Ατομική Διπλωματική Εργασία

ΜΕΛΕΤΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΚΑΙ ΠΕΡΙΟΡΙΣΜΩΝ ΔΙΑΦΟΡΩΝ ΜΗΧΑΝΩΝ

Αντωνιάδου Μαρίνα

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2013

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη χαρακτηριστικών και περιορισμών διαφόρων μηχανών

Αντωνιάδου Μαρίνα

Επιβλέπων Καθηγητής

Pedro Trancoso

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2013

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Pedro Trancoso, για την καθοδήγηση και τη βοήθεια που μου παρείχε κατά τη διάρκεια της υλοποίησης της ατομικής διπλωματικής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω θερμά τους Διδακτορικούς φοιτητές Ανδρέα Διαβαστό και Κωνσταντίνο Χριστοφή για όλη τη βοήθεια που μου παρείχαν, για το χρόνο που αφιέρωσαν για την επίλυση οποιονδήποτε αποριών μου και για τις πολύτιμες συμβουλές που μου έδωσαν.

Ένα μεγάλο ευχαριστώ στη φίλη μου Άντρη, για την στήριξη και τις γνώσεις που μου παρείχε, καθώς και στην Ξάνθη για τις συμβουλές της και το χρόνο που αφιέρωσε για να με βοηθήσει.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου και τα αδέρφια μου, και ιδιαίτερα τις αδερφές μου Ελένη και Αντριανή, που ήταν πάντα δίπλα μου και με βοηθούσαν, με στήριζαν και με συμβούλευαν.

Περίληψη

Το θέμα μου είναι η μελέτη των χαρακτηριστικών και των τυχόν περιορισμών που μπορεί να έχουν τρεις διαφορετικές μηχανές και η σύγκριση των μηχανών αυτών ανάλογα με τα αποτελέσματα που θα αποκομίσω.

Οι μηχανές που χρησιμοποίησα για την εργασία μου είναι η μηχανή csteras, η μηχανή cs6472 από τις μηχανές ADE του Πανεπιστημίου Κύπρου και η μηχανή csatom. Κάθε μηχανή έχει διαφορετικά χαρακτηριστικά.

Η μελέτη γίνεται με τη χρήση του συστήματος διαχείρισης βάσεων δεδομένων PostgreSQL, των performance counters του υποσυστήματως perf των linux και των TPC-H benchmarks. Με την εκτέλεση των queries που προσφέρουν τα TPC-H benchmarks και τη χρήση των performance counters θα πάρω διάφορες μετρήσεις, τις οποίες χρησιμοποιήσα μαζί με τα χαρακτηριστικά των μηχανών για σύγκριση των μηχανών.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο

Το πώς τα χαρακτηριστικά μιας μηχανής επηρεάζουν την επίδοσή της είναι ένα θέμα που απασχολεί όσους ασχολούνται με υπολογιστές. Ακόμη και για την αγορά ενός απλού υπολογιστή, που θα χρησιμοποιείται για απλές εργασίες όπως πλοήγηση στο ίντερνετ και χρήση μη εξειδικευμένων προγραμμάτων, ο χρήστης θα πρέπει να αποφασίσει ποια είναι η κατάλληλη αγορά για εκείνον, σύμφωνα με τα χαρακτηριστικά που διαθέτει ο υπολογιστής. Για πιο πολύπλοκες εργασίες, ο χρήστης πρέπει να είναι ακόμη πιο επιλεκτικός και σίγουρος ότι η μηχανή μπορεί να του προσφέρει ότι χρειάζεται. Δυστυχώς, κανένας δεν μπορεί να εγγυηθεί ότι συγκεκριμένα χαρακτηριστικά θα προσφέρουν την επίδοση που περιμένουμε. Γι' αυτό έχουν δημιουργηθεί διάφορα εργαλεία για την ανάλυση της επίδοσης ενός υπολογιστή και την εξέταση του πώς τα χαρακτηριστικά του επηρεάζουν αυτή την επίδοση.

