14: **return** S_G

Αλγόριθμος 2: Γενικός Άπληστος Αλγόριθμος

Ορίζουμε τέσσερις άπληστους αλγόριθμους. Ο G-BRT εξισορροπεί την χρήση των υποδοχέων, ο G-MNT ελαχιστοποιεί την μεταφορά δεδομένων μέσω του δικτύου, ο G-MPT ελαχιστοποιεί το χρόνο ολοκλήρωσης, και ο G-MPM ελαχιστοποιεί το χρηματικό κόστος. Πιο συγκεκριμένα, σε κάθε βήμα, ο G-BRT αναθέτει τον τελεστή με την μέγιστη διάρκεια στον υποδοχέα που ελαχιστοποιεί την τυπική απόκλιση της χρήσης των υποδοχέων. Το τελευταίο είναι το άθροισμα του χρόνου εκτέλεσης των τελεστών που έχουν ανατεθεί στον υποδοχέα. Ο G-MNT αναθέτει τον τελεστή με το μέγιστο μέγεθος εξόδου στον υποδοχέα που ελαχιστοποιεί τα δεδομένα που μεταφέρονται μέσω του δικτύου. Ο G-MPT αναθέτει τον τελεστή με το μεγαλύτερο χρόνο εκτέλεσης στον υποδοχέα που ελαχιστοποιεί το χρόνο ολοκλήρωσης. Τέλος, ο G-MPM αναθέτει τον τελεστή με το μέγιστο μέγεθος εξόδου στον υποδοχέα που ελαχιστοποιεί το χρηματικό κόστος του χρονοπρογράμματος.

6.3 Αλγόριθμοι Τοπικής Αναζήτησης

Η μέθοδος τοπικής αναζήτησης που χρησιμοποιούμε είναι η προσομοιωμένη ανόπτηση [14]. Υλοποιήσαμε ένας γενικό αλγόριθμο προσομοιωμένης ανόπτησης που απαιτείται τον ορισμό μόνο τριών ρουτινών. Η INIT καθορίζει το αρχικό χρονοπρόγραμμα από το οποίο αρχίζει την αναζήτηση, η COST επιστρέφει την τιμή της παραμέτρου βελτιστοποίηση για ένα συγκεκριμένο χρονοπρόγραμμα, και η NEIGHBOR επιστρέφει έναν γείτονα ενός συγκεκριμένου χρονοδιαγράμματος. Δεν δεχόμαστε γείτονες που δεν πληρούν τους περιορισμούς.

Ορίζουμε τους εξής αλγορίθμους: ο SA-MPT ξεκινά με μια τυχαία ανάθεση και τη ρουτίνα του COST επιστρέφει το χρόνο ολοκλήρωσης ενός συγκεκριμένου χρονοδιαγράμματος, SA-MPT2 αρχίζει με την έξοδο του G-MPT ως αρχική κατάσταση

και έχει την ίδια ρουτίνα COST με τον SA-MPT. Ο SA-MPM ξεκινά με μια τυχαία ανάθεση και η COST επιστρέφει το χρηματικό κόστος ενός συγκεκριμένου χρονοδιαγράμματος, και τέλος, ο SA-MPM2 αρχίζει με την έξοδο του G-MPM ως αρχική κατάσταση και έχει την ίδια ρουτίνα COST με τον SA-MPM. Όλοι οι αλγόριθμοι παράγουν ένα τυχαίο γείτονα αναθέτοντας έναν τελεστή επιλεγμένο τυχαία σε κάποιον άλλον υποδοχέα, επίσης επιλεγμένο τυχαία. Πιο εξελιγμένες μέθοδοι επιλογής γείτονα αφήνονται για μελλοντικές εργασίες.

6.4 Εκτίμηση Χρόνου Εκτέλεσης

Ο αλγόριθμος 3 είναι στην καρδιά όλων των αλγορίθμων που περιγράψαμε. Δεδομένου ενός χρονοδιαγράμματος, αυτός ο αλγόριθμος υπολογίζει πότε και για πόσο χρόνο κάθε τελεστής θα είναι ενεργός. Έτσι, ο χρόνος ολοκλήρωσης και το χρηματικό κόστους υπολογίζεται για ολόκληρο το χρονοπρόγραμμα.

Input:
G: The dataflow graph
S_G: A (partial) $schedule(assigns)$
Output:
S_G: The schedule with calculated $start$ and end for all ops in S_G

1: $ready \leftarrow \oslash$
2: $time \leftarrow 0$
3: $G \leftarrow G$ + {data transfer ops before & after S&F operators if needed}
4: $queued \leftarrow$ {ops in G that have no dependencies}
5: **for** all operators A in $queued$ that satisfy memory constraints **do**
6: $ready \leftarrow ready + \{assign(A, -, time, -)\}$
7: **end for**
8: **while** $ready \neq \oslash$ **do**
9: $next\_event \leftarrow$ `find_next_event`$(ready)$
10: $terminated \leftarrow$ `forward_in_time`$(next\_event, ready)$
11: $time \leftarrow next\_event$
12: **for** all operators A in $terminated$ **do**
13: $S_G \leftarrow S_G + \{assign(A, -, -, time)\}$
14: **end for**
15: $ready \leftarrow ready - terminated$
16: $queued \leftarrow queued$ + {ops in G that constraints no longer exist}
17: **for** all ops A in $queued$ that satisfy memory constraints **do**
18: $ready \leftarrow ready + \{assign(A, -, time, -)\}$
19: **end for**
20: **end while**
21: **return** S_G

Αλγόριθμος 3: Εκτίμηση Χρόνου Εκτέλεσης

Η ρουτίνα find_next_event επιστρέφει τον χρόνο κατά τον οποίο συμβαίνει ο πρώτος τερματισμός τελεστή υποθέτοντας ομοιόμορφη συμπεριφορά. Η ρουτίνα forward_in_time προσομοιώνει την εκτέλεση όλων των τελεστών μέχρι το χρονικό σημείο που υπολογίζονται από την find_next_event, και επιστρέφει τους τελεστές που έχουν τερματίσει.

6.5 Πολυπλοκότητα

Μελετάμε τα χειρότερα σενάρια για τον αλγόριθμο Εκτίμησης Χρόνου Εκτέλεσης, τον Γενικό Άπληστο Αλγόριθμο, και τον Γενικό Αλγόριθμο Προσομοιωμένης Ανόπτησης.

Θεωρούμε ως δεδομένο έναν γράφο ροής δεδομένων G με N τελεστές και L ροές και C τον ανώτατο αριθμό υποδοχέων.

6.5.1 Αλγόριθμος Εκτίμησης Χρόνου Εκτέλεσης

Ο μέγιστος αριθμός των τελεστών μετά από την προσθήκη τελεστών μεταφοράς είναι $N_o = N + 2 \bullet L$ και ο μέγιστος αριθμός των ροών είναι $N_L = 3 \bullet L$. Το χειρότερο σενάριο είναι όλοι οι τελεστές να είναι έτοιμοι να εκτελεστούν στην αρχή και μόνο ένας τελεστής να τερματίζει κάθε φορά. Έτσι,

$$C_{pde} = L + N_O + 2 \cdot \sum_{i=1}^{N_O} i = L + N_O + 2 \cdot \frac{N_O \cdot (N_O + 1)}{2}$$

Άρα,

$$C_{pde} = O(N_O^2) = O((N + 2 \cdot L)^2)$$

6.5.2 Γενικός Άπληστος Αλγόριθμο

Το χειρότερο σενάριο είναι όλοι οι τελεστές να είναι έτοιμοι στην αρχή και σε κάθε βήμα, ο επόμενος τελεστής να ανατίθεται σε διαφορετικό υποδοχέα. Υποθέτουμε ότι η πολυπλοκότητα της NEXT και ASSIGN είναι γραμμική ως προς το μέγεθος της εισόδου τους. Ως εκ τούτου, η συνολική πολυπλοκότητα που προστίθεται από τις κλήσεις του NEXT είναι $\sum_{i=1}^{N} i$. Η πολυπλοκότητα της ASSIGN είναι $\sum_{i=1}^{C} i + (N - C)$. Τέλος, η πολυπλοκότητα από τις κλήσεις του SDE είναι: $C_{pde} \cdot (\sum_{i=1}^{C} i + (N - C) \cdot C)$. Υποθέτοντας ότι N >> C, η κλήσεις του SDE είναι πιο χρονοβόρος από την ASSIGN, και ως εκ τούτου

$$C_{greedy} = O(N \cdot (N + 2 \cdot L)^2)$$

6.5.3 Γενικός Αλγόριθμος Προσομοιωμένης Ανόπτησης

Ο αλγόριθμος προσομοιωμένης ανόπτηση κάνει το πολύ k βήματα για να τερματίσει και σε κάθε βήμα, γίνετε μια κλήση της του αλγορίθμου εκτίμησης χρόνου. Ως εκ τούτου, η πολυπλοκότητα είναι:

$$C_{annealing} = C_{init} + k \cdot C_{pde}$$

όπου C_{init} είναι η πολυπλοκότητα της ρουτίνας INIT.

7. ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ

Σε αυτό το κεφάλαιο, περιγράφουμε τη προσπάθεια πειραματισμού μας και τα αποτελέσματα που έχουμε εξάγει από αυτό. Παρά το γεγονός ότι έχουμε πειραματιστεί και με τα τρία προβλήματα βελτιστοποίησης, παρουσιάζουμε μόνο τα αποτελέσματα της βελτιστοποίησης κορυφογραμμής, δηλαδή, χωρίς περιορισμούς του χρήματος ή του χρόνου εκτέλεσης, καθώς έχει τη μεγαλύτερη πρόκληση. Παρομοίως, για όλες τις παραμέτρους που επηρεάζουν τα πειράματα μας, συζητούνται μόνο τα πιο ενδιαφέροντα αποτελέσματα ή χαρακτηριστικά.

7.1 Παράμετροι Πειραμάτων

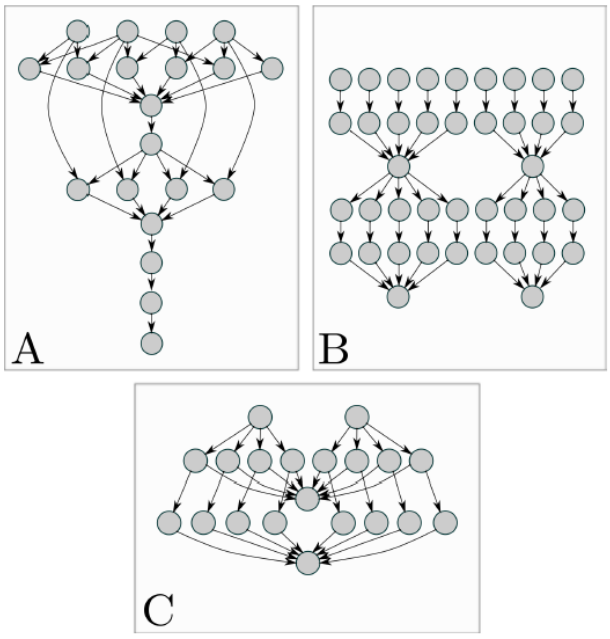
Τα πειράματα που πραγματοποιήσαμε χαρακτηρίζονται από τέσσερα στοιχεία τα οποία περιγράφονται στα παρακάτω κεφάλαια.

7.1.1 Περιβάλλον Εκτέλεσης

Στα πειράματά μας, υποθέτουμε ότι όλοι οι υποδοχείς είναι παρόμοιοι, δηλαδή, διαθέτουν τους ίδιους πόρους (επεξεργαστή, μνήμη, και ταχύτητα δικτύου). Πιο συγκεκριμένα, υποθέτουμε ότι οι υποδοχείς έχουν έναν επεξεργαστή και συνολική μνήμη και δίκτυο ίσο με 1,0.

7.1.2 Δομή Γράφου Ροής Δεδομένων

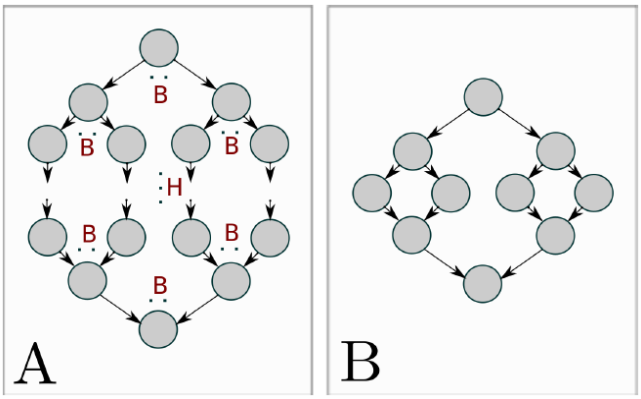
Έχουμε εξετάσει τέσσερις οικογένειες γράφων ροών δεδομένων: Montage [16] (Εικόνα 7Α), Ligo [8] (Εικόνα 7Β), Cybershake [7] (Εικόνα 7Γ), και Lattice (Εικόνα 8α).



Εικόνα 7: Οι γράφοι Montage(A), Ligo(B), και Cybershake(C).

Οι τρεις πρώτοι είναι γράφοι ροών δεδομένων που χρησιμοποιούνται σε πραγματικές εφαρμογές. Το Montage χρησιμοποιείται από τη NASA για τη δημιουργία του ουράνιου

θόλου. Το Ligo χρησιμοποιείται από το Laser Interferometer Gravitational-wave Observatory για να αναλύσει γαλαξιακά δυαδικά συστήματα. Τέλος, το Cybershake χρησιμοποιείται από το Southern California Earthquake Center για να χαρακτηρίσει σεισμούς.



Εικόνα 8: Ο γράφος Lattice (A) και ένα παράδειγμα Lattice 5-2 (B).

Αν παρατηρήσουμε τη συμπεριφορά του φόρτου εργασίας στην διάρκεια του χρόνου, μπορούμε να εντοπίσουμε φάσεις υψηλού και χαμηλού παραλληλισμού. Ο παραλληλισμός μειώνεται όταν οι τελεστές συλλέγουν ή κατανέμουν δεδομένα (σημεία συμφόρησης). Το Montage έχει μια φάση με υψηλό παραλληλισμό, ενώ έχει αρκετούς τελεστές που δημιουργούν συμφόρηση. Το Ligo έχει ομοιόμορφη κατανομή παραλληλισμού και μέτριο αριθμό τελεστών συμφόρησης. Το Cybershake έχει δυο φάσεις υψηλού παραλληλισμού και πολύ λίγους τελεστές που δημιουργούν συμφόρηση και μεταφέρει μεγάλους όγκους δεδομένων. Το Lattice, εκ κατασκευής, μας παρέχει την επιλογή του βαθμού της παραλληλίας και της συμφόρησης. Αυτό το είδος είναι μια οικογένεια από συνθετικές ροές δεδομένων που έχουμε σχεδιάσει γενικεύοντας την τυπική MapReduce (όπως αυτή του Σχήματος 1). Αυτές οι ροές δεδομένων έχουν ένα ορισμένο ύψος (H) και παράγοντα διακλάδωσης (B) και συμβολίζεται με H-B Lattice. Για παράδειγμα, το σχήμα. 8B δείχνει το 5-2 Lattice.

Έχουμε πειραματιστεί με διάφορα μεγέθη γραφών ροής δεδομένων, από 5 έως 500 τελεστές. Εδώ παρουσιάζουμε τα αποτελέσματα του μεγαλύτερου γράφου (500 τελεστές), επειδή έχουν τη μεγαλύτερη πρόκληση. Η φύση των αποτελεσμάτων για τους μικρότερους γράφους είναι παρόμοια.

Πίνακας 1: Οι ροές δεδομένων Lattice.

H-B	500-1	11-3	9-4	7-7	5-21	3-498
Μέγεθος	500	485	426	457	485	500

Το Montage, Ligo, και Cybershake έχουν παραχθεί από τη γεννήτρια Pegasus [4]. Οι ροές δεδομένων Lattice έχουν παραχθεί από τις παραμέτρους HB που παρουσιάζονται στον Πίνακα 1.

7.1.3 Τύποι Τελεστών

Στα πειράματά μας, εκφράζουμε τις ιδιότητες των τελεστών ως ποσοστά των αντίστοιχων παραμέτρων των πόρων των υποδοχέων. Για παράδειγμα, ένας τελεστής που έχει ανάγκες μνήμης 0,4 χρησιμοποιεί το 40% της μνήμης του υποδοχέα. Επιπλέον, οι χρόνοι εκτέλεσης δίνεται ως ποσοστά του χρόνου του κβάντου του

νέφους, όπως επίσης και το μέγεθος των δεδομένων (είσοδοι / έξοδοι των τελεστών), λαμβάνοντας υπόψη την ταχύτητα του δικτύου. Για παράδειγμα, ο χρόνος εκτέλεσης 0,5 υποδεικνύει ότι ένας τελεστής απαιτεί μισό κβάντο χρόνου για να ολοκληρωθεί η εκτέλεσή του (δηλαδή, 30 λεπτά αν χρησιμοποιήσουμε την πολιτική της Amazon). Ομοίως, μια έξοδος τελεστή μεγέθους 0,2 απαιτεί το ένα πέμπτο του κβάντου για να μεταφερθεί μέσω του δικτύου (αν χρειαστεί να μεταφερθεί). Έτσι, το μέγεθος των δεδομένων εξόδου είναι σε αντίστροφη σχέση με την ταχύτητα του δικτύου. Το χρηματικό κόστος μετριέται όπως περιγράφεται στο κεφάλαιο 4.

Έχουμε χρησιμοποιήσει συνθετικό φόρτο εργασίας με βάση τα Montage, Ligo, και Cybershake, όπως ορίζονται στο [4]. Δεδομένου ότι η πρόθεσή μας ήταν να μελετήσουμε ροές με μεγάλο όγκο δεδομένων, κλιμακώνουμε τον χρόνο εκτέλεσης και το μέγεθος των δεδομένων που παράγουν οι τελεστές με έναν παράγοντα 50 και 1000 αντίστοιχα. Επίσης θέσαμε τη μνήμη των τελεστών στο 10% της χωρητικότητας των υποδοχέων, δεδομένου ότι δεν είχαν καθοριστεί από το αρχικό σημείο αναφοράς. Οι ιδιότητες των τελεστών του Lattice επιλέγεται ισοπίθανα από τα σύνολα τιμών του πίνακα 2.

Πίνακας 2 Ιδιότητες τελεστών

Ιδιότητα	Τιμές				
Χρόνος	0.20	0.40	0.60	0.80	1.00
CPU	0.40	0.45	0.50	0.55	0.60
Μνήμη	0.05	0.10	0.15	0.20	0.25
Έξοδος	0.20	0.40	0.60	0.80	1.00

Η χρήση του επεξεργαστή των τελεστών μεταφοράς είναι DT_{CPU} = 0,05 και η μνήμη DT_{MEM} = 0,001. Επίσης έχουμε πειραματιστεί με την αλλαγή του μεγέθους των δεδομένων που παράγονται με συντελεστές 1/0,64, 1/2,56, 1/40.96, και 1/10.485,76. Τέλος, οι ροές δεδομένων που έχουμε χρησιμοποιήσει είναι με όλους τους τελεστές Α&Π ή ΣΛ.

7.1.4 Αλγόριθμοι Βελτιστοποίησης

Για το εξωτερικό βρόχο του αλγόριθμου 1, ο αριθμός των ορίων των υποδοχέων είναι 10, 30, 50, 70, 90, 110, 130, και 150. Για το εσωτερικό βρόχο, έχουμε πειραματιστεί με όλους τους αλγορίθμους που περιγράφονται στο κεφάλαιο 6: άπληστοι (G-BRT, G-MNT, G-MPT, G-MPM) και προσομοιωμένης ανόπτησης (SA-MPT, SA-MPT2, SA-MPM, SA-MPM2). Επιπλέον, έχουμε τρέξει πειράματα με μια γεννήτρια τυχαίων χρονοπρογραμμάτων για να εξερευνήσουμε τον χώρο των λύσεων και να αποκτήσουμε γνώσεις σχετικά με τη φύση του προβλήματος βελτιστοποίησης. Στα επόμενα κεφάλαια παρουσιάζουμε τα ευρήματα από αυτά τα πειράματα, και ύστερα τα αποτελέσματα με τους αλγορίθμους βελτιστοποίησης.

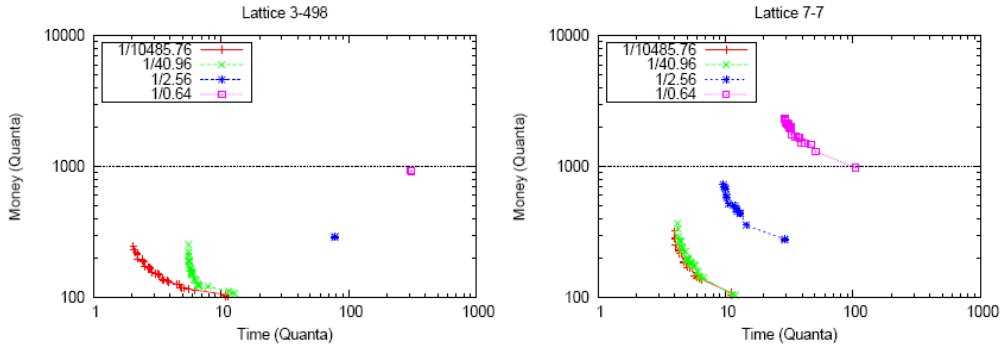
7.2 Εξερεύνηση του δισδιάστατου χώρου Χρόνου/Χρήματος

Η γεννήτρια τυχαίων χρονοπρογραμμάτων που χρησιμοποιήσαμε για αυτό το πείραμα παράγει 10.000 τυχαία χρονοδιαγράμματα για κάθε γράφο ροής δεδομένων που μελετήσαμε. Παρακάτω αναλύουμε τους τρόπους με τους οποίους επηρεάζεται ο χώρος

αναζήτησης από τα διάφορα χαρακτηριστικά της ροής δεδομένων ή του περιβάλλοντος εκτέλεσης.

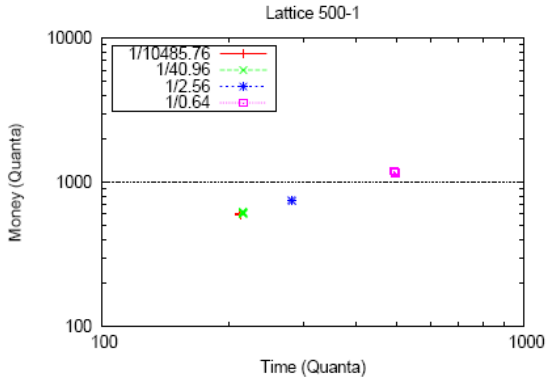
7.2.1 Μέγεθος Δεδομένων Εξόδου

Στην εικόνα 9, δείχνουμε την κορυφογραμμή για διάφορα μεγέθη εξόδου των ροών δεδομένων Lattice (3-498 και 7-7) και στην εικόνα 12 δείχνουμε την κορυφογραμμή του Ligo, Montage, και Cybershake. Η βασική παρατήρηση είναι ότι, για ένα συγκεκριμένο διάγραμμα ροής δεδομένων G, η ελαστικότητα αυξάνει όταν το μέγεθος των δεδομένων που παράγονται μειώνεται. Επίσης, για μικρό μέγεθος δεδομένων, η ελαστικότητα είναι μεγαλύτερη όταν ο βαθμός παραλληλισμού είναι μεγάλος ($E^{3\text{-}498}_{small} > E^{7\text{-}7}_{small}$). Σε ορισμένες περιπτώσεις το μέγεθος των δεδομένων δεν επηρεάζει τη ελαστικότητα (Lattice 7-7 στην εικόνα 9 και Ligo στην εικόνα 12).