Για την εργασία μου επέλεξα να χρησιμοποιήσω ένα από αυτά τα εργαλεία, το TPCH-Benchmark, ώστε να μελετήσω τα χαρακτηριστικά τριών διαφορετικών μηχανών και το πώς τα χαρακτηριστικά αυτά επηρεάζουν την επίδοση κάθε μηχανής. Για τη χρήση του εργαλείου αυτού χρησιμοποίησα επίσης το σύστημα διαχείρησης βάσεων δεδομένων PostgreSQL. Επίσης, με τη χρήση του υποσυστήματος perf, συνέλεξα κάποιες μετρήσεις οι οποίες με βοήθησαν στη διεξαγωγή της σύγκρισης των τριών μηχανών και την εξαγωγή κάποιων συμπερασμάτων σχετικά με το πώς συμπεριφέρεται κάθε μηχανή ανάλογα με τα χαρακτηριστικά της.

Οι μετρήσεις που πήρα είναι τα Cache references και τα Cache misses, τα οποία χρησιμοποιώ για υπολογισμό του Cache miss rate και ο χρόνος εκτέλεσης. Σε επόμενο κεφάλαιο θα αναφέρω περισσότερες πληροφορίες για αυτές τις μετρήσεις.
Οι μηχανές που χρησιμοποιώ για την εργασία μου είναι η μηχανή csteras, η μηχανή cs6472 από τις μηχανές ADE του Πανεπιστημίου Κύπρου και ο φορητός υπολογιστής csatom, οι οποίες επιλέχθηκαν λόγω των διαφορετικών χαρακτηριστικών τους, στα οποία θα ανφερθώ σε επόμενο κεφάλαιο.

1.2 Οργάνωση Διπλωματικής Εργασίας

Η παρούσα ατομική διπλωματική εργασία αποτελείται από 6 κεφάλαια.

Στο 1^{ο} κεφάλαιο μίλησα εισαγωγικά για το θέμα της εργασίας μου και το κίνητρο πίσω από την επιλογή του.

Στο 2^{ο} κεφάλαιο θα μιλήσω για το λογισμικό που χρησιμοποίησα για την εκπόνηση της διπλωματικής μου, δηλαδή για την PostgreSQL, τα queries των TPC-H benchmarks και το perf, και το λόγο που χρησιμοποίησα το συγκεκριμένο λογισμικό.

Στο 3^{ο} κεφάλαιο θα μιλήσω για τις τρεις μηχανές που χρησιμοποίησα και για τα χαρακτηριστικά της κάθε μηχανής τα οποία θα χρησιμοποιήσω για την σύγκρισή τους.

Στο 4^{ο} κεφάλαιο θα υπάρξει μια περιγραφή των μετρήσεων που πήρα.

Στο 5^{ο} κεφάλαιο θα μιλήσω για τη μεθοδολογία που ακολούθησα και στη συνέχεια θα παρουσιάσω τα αποτελέσματα από τις διάφορες μετρήσεις που πήρα.

Στο 6^{ο} κεφάλαιο θα παραθέσω τα συμπεράσματα που εξήγαγα και θα μιλήσω για την πιθανή μελλοντική δουλειά που μπορεί να γίνει με βάση την εργασία μου.

Κεφάλαιο 2

Λογισμικό

2.1 PostgreSQL

Η PostgreSQL[1] είναι ένα αντικειμενοστρεφές σύστημα διαχείρισης βάσεων δεδομένων, βασισμένο στην τελευταία έκδοση της POSTGRES, την έκδοση 4.2 από το πανεπιστήμιο της California, του Berkeley Computer Science.

Η PostgreSQL είναι open source και τρέχει σε όλα τα κύρια λειτουργικά συστήματα, όπως Linux, UNIX, Windows και Mac OS X. Διαθέτει όλα τα χαρακτηριστικά της standard SQL, καθώς και επιπλέον χαρακτηριστικά, όπως σύνθετα queries, foreign keys, triggers, views και άλλα. Επίσης, η PostgreSQL μπορεί να επεκταθεί από το χρήστη, ο οποίος μπορεί να δημιουργήσει καινούριους τύπους δεδομένων, συναρτήσεις, μεθόδους, διαδικαστικές γλώσσες και άλλα.