Εικόνα 9: Οι κορυφογραμμές του Lattice A&Π για διαφορετικό μέγεθος δεδομένων (λογαριθμική κλίμακα).

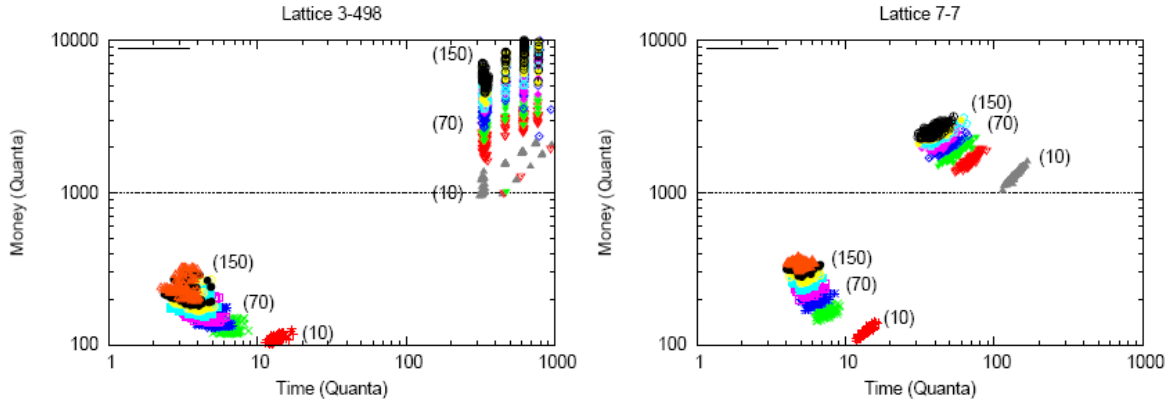
Η ελαστικότητα είναι απροσδιόριστη ($E^{3\text{-}498}_{large} = 0$) για τον γράφο ροής δεδομένων Lattice 3-498 με μεγάλο μέγεθος δεδομένων. Αυτό ισχύει και για τον γράφο Lattice 500-1 που δείχνουμε στο Σχήμα 10. Αυτό σημαίνει ότι δεν υπάρχουν συμβιβασμοί μεταξύ χρόνου και χρήματος, συνεπώς, το γρηγορότερο χρονοπρόγραμμα είναι και το φθηνότερο.



Εικόνα 10: Οι κορυφογραμμές του Lattice 500-1 Lattice για διαφορετικό μέγεθος δεδομένων.

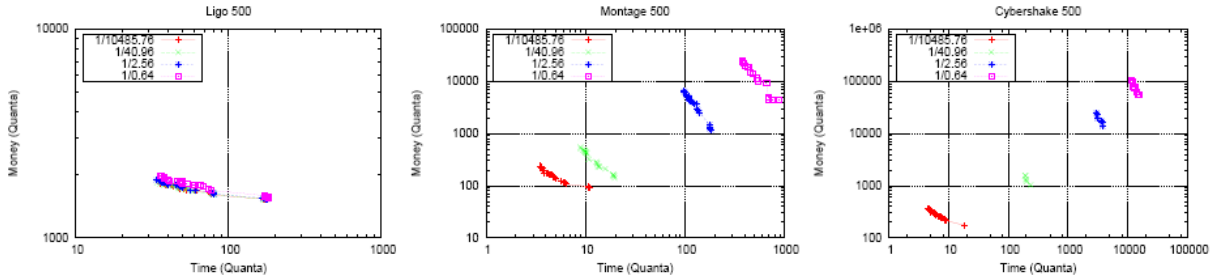
Στο Σχήμα. 11, δείχνουμε όλο τον χώρο των λύσεων (όχι μόνο την κορυφογραμμή) για τους ιδίους γράφους ροής δεδομένων, αλλά μόνο για μικρό (1 / 10.485,76) και μεγάλο (1 / 0,64) μέγεθος δεδομένων. Για τα μικρά μεγέθη δεδομένων, όσο περισσότερους υποδοχείς χρησιμοποιούμε, τόσο πιο γρήγορα ολοκληρώνονται τα χρονοπρογράμματα, δηλαδή, ο χρόνος και το χρήμα, είναι αντί-συσχετιζόμενα και αληθινή ελαστικότητα είναι

παρούσα. Για μεγάλα μεγέθη δεδομένων, σε ακραίες περιπτώσεις, συσχετίζονται και το ταχύτερο χρονοπρόγραμμα είναι και το φθηνότερο.



Εικόνα 11: Όλος ο χώρος των λύσεων για Lattice ροές για μικρό και μεγάλο μέγεθος δεδομένων.

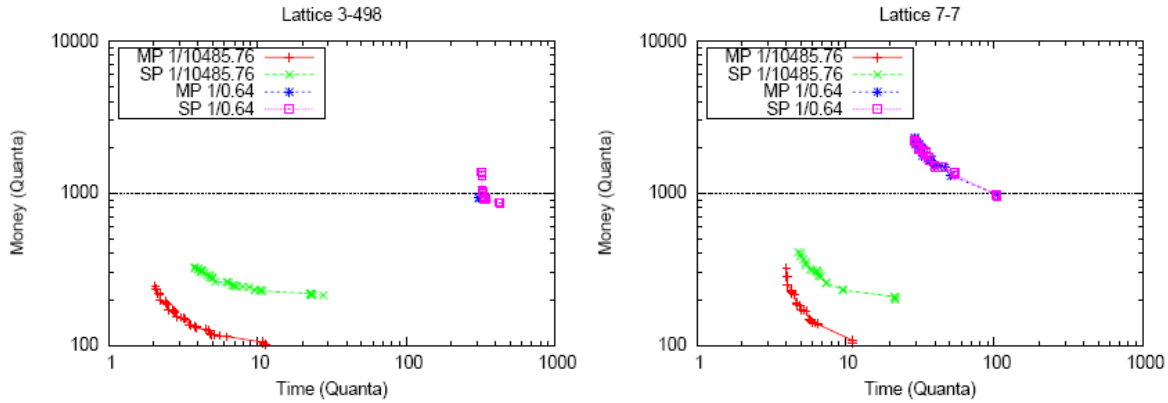
Στο Σχήμα. 12, δείχνουμε τη κορυφογραμμή των Ligo, Montage, και Cybershake. Παρατηρούμε ότι $E^{Ligo}_{small} > E^{Montage}_{small} > E^{Cybershake}_{small}$, δηλαδή, η ελαστικότητα είναι ανάλογη με το μέγεθος του παραλληλισμού που παρουσιάζει ο γράφος ροής δεδομένων. Θυμίζουμε ότι το Ligo έχει περισσότερο παραλληλισμό από το Montage, το οποίο έχει περισσότερο από το Cybershake.



Εικόνα 12: Οι κορυφογραμμές των Ligo, Montage, και Cybershake.

7.2.2 Πολύ-επεξεργασία - Μονό-επεξεργασία

Το μοντέλο μας επιτρέπει σε πολλούς τελεστές να τρέχουν ταυτόχρονα στο ίδιο υποδοχέα. Οι κορυφογραμμές των ροών δεδομένων Lattice που παράγονται με πολύ- και μονό-επεξεργασία φαίνονται στο Σχήμα 13. Χρονοπρογράμματα που παράγονται με μονό-επεξεργασία είναι πιο αργά και πιο ακριβά από τα αντίστοιχα που παράγονται από πολύ-επεξεργασία για μικρό μέγεθος εξόδου, ενώ δεν διαφέρουν πολύ για μεγάλο μέγεθος δεδομένων εξόδου.

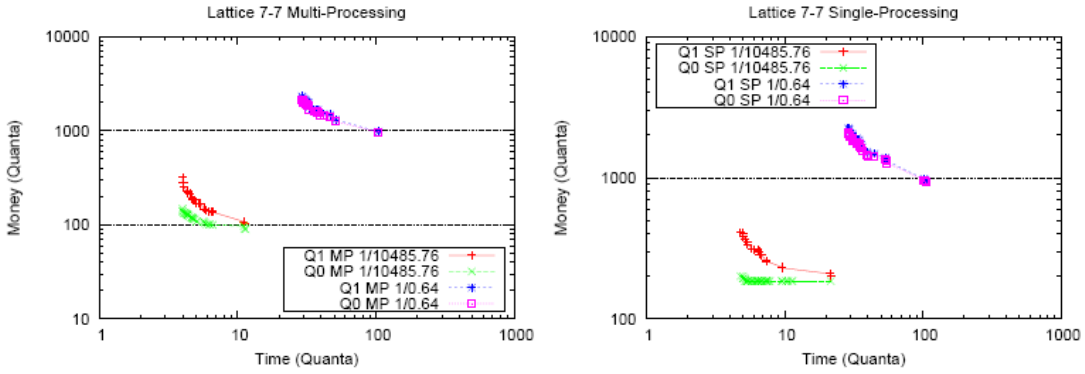


Εικόνα 13: Οι κορυφογραμμές με μονό-επεξεργασία και πολύ-επεξεργασία των Lattice ροών.

Ο κύριος λόγος που αυτό συμβαίνει είναι η υπό-χρησιμοποίηση του υποδοχέα όταν εκτελείται μόνο ένας τελεστής κάθε φορά. Η κυριαρχία του χρόνου χρήσης του δικτύου έναντι του χρόνου εκτέλεσης των τελεστών για τις μεγάλες εξόδους δεδομένων εξηγεί το γεγονός ότι η κορυφογραμμές δεν διαφέρουν πολύ.

7.2.3 Μέγεθος Κβάντου Χρέωσης του Νέφους

Πειραματιστήκαμε με το μέγεθος του κβάντου του χρόνου που χρεώνει το νέφος. Στην εικόνα 14, δείχνουμε την κορυφογραμμή των ροών δεδομένων Lattice για μέγεθος κβάντου ε (πολύ μικρό, κοντά στο 0) και 1 με χρήση πολύ- και μονό-επεξεργασίας. Παρατηρούμε ότι το μέγεθος του κβάντου επηρεάζει περισσότερο τα ταχύτερα χρονοπρογράμματα (αυτά που χρησιμοποιούν πολλούς υποδοχείς) από ότι τα πιο αργά (με λίγους υποδοχείς). Αυτό ήταν αναμενόμενο αφού, σε γενικές γραμμές, όσο περισσότερους υποδοχείς χρησιμοποιούμε, τόσο μεγαλύτερος είναι ο κατακερματισμός. Στην εικόνα 14, παρατηρούμε ότι αν χρησιμοποιούμε μονό-επεξεργασία, η ελαστικότητα είναι πολύ υψηλή ($E^{7\text{-}7}_{small} \approx (T_{max}\text{-}T_{min})/0$), έτσι, το ταχύτερο και το πιο αργό χρονοπρόγραμμα έχουν περίπου το ίδιο κόστος. Ως εκ τούτου, για τις συγκεκριμένες αυτές περιπτώσεις, το πρόβλημα βελτιστοποίησης γίνεται μονοδιάστατο.

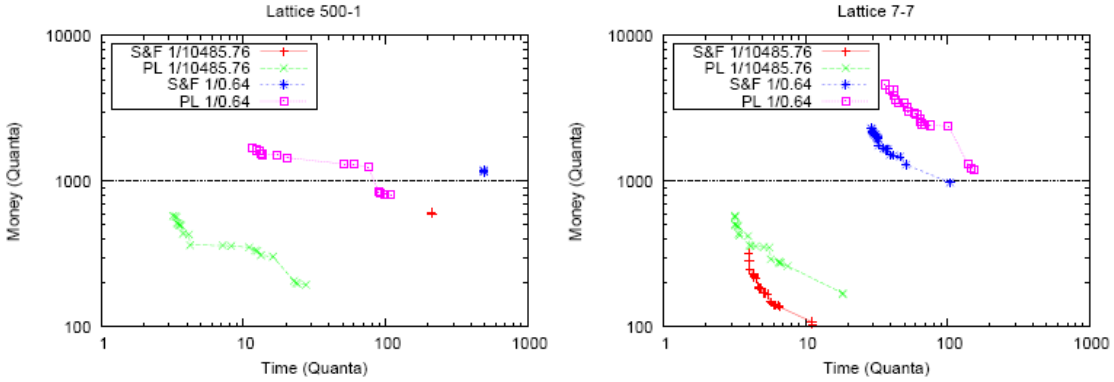


Εικόνα 14: Οι κορυφογραμμές των Lattice ροών με κβάντο 0 και 1.

7.2.4 Σωλήνωση – Αποθήκευση και Προώθηση

Στο Σχήμα 15, παρατηρούμε ότι ο γράφος ροής δεδομένων Lattice 500-1 με τελεστές ΣΛ έχει ελαστικότητα, ενώ αυτό με τελεστές Α&Π δεν έχει. Επιπλέον, τα χρονοπρογράμματα του πρώτου γράφου είναι ταχύτερα και κάποια από αυτά

φθηνότερα. Από την άλλη πλευρά, τα χρονοδιαγράμματα για το Lattice 7-7 με τελεστές σωλήνωσης είναι πιο αργά και πιο ακριβά από ότι αυτά με τελεστές S&F. Αυτό συμβαίνει για δύο λόγους: 1) λόγω της πλήρης επικάλυψης στην εκτέλεση των τελεστών σωλήνωσης, όλοι οι τελεστές συγχρονίζονται με αυτό που έχει τη μεγαλύτερη διάρκεια εκτέλεσης, και 2) λόγω της επικάλυψης της μεταφοράς δεδομένων με την επεξεργασία, οι αδιαίρετοι πόροι (μνήμη) που καταναλώνεται από τους τελεστές, δεσμεύετε για μεγαλύτερο χρονικό διάστημα. Αυτό εξηγεί επίσης και το γεγονός ότι για μεγάλο αριθμό υποδοχέων, τα χρονοδιαγράμματα του Lattice 7-7 είναι πιο γρήγορα από αυτά που έχουν τελεστές Α&Π, επειδή περισσότερη μνήμη είναι διαθέσιμη. Δύο συνδεδεμένοι τελεστές ΣΛ οι οποίοι έχουν ανατεθεί στον ίδιο υποδοχέα θα εκτελεστούν γρηγορότερα από δύο Α&Π με τα ίδια χαρακτηριστικά, αλλά όταν βρίσκονται σε διαφορετικούς υποδοχείς, αυτό ίσος να μην συμβαίνει.



Εικόνα 15: Οι κορυφογραμμές των Lattice ροών με τελεστές Α&Π και ΣΛ.

7.2.5 Γενικά συμπεράσματα

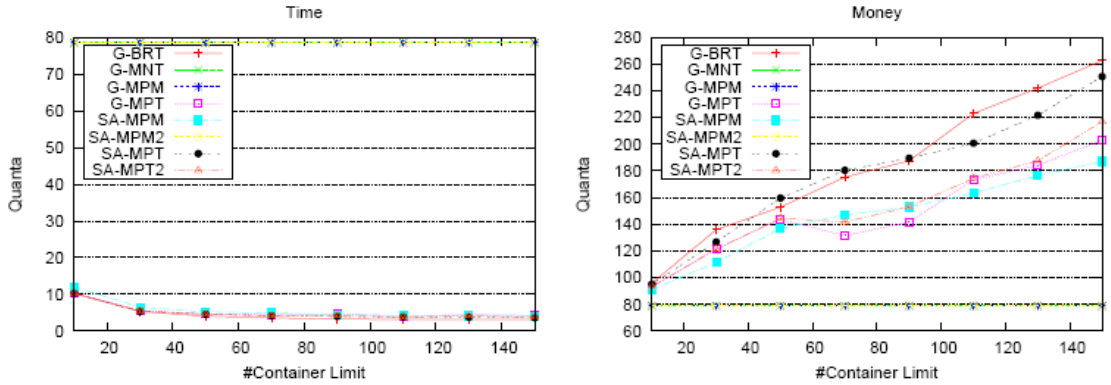
Με βάση την προηγούμενη συζήτηση, μπορούν να εξαχθούν τα ακόλουθα συμπεράσματα σχετικά με το χώρο χρόνου/χρήματος των εναλλακτικών χρονοπρογραμμάτων:

1. Υπάρχει συσχετισμός μεταξύ του χρόνου ολοκλήρωσης ενός τελεστή και του χρόνου που δαπανάται για τη μεταφορά δεδομένων. Για μεγάλες τιμές του κλάσματος χρόνος/δεδομένα ο χρόνος ολοκλήρωσης και το χρηματικό κόστος είναι αντί-συσχετισμένα και ο βαθμός της ελαστικότητας είναι υψηλός. Από την άλλη πλευρά, για μικρές τιμές του κλάσματος αυτού, ο χρόνος ολοκλήρωσης και το χρηματικό κόστος συσχετίζονται και υπάρχει περιορισμένη ελαστικότητα. Για αυτές τις περιπτώσεις, η επίλυση ενός μονοδιάστατου προβλήματος βελτιστοποίησης είναι αρκετό. Για παράδειγμα, γραμμικές ροές δεδομένων με τελεστές Α&Π δεν είναι ελαστικές, ενώ αυτά με τελεστές ΣΛ είναι.
2. Η πολύ-επεξεργασία, σε γενικές γραμμές, παράγει ταχύτερα και φθηνότερα χρονοπρογράμματα σε σύγκριση με την μονό-επεξεργασία.
3. Η σωλήνωση θα πρέπει να χρησιμοποιείται με προσοχή, καθώς μπορεί να προσθέσει σημαντική επιβάρυνση τόσο στον χρόνο ολοκλήρωσης όσο και το χρηματικό κόστος.
4. Ο κατακερματισμός των πόρων προκαλεί σημαντικές αυξήσεις στο χρηματικό κόστος, ειδικά όταν χρησιμοποιούνται πολλοί υποδοχείς.

5. Υπάρχουν διαφορετικοί τύποι ελαστικότητας για διάφορα είδη ροών δεδομένων. Μερικές ροές είναι πιο ελαστικές από άλλες, και μερικές δεν έχουν καθόλου ελαστικότητα.

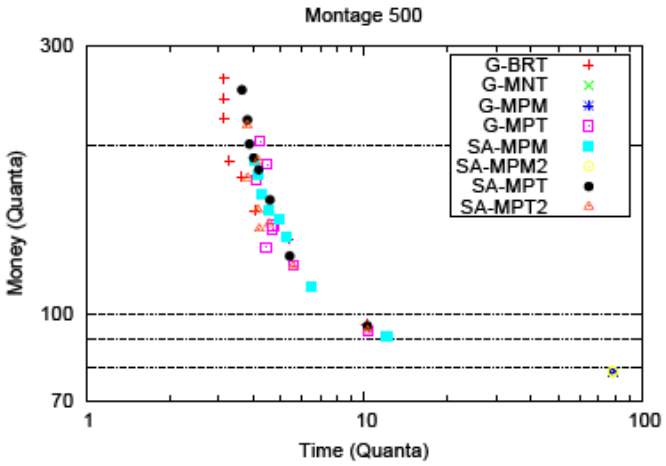
7.3 Αλγόριθμοι Βελτιστοποίησης

Στην εικόνα 16, παρουσιάζουμε τα χρονοπρογράμματα που παράγονται από τις διαδοχικές κλήσεις των αλγορίθμων με διαφορετικό πλήθος υποδοχέων στο Montage με τελεστές Α&Π με μικρό μέγεθος δεδομένων. Βλέπουμε ότι υπάρχει ελαστικότητα μεταξύ χρόνου και χρήματος. Είναι σαφές ότι ο χρόνος ολοκλήρωσης μειώνεται όταν ο αριθμός των υποδοχέων αυξάνει. Το χρηματικό κόστος τείνει να αυξάνεται όσο αυξάνετε το πλήθος των υποδοχέων, αλλά όχι πάντα. Υπάρχουν δύο λόγοι που αυτό συμβαίνει. Πρώτον, η διαθεσιμότητα των υποδοχέων δεν συνεπάγεται και τη χρήση τους. Ο μέγιστος παραλληλισμός του χρονοπρογράμματος μπορεί να είναι μικρότερος από τον αριθμό των διαθέσιμων υποδοχέων ή το χρονοπρόγραμμα που προσδιορίζονται ως βέλτιστο από τον αλγόριθμο να μην χρησιμοποιεί όλο το διαθέσιμο παραλληλισμό. Δεύτερον, λόγω της κβαντισμένης τιμολογιακής πολιτικής του νέφους, η μέση χρήση των κβάντων είναι καλύτερη και το παραγόμενο χρονοπρόγραμμα πιο συμπαγές.



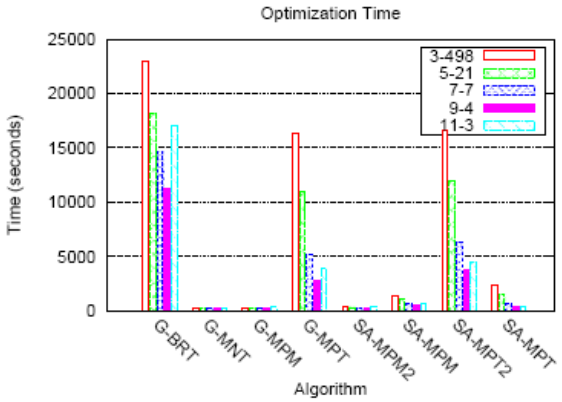
Εικόνα 16: Ο χρόνος και το χρήμα σαν συνάρτηση του αριθμού των υποδοχέων για διάφορους αλγορίθμους στο Montage.

Στην εικόνα 17, δείχνουμε το χώρο του χρόνου/χρήματος για το Montage και την κορυφογραμμή που παράγεται από διάφορους αλγορίθμους. Είναι ενδιαφέρον ότι, όταν αυτές συγχωνεύονται, το αποτέλεσμα έχει χρονοπρογράμματα από διαφορετικούς αλγόριθμους. Τα χρονοπρογράμματα που παράγει ο SA-MPM δίνουν φτηνά χρονοπρογράμματα, ο SA-MPT δίνει πιο γρήγορες λύσεις από ότι ο SA-MPM αλλά πιο ακριβές, και ο G-BRT δίνει τις ταχύτερες και πιο δαπανηρές λύσεις. Παρατηρούμε επίσης ότι ο G-MNT (ελαχιστοποίηση της κίνησης του δικτύου) δίνει λύσεις παρόμοιες με τον G-MPM χρησιμοποιώντας μόνο έναν μικρό αριθμό υποδοχέων. Ο αλγόριθμος SA-MPT ξεκινά με μια τυχαία ανάθεση και η λύσεις που βρίσκει είναι πολύ κοντά με εκείνων που παράγονται από τον G-MPT. Οι αλγόριθμοι G-MPM και SA-MPM2 δεν παράγουν πολλαπλές λύσεις. Αυτό συμβαίνει γιατί όλοι οι τελεστές είναι Α&Π και ως εκ τούτου, το φθηνότερο χρονοπρόγραμμα είναι πάντα αυτό που χρησιμοποιεί έναν μόνο υποδοχέα. Από την άλλη πλευρά, ο SA-MPM, ξεκινώντας με ένα τυχαίο χρονοπρόγραμμα, δίνει λύσεις που έχουν ελαστικότητα.



Εικόνα 17: Τα χρονοπρογράμματα που παράγουν διάφοροι αλγόριθμοι για το Montage.

Στην εικόνα 18, δείχνουμε τον χρόνο εκτέλεσης των διαφόρων αλγορίθμων για διάφορους τύπους Lattice ροών δεδομένων με τελεστές Α&Π. Ξεκάθαρα, ο βαθμός παραλληλισμού των γράφων επηρεάζει τον χρόνο λειτουργίας των αλγορίθμων, μερικούς λίγο περισσότερο (π.χ. G-MPT, SA-MPT2) από άλλους (π.χ. G-BRT, SA-MPT).