Για την επιλογή του συστήματος διαχείρισης βάσεων δεδομένων που θα χρησιμοποιούσα έκανα μια μικρή έρευνα για συστήματα διαχείρισης βάσεων δεδομένων τα οποία διατίθενται δωρεάν. Η έρευνα αυτή με οδήγησε στο συμπέρασμα ότι τα 2 πιο δημοφιλή συστήματα, τα οποία θεωρούνται από τα καλύτερα, είναι η PostgreSQL και η MySQL. Η επιλογή της PostgreSQL μεταξύ των δύο ήταν εύκολη, γιατί φαίνεται ότι η MySQL δεν αποδίδει σωστά κατά τη χρήση των TPC-H Benchmarks λόγω του ότι δεν έχει αρκετά καλές δυνατότητες για εκτέλεση σύνθετων queries [2],[3]. Επίσης, υπάρχει κάποια δυσκολία στη χρήση των queries των TPC-H

Benchmarks με την MySQL, γιατί χρειάζονται μετατροπές στον κώδικα πριν να μπορέσουν να χρησιμοποιηθούν[4] και μπορεί να δημιουργηθούν διάφορα προβλήματα κατά την εισαγωγή των αρχείων των TPC-H Benchmarks στη βάση[5].

Στην εργασία μου χρησιμοποιώ την έκδοση 8.4 της PostgreSQL για τη δημιουργία της βάσης και την εκτέλεση των queries που περιέχονται στα TPC-H Benchmarks, για τα οποία αναφέρομαι στο επόμενο υποκεφάλαιο.

2.2 TPC-H Benchmarks

Στην επιστήμη της πληροφορικής το Benchmark είναι η εκτέλεση ενός προγράμματος ή ενός συνόλου από προγράμματα ή άλλες λειτουργίες με τη χρήση κάποιων τυποποιημένων δοκιμών. Είναι σχεδιασμένα ώστε να μιμούνται ένα συγκεκριμένο τύπο φόρτου εργασίας σε ένα σύστημα και σκοπός τους είναι η αξιολόγηση της σχετικής επίδοσης μιας μηχανής. Από τα αποτελέσματα που δίνει η εκτέλεση των Benchmarks μπορούμε να συγκρίνουμε την επίδοση διάφορων και διαφορετικών συστημάτων υπολογιστών και μέσω των μετρήσεων παίρνουμε ενδείξεις για το πώς μπορούμε να βελτιώσουμε την επίδοση της μηχανής.

Η διαδικασία εκτέλεσης των Benchmarks ονομάζεται Benchmarking και συνήθως χρησιμοποιείται για την μελέτη των χαρακτηριστικών που αφορούν την επίδοση του υλικού ενός υπολογιστή, όπως π.χ. την επίδοση της CPU, αλλά υπάρχουν και περιπτώσεις κατά τις οποίες χρησιμοποιείται για να αξιολογεί επιδόσεις που αφορούν λογισμικό. Τα Software benchmarks, παραδείγματος χάριν, χρησιμοποιούνται για μεταγλωττιστές ή συστήματα διαχείρισης βάσεων δεδομένων.

Από τον μεγάλο αριθμό διαθέσιμων benchmarks επέλεξα το TPC-H Benchmark σε έκδοση 2.14.0, που δημοσιεύτηκε από το Transaction Processing Performance Council (TPC) για τους σκοπούς της εργασίας μου, γιατί χρησιμοποιείται ευρέως από τις επιχειρήσεις για ανάλυση της επίδοσης και παρέχει εκτός από τα queries, και τα tables και relations για δημιουργία της βάσης. Στην ουσία παρέχει ότι χρειαζόμαστε για τη δημιουργία της βάσης και επίσης δίνει τη δυνατότητα για δημιουργία βάσεων διαφόρων μεγεθών.

Το TPC-H Benchmark[6] είναι ένα decision support benchmark, δηλαδή προσομοιώνει συστήματα για λήψη αποφάσεων, τα οποία μπορούν να επεξεργαστούν μεγάλο όγκο δεδομένων, εκτελούν queries με μεγάλο βαθμό πολυπλοκότητας και δίνουν απαντήσεις σε κρίσιμα ερωτήματα που απασχολούν τις επιχειρήσεις για τα συστήματά βάσεων δεδομένων τους. Το Benchmark περιλαμβάνει ένα σύνολο από ad-hoc, δηλαδή εξειδικευμένα, queries των οποίων η εξειδίκευση στοχεύει στην κάλυψη των απαιτήσεων των επιχειρήσεων σχετικά με το πώς θέλουν τα συστήματά τους. Επίσης, περιλαμβάνει ένα σύνολο δεδομένων, τα οποία έχουν επιλεχθεί έτσι ώστε να καλύπτουν ένα εύρος επιχειρηματικών απαιτήσεων και άρα να δίνουν πιο αντιπροσωπευτικά αποτελέσματα. Το TPC-H Benchmark χρησιμοποιείται για να δώσει πληροφορίες για το πόσο ικανό είναι ένα σύστημα στην επεξεργασία queries. Κατά συνέπεια, δίνει πληροφορίες για το αν μπορεί το σύστημα αυτό να εξυπηρετήσει σωστά μια επιχείρηση, η οποία έχει ανάγκη από συστήματα που θα μπορούν να επεξεργαστούν αποτελεσματικά μεγάλο όγκο δεδομένων.

Όπως προανέφερα, το TPC-H Benchmark δίνει τη δυνατότητα για δημιουργία βάσεων διαφόρων μεγεθών, με τη χρήση του Scale Factor (SF). Για την εργασία μου χρησιμοποίησα SF=1 για τη δημιουργία μιας βάσης 1GB. Ο Πίνακας 2.1 δείχνει τα tables που προσφέρει το TPC-H Benchmark, δίνει μία σύντομη περιγραφή του κάθε table και δείχνει το μέγεθος των tables σε σχέση με το SF.

REGION	Περιέχει 5 περιοχές του κόσμου	5
NATION	Περιέχει 25 χώρες του κόσμου, συσχετίζοντάς τα με την περιοχή (REGION) στην οποία ανήκουν	25
SUPPLIER	Περιέχει προμηθευτές, συσχετίζοντάς τους με τη χώρα (NATION) στην οποία βρίσκονται	SF*10,000
PART	Περιέχει εξαρτήματα (parts) διάφορων τύπων	SF*200,000
CUSTOMER	Περιέχει πελάτες, συσχετίζοντάς τους με την χώρα τους και από ποιες αγορές προμηθεύονται εξαρτήματα	SF*150,000
PARTSUPP	Περιέχει πληροφορίες για το ποιος προμηθευτής προμηθεύει ποια εξαρτήματα, μαζί με την ποσότητα και	SF*800,000

	την τιμή ανά εξάρτημα	
ORDERS	Περιέχει παραγγελίες συσχετιζόμενες με τον πελάτη που τις έκανε	SF*1,500,000
LINEITEM	Περιέχει πληροφορίες για κάθε αντικείμενο που ανήκει σε μια παραγγελία	SF*6,000,000

Πίνακας 2.1 Tables του TPC-H Benchmark

Το TPC-H Benchmark περιέχει 22 queries, από τα οποία χρησιμοποιώ τα 19 πρώτα, δηλαδή Q1-Q19. Ο λόγος που δεν χρησιμοποιώ τα 3 τελευταία queries, είναι επειδή η εκτέλεσή τους δεν ήταν επιτυχής και συγκεκριμένα τα queries αυτά δεν τερμάτιζαν. Τα 19 queries που χρησιμοποιώ εκτελούν διαφορετικές λειτουργίες και δίνουν διαφορετικά αποτελέσματα. Πιο κάτω δίνονται αναλυτικά οι ορισμοί των queries[6].

1. Q1: Το query αυτό ονομάζεται Pricing Summary Report Query και δίνει μία αναφορά με τιμολόγιση για όλα τα lineitems που έχουν τιμολογηθεί και αποσταλεί μετά από μια δοσμένη ημερομηνία. Η αναφορά περιέχει τα ολικά ποσά για την εκτεταμένη τιμή (extended price), την εκτεταμένη τιμή με έκπτωση (discounted extended price), την εκτεταμένη τιμή με έκπτωση συν φόρο (discounted extended price plus tax), μέσο όρο ποσότητας (average quantity), μέσο όρο εκτεταμένης τιμής (average extended price) και μέσο όρο έκπτωσης (average discount). Οι τιμές ομαδοποιούνται με βάση τις τιμές RETURNFLAG και LINESTATUS και ταξινομούνται σε αύξουσα σειρά σύμφωνα με τις ίδιες τιμές. Επίσης, στην αναφορά περιλαμβάνεται και ο αριθμός των lineitems που βρίσκονται σε κάθε ομάδα.
2. Q2: Το query αυτό ονομάζεται Minimum Cost Supplier Query και βρίσκει τον προμηθευτή (supplier) ο οποίος προμηθεύει στη μικρότερη τιμή ένα συγκεκριμένο εξάρτημα (part) με συγκεκριμένο τύπο και μέγεθος, σε μια δεδομένη περιοχή (region). Αν βρεθούν περισσότεροι από ένας προμηθευτές, επιστρέφονται τα στοιχεία για τους 100 προμηθευτές με τα υψηλότερα υπόλοιπα λογαριασμών (highest account balances). Τα στοιχεία που επιστρέφει το query είναι για κάθε προμηθευτή το υπόλοιπο λογαριασμού του (account balance), το όνομα (name) και τη χώρα (nation) του, τον αριθμό (number) και κατασκευαστή

(manufacturer) του είδους και τη διεύθυνση (address), τον αριθμό τηλεφώνου (phone number) και τα σχόλια (comment information) του προμηθευτή.

3. Q3: Το query αυτό ονομάζεται Shipping Priority Query και επιστρέφει τις 10 μη αποσταλμένες παραγγελίες με τη μεγαλύτερη αξία. Τα στοιχεία που επιστρέφει είναι η προτεραιότητα αποστολής (shipping priority) και τα πιθανά έσοδα (potential revenue) των 10 παραγγελιών που δεν έχουν αποσταλεί από τη δεδομένη ημερομηνία και έχουν τα μεγαλύτερα πιθανά έσοδα. Οι παραγγελίες ταξινομούνται με φθίνουσα σειρά ανάλογα με τα πιθανά έσοδα τους.
4. Q4: Το query αυτό ονομάζεται Order Priority Checking Query και προσδιορίζει πόσο καλά δουλεύει το σύστημα προτεραιότητας παραγγελιών και δίνει μια εκτίμηση της ικανοποίησης των πελατών. Το query επιστρέφει τον αριθμό των παραγγελιών οι οποίες παραγγέλθηκαν μέσα σε ένα δεδομένο τρίμηνο και αποστάληκαν στον πελάτη καθυστερημένα, δηλαδή μετά την καθορισμένη ημερομηνία. Οι παραγγελίες ομαδοποιούνται και ταξινομούνται σε αύξουσα σειρά σύμφωνα με την προτεραιότητα παραγγελίας τους (priority order).
5. Q5: Το query αυτό ονομάζεται Local Supplier Volume Query και απαριθμεί τον όγκο εσόδων από τους τοπικούς προμηθευτές, δηλαδή για κάθε χώρα βρίσκει τον όγκο που προέκυψε μόνο από παραγγελίες που οι πελάτες και προμηθευτές τους βρίσκονται στη συγκεκριμένη χώρα. Λαμβάνονται υπόψιν μόνο οι παραγγελίες που έγιναν στο δεδομένο έτος και εμφανίζονται η χώρα (nation) και ο όγκος εσόδων (revenue volume) κατά φθίνουσα σειρά εσόδων.
6. Q6: Το query αυτό ονομάζεται Forecasting Revenue Change Query και ποσοτικοποιεί το ποσό αύξησης των εσόδων που θα προέκυπτε από την κατάργηση εκπτώσεων συγκεκριμένων εταιριών, σε ένα δεδομένο εύρος ποσοστού, σε ένα δεδομένο έτος.
7. Q7: Το query αυτό ονομάζεται Volume Shipping Query και με δεδομένες δύο διαφορετικές χώρες (nation) βρίσκει τα ακαθάριστα έσοδα με έκπτωση που προέρχονται από lineitems που αποστάληκαν από προμηθευτή οποιασδήποτε από τις δύο χώρες σε πελατή της άλλης χώρας, μεταξύ του 1995 και 1996. Επιστρέφονται η χώρα του προμηθευτή (supplier nation), η χώρα του πελάτη (customer nation), το έτος (year) και τα έσοδα (revenue) από αποστολές που έγιναν το συγκεκριμένο έτος. Η ταξινόμηση γίνεται με αύξουσα σειρά σύμφωνα με τη χώρα προμηθευτή, τη χώρα πελάτη και το έτος.

8. Q8: Το query αυτό ονομάζεται National Market Share Query και υπολογίζει πώς άλλαξε το μερίδιο αγοράς (Market Share) σε μια δεδομένη χώρα μέσα στα τελευταία 2 χρόνια για δεδομένο τύπο εξαρτήματος (part type).
9. Q9: Το query αυτό ονομάζεται Product Type Profit Measure Query και υπολογίζει το κέρδος που έδωσε η δεδομένη σειρά εξαρτημάτων, ανά χώρα προμηθευτή και έτος. Για κάθε χώρα (nation) και έτος (year) υπολογίζεται το κέρδος για όλα τα εξαρτήματα που παραγγέλθηκαν το συγκεκριμένο έτος, πληρώθηκαν από προμηθευτή της δεδομένης χώρας και περιέχουν ένα δεδομένο substring στο όνομά τους. Η λίστα περιέχει τις χώρες σε αύξουσα αλφαβητική σειρά και για κάθε χώρα περιέχει το έτος και το κέρδος σε φθίνουσα σειρά σύμφωνα με το έτος.
10. Q10: Το query αυτό ονομάζεται Returned Item Reporting Query και βρίσκει τους πρώτους 20 πελάτες που έχουν επιστρέψει εξαρτήματα και είχαν τη μεγαλύτερη επίδραση στην απώλεια εσόδων για ένα δεδομένο τρίμηνο. Επιστρέφεται μια λίστα ταξινομημένη βάσει απώλειας εσόδων, η οποία περιλαμβάνει το όνομα (name) των πελατών, τη διεύθυνσή (address) τους, τη χώρα (nation) τους, τον αριθμό τηλεφώνου (phone number) τους, το υπόλοιπο λογαριασμού (account balance) τους, τα σχόλια (comment information) και την απώλεια εσόδων.
11. Q11: Το query αυτό ονομάζεται Important Stock Identification Query και βρίσκει όλα τα εξαρτήματα που δίνουν ένα σημαντικό ποσοστό της συνολικής αξίας όλων των εξαρτημάτων που είναι διαθέσιμα στο στοκ των προμηθευτών μιας δεδομένης χώρας. Παρουσιάζονται ο αριθμός εξαρτήματος (part number) και η τιμή (value) του, σε φθίνουσα σειρά βάσει της τιμής τους.
12. Q12: Το query αυτό ονομάζεται Shipping Modes and Order Priority Query. Το query παίρνει όλα τα lineitems που παραλήφθηκαν από πελάτες στη δεδομένη χρονιά και μετράει, με βάσει δύο διαφορετικά δεδομένα ship modes, τον αριθμό των lineitems που ανήκουν σε παραγγελίες για τις οποίες το l_receiptdate ξεπερνά το l_commitdate. Λαμβάνονται υπόψιν μόνο τα lineitems τα οποία αποστάληκαν πριν την l_commitdate και τα επιλεγμένα lineitems χωρίζονται σε τρεις ομάδες, αυτά με προτεραιότητα ΥΨΗΛΗ (HIGH) ή ΕΠΕΙΓΟΥΣΑ (URGENT) και αυτά με άλλη προτεραιότητα.

13. Q13: Το query αυτό ονομάζεται Customer Distribution Query και κάνει κατανομή όλων των πελατών ανάλογα με το πόσες παραγγελίες έχουν κάνει, συμπεριλαμβανομένων και των πελατών που δεν έχουν καταγεγραμμένες παραγγελίες. Επιστρέφεται ο αριθμός των πελατών που έχουν κάνει 0, 1, 2, 3 κτλ παραγγελίε και σημειώνονται οι παραγγελίες που δεν εμπίπτουν σε ειδική κατηγορία παραγγελιών.
14. Q14: Το query αυτό ονομάζεται Promotion Effect Query και υπολογίζει το ποσοστό εσόδων σε ένα δεδομένο μήνα μιας δεδομένης χρονιάς, που προέρχονται από διαφημισμένα εξαρτήματα. Παρουσιάζεται το ποσοστό εσόδων για τα εξαρτήματα που αποστάληκαν το συγκεκριμένο μήνα.
15. Q15: Το query αυτό ονομάζεται Top Supplier Query και βρίσκει τον προμηθευτή που συνέλαβε περισσότερο στα συνολικά έσοδα από εξαρτήματα που αποστάληκαν κατά τη διάρκεια ενός δεδομένου τριμήνου ενός δεδομένου έτους. Σε περίπτωση ισοπαλίας, παρουσιάζονται όλοι οι προμηθευτές των οποίων η συμβολή ήταν ίση με τη μέγιστη, ταξινομημένοι σε αύξουσα σειρά βάσει του αριθμού (supplier number) τους.
16. Q16: Το query αυτό ονομάζεται Parts/Supplier Relationship Query και μετρά τον αριθμό