Εικόνα 18: Οι χρόνοι βελτιστοποίηση όλων των αλγορίθμων για τις Lattice ροές.

Δεδομένου ότι ο G-BRT αναθέτει ομοιόμορφα τους τελεστές σε υποδοχείς, ο αναμενόμενος αριθμός των υποδοχέων που χρησιμοποιεί είναι κοντά στους διαθέσιμους. Όπως είναι αναμενόμενο, αυτός ο αλγόριθμος είναι κοντά στη χειρότερη περίπτωση μεταξύ των άπληστοι αλγόριθμων.

7.3.1 Γενικά συμπεράσματα

Οι κυριότερες παρατηρήσεις για τη συμπεριφορά των διαφόρων αλγορίθμων βελτιστοποίησης είναι οι εξής:

1. Διαφορετικοί αλγόριθμοι τείνουν να εξερευνήσουν διαφορετικές περιοχές του χώρου. Μερικοί από αυτούς δημιουργούν χρονοπρογράμματα που παρουσιάζουν ενδιαφέροντες συμβιβασμούς που κανένας άλλος αλγόριθμος δεν μπορεί να βρει. Αυτό δημιουργεί ένα ενδιαφέρον μετα-ερώτημα, δηλαδή πως μπορεί να γίνει η επιλογή του βέλτιστου αλγόριθμου βελτιστοποίησης ανάλογα με τις ανάγκες των χρηστών. Το ερώτημα αυτό αφήνεται για μελλοντική εργασία.

2. Εξελιγμένες μέθοδοι αναζήτησης, όπως η προσομοιωμένη ανόπτηση, δεν φαίνεται να βελτιώνουν σημαντικά τα χρονοπρογράμματα που παράγονται από άπληστους αλγόριθμους, ενώ απαιτούν πολύ περισσότερο χρόνο για να εκτελεστούν. Ο χώρος των εναλλακτικών λύσεων είναι τεράστιος και περισσότερη μελέτη πρέπει να γίνει στην εξεύρεση τρόπων για αποτελεσματική εξερεύνησή του. Για τις μεγάλες ροών δεδομένων, άπληστοι αλγόριθμοι φαίνεται να είναι η σωστή επιλογή.

8. ΣΧΕΤΙΚΗ ΕΡΓΑΣΙΑ

Συνήθως, κάποιο ενδιάμεσο λογισμικό (middleware) χρησιμοποιείται για να εκτελέσει κώδικα χρηστών σε κατανεμημένα περιβάλλοντα. Ο συνδυασμός των συστημάτων Condor / DAGMan / Stork [20] είναι η κυρίαρχη τεχνολογία του υπολογισμού υψηλής απόδοσης (High Performance Computing). Παρ' όλα αυτά, το Condor σχεδιάστηκε για να εκμεταλλεύεται τους επεξεργαστές αχρησιμοποίητων υπολογιστών. Η επεξεργασία ροών δεδομένων που παράγουν πολλά δεδομένα χρησιμοποιώντας το DAGMan είναι πολύ αναποτελεσματική [26]. Πολλά συστήματα χρησιμοποιούν το DAGMan σαν ενδιάμεσο λογισμικό, όπως το Pegasus [6] και το GridDB [21]. Προτάσεις για επεκτάσεις του Condor για να μπορεί να εκτελέσει επιστημονικές ροές δεδομένων που παράγουν πολλά δεδομένα υπάρχουν [26], αλλά απ όσο γνωρίζουμε, δεν έχουν υλοποιηθεί ακόμη. Στο [9] παρουσιάζεται μια μελέτη της εκτέλεσης του Montage στο νέφος εξετάζοντας τους συμβιβασμούς των διαφορετικών τρόπων εκτέλεσής τους στο νέφος.

Το Hadoop είναι μια πλατφόρμα που υλοποιεί το πρότυπο MapReduce [5] για την επίτευξη ανοχής σε σφάλματα και μαζικού παραλληλισμού. Πολλές γλώσσες ερωτημάτων υψηλού επιπέδου έχουν προταθεί και υλοποιηθεί πάνω από το Hadoop, για παράδειγμα, το PigLatin [22] και το Hive [28] (που χρησιμοποιείται από το Facebook), και το ίδιο ισχύει για πολλές εφαρμογές, π.χ., το Mahout [3], η οποία είναι μια πλατφόρμα μηχανικής μάθησης με παράλληλους αλγορίθμους. Τα διαγράμματα ροής δεδομένων που χρησιμοποιούνται στο MapReduce είναι σχετικά περιορισμένα, δεδομένου ότι είναι κυρίως Lattice 3-N, για κάποιο N, και αυτό μειώνει τις ευκαιρίες για βελτιστοποίηση.

Το Dryad [15] είναι ένα εμπορικό ενδιάμεσο λογισμικό της Microsoft που έχει μια πιο γενική αρχιτεκτονική από το MapReduce, καθώς μπορεί να παραλληλοποίησει οποιοδήποτε γράφο ροής δεδομένων. Η βελτιστοποίηση που χρησιμοποιεί ωστόσο, εξαρτάται σε μεγάλο βαθμό από υποδείξεις (hints) που απαιτούν τη γνώση των κόμβων που θα εκτελεστούν οι τελεστές, κάτι το οποίο γενικά δεν είναι διαθέσιμο σε περιβάλλον νέφους. Ασχολείται επίσης με τη μετεγκατάσταση των τελεστών εκτελώντας ένα άλλο αντίγραφο του σε κάποιον άλλο κόμβο. Αυτό μπορεί να είναι αποδεκτό αν γίνετε βελτιστοποίηση μόνο ως προς τον χρόνο, αλλά όχι όταν το οικονομικό κόστος μετράει.

Η Νεφέλη [29] είναι μια πύλη νέφους που χρησιμοποιεί υποδείξεις για την αποτελεσματική εκτέλεση του φόρτου εργασίας ενός χρήστη. Χρησιμοποιεί το σύννεφο σε χαμηλότερο επίπεδο από ότι αυτό που προτείνουμε εμείς, έχοντας επίγνωση των φυσικών πόρων και της πραγματικής θέση των εικονικών μηχανών. Αυτή η πληροφορία μπορεί να μην είναι γενικά διαθέσιμη, ιδίως στις εμπορικά νέφη.

Υπάρχουν επίσης πολλές προσπάθειες, οι οποίες κινούνται προς την ίδια κατεύθυνση με την δουλειά μας, αλλά προσπαθούν να λύσουν απλούστερες εκδόσεις του προβλήματος. Παραδείγματα περιλαμβάνουν βελτιστοποίηση με χρήση σμήνους σωματιδίων που χρησιμοποιεί ένα μονοδιάστατο σταθμισμένο μέσο όρο των παραμέτρων για κριτήριο βελτιστοποίησης [23] και μια ευριστική βελτιστοποίηση ανεξάρτητων εργασιών που βρίσκει τον αριθμό των μηχανημάτων που θα πρέπει να διατεθούν για τη μεγιστοποίηση του χρόνου εκτέλεσης δοθέντος ενός προκαθορισμένου προϋπολογισμού [27].

Τέλος, σε θεμελιώδες επίπεδο, έχουμε ακολουθήσει το μοντέλο παραλληλισμού και κοινής χρήσης πόρων για βέλτιστο χρονοπρογραμματισμό σε σχεσιακούς τελεστές [11] και το έχουμε γενικεύσει για αυθαίρετους τελεστές.

Εν ολίγοις, έχουμε αξιοποιήσει την ελαστικότητα που προσφέρει το νέφους ώστε να παράγουμε πολλαπλά χρονοπρογράμματα, που επιτρέπουν στον χρήστη να επιλέξει τον επιθυμητό συμβιβασμό μεταξύ χρόνου και χρήματος. Απ όσο γνωρίζουμε, κανένα διαθέσιμο σύστημα που επεξεργάζεται ροές δεδομένων δεν ασχολείται με την έννοια της ελαστικότητας ή των δύο διαστάσεων του χρόνου και χρήματος τα οποία αποτελούν τις βασικές καινοτομίες μας.

9. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

Σε αυτήν την εργασία παρουσιάσαμε ένα πλαίσιο για τρία διαφορετικά προβλήματα βελτιστοποίησης γράφων ροής δεδομένων για το νέφος, για ένα δισδιάστατο κριτήριο βελτιστοποίησης του χρόνου και του χρηματικού κόστους, καθώς και μια μεθοδολογία για την επίλυση αυτών των προβλημάτων. Τα πειραματικά αποτελέσματα έχουν αποκαλύψει ενδιαφέρουσες ιδιότητες του χώρου των εναλλακτικών λύσεων που πρέπει να διερευνηθούν σχετικά με τα διάφορα επίπεδα της ελαστικότητας που προσφέρονται από διαφορετικού τύπους ροών δεδομένων, τα χαρακτηριστικά των τελεστών, και άλλες παραμέτρους, καθώς και για την αποτελεσματικότητα και την αποδοτικότητα των διαφόρων αλγορίθμων βελτιστοποίησης.

Η μεθοδολογία που προτείνουμε χρησιμοποιείται στο τελικό στάδιο της βελτιστοποίησης του συστήματος ADP και όλες οι παράμετροι αρχικοποιούνται αυτόματα χρησιμοποιώντας στατιστικά δεδομένα που συλλέγονται κατά τη διάρκεια προηγούμενων εκτελέσεων. Επίσης, αυτή η τεχνική μπορεί να χρησιμοποιηθεί ξεχωριστά, βοηθώντας τους παρόχους υπηρεσιών νέφους να παρέχουν συμβουλές στους καταναλωτές σχετικά με την επιλογή του κατάλληλου συμβιβασμού μεταξύ χρόνου και χρήματος.

Τα σχέδιά μας για μελλοντική δουλειά κινούνται σε διάφορες κατευθύνσεις. Έχουμε την πρόθεση να μελετήσουμε περαιτέρω την έννοια της ελαστικότητας και να προσδιορίσουμε την ευαισθησία της σε διαφορετικά φορτία και χαρακτηριστικά περιβάλλοντος εκτέλεσης, καθώς και την ικανότητα των αλγορίθμων βελτιστοποίησης για να την εκμεταλλευτούν. Σχεδιάζουμε επίσης να αξιολογήσουμε το ίδιο το μοντέλο σχετικά με το πόσο καλός είναι στην πρόβλεψη του πραγματικού φόρτου εργασίας. Επιπλέον, σχεδιάζουμε να διερευνήσουμε το πρόβλημα της κλιμάκωσης που παρουσιάζουν οι μεγάλοι γράφοι χρησιμοποιώντας προσαρμοστικές τεχνικές (adaptive techniques). Τέλος, έχουμε την πρόθεση να συγκρίνουμε την προσέγγισή μας με τις μεθόδους βελτιστοποίησης άλλων κατανεμημένων συστημάτων, όπως το σύστημα Hadoop.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Virtualization	Εικονικοποίηση
Scheduling	Χρονοπρογραμματισμός
Heuristic	Ευριστικό
Cluster	Συστοιχίες Υπολογιστών
Grid	Πλέγμα
Container	Υποδοχέας
Schedule	Χρονοπρόγραμμα
Time-Shared	Διαμοιραζόμενος πόρος
Space-Shared	Αδιαίρετος πόρος
Skyline	Κορυφογραμμή
Bottleneck	Σημείο συμφόρησης

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ACM	Association for Computing Machinery
SQL	Structured Query Language
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
QoS	Quality of Service
DAG	Directed Acyclic Graph

ΑΝΑΦΟΡΕΣ

[1] R. Agrawal et al. "The Claremont report on database research". SIGMOD Record, 37(3):9–19, 2008.
[2] Amazon. "Amazon Elastic Compute Cloud (EC2), http://aws.amazon.com/ec2/", 2010.
[3] Apache. "Mahout : Scalable machine-learning and data-mining library, http://mahout.apache.org/", 2010.
[4] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi. Characterization of scientific workflows. pp. 1–10, nov. 2008.
[5] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In 6th Symposium on Operating System Design and Implementation, pp. 137–150, 2004.
[6] E. Deelman and e. al. "Pegasus: Mapping Large Scale Workflows to Distributed Resources in Workflows in e-Science". Springer, 2006.
[7] E. Deelman et al. Managing large-scale workflow execution from resource provisioning to provenance tracking: The cybershake example. In IEEE e-Science, pp. 14, 2006.
[8] E. Deelman, C. Kesselman, and more. "GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists". In IEEE HPDC, pp. 225–, 2002.
[9] E. Deelman, G. Singh, M. Livny, G. B. Berriman, and J. Good. The cost of doing science on the cloud: the montage example. In IEEE/ACM SC, pp. 50, 2008.
[10] I. T. Foster. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations". In CCGRID, pp. 6–7, 2001.
[11] M. N. Garofalakis and Y. E. Ioannidis. "Parallel Query Scheduling and Optimization with Time- and Space-Shared Resources". In VLDB, pp. 296–305, 1997.
[12] L. M. V. Gonzalez, L. R. Merino, J. Caceres, and M. Lindner. "A break in the clouds: towards a cloud definition". Computer Communication Review, 39(1):50–55, 2009.
[13] R. L. Graham. "Bounds on Multiprocessing Timing Anomalies". SIAM Journal of Applied Mathematics, 17(2):416–429, 1969.
[14] Y. E. Ioannidis and E. Wong. "Query Optimization by Simulated Annealing". In SIGMOD Conference, pp. 9–22, 1987.
[15] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. "Dryad: distributed data-parallel programs from sequential building blocks". In EuroSys, pp. 59–72, 2007.
[16] J. C. Jacob et al. "Montage: a grid portal and software toolkit for science, grade astronomical image mosaicking". Int. J. Comput. Sci. Eng., 4(2):73–87, 2009.
[17] J. Kleinberg and E. Tardos. "Algorithm Design". Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.
[18] D. Kossmann. "The state of the art in distributed query processing". ACM Comput. Surv., 32(4):422–469, 2000.
[19] Y.-K. Kwok and I. Ahmad. "Benchmarking and Comparison of the Task Graph Scheduling Algorithms". J. Parallel Distrib. Comput., 59(3):381–422, 1999.
[20] M. J. Litzkow, M. Livny, and M. W. Mutka. "Condor – A Hunter of Idle Workstations". In ICDCS, pp. 104–111, 1988.
[21] D. T. Liu and M. J. Franklin. "The Design of GridDB: A Data-Centric Overlay for the Scientific Grid". In VLDB, pp. 600–611, 2004.
[22] C. Olston et al. "Pig latin: a not-so-foreign language for data processing". In SIGMOD Conference, pp. 1099–1110, 2008.
[23] S. Pandey, L. Wu, S. M. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In IEEE AINA, pp. 400–407, 2010.
[24] C. H. Papadimitriou and M. Yannakakis. "Multiobjective Query Optimization". In PODS, 2001.
[25] G. M. R., J. D. S., and S. Ravi. "The Complexity of Flowshop and Jobshop Scheduling". Mathematics of operations research, 1(2):117–129, 1976.
[26] S. Shankar and D. J. DeWitt. "Data driven workflow planning in cluster management systems". In HPDC, pp. 127–136, 2007.
[27] J. N. Silva, L. Veiga, and P. Ferreira. Heuristic for resources allocation on utility computing infrastructures. In B. Schulze and G. Fox, editors, MGC, pp. 9. ACM, 2008.
[28] A. Thusoo et al. "Hive - a petabyte scale data warehouse using Hadoop". In ICDE, pp. 996–1005, 2010.
[29] K. Tsakalozos, M. Roussopoulos, V. Floros, and A. Delis. Nefeli: Hint-based execution of workloads in clouds. In IEEE ICDCS, pp. 74–85, 2010.
[30] M. M. Tsangaris and more. Dataflow processing and optimization on grid and cloud infrastructures. IEEE Data Eng. Bull., 32(1):67–74, 2009.
[31] H. R. Varian. "Intermediate Microeconomics : A Modern Approach", chapter 15, Market Demand. W. W. Norton and Company, 7th edition, Dec. 2005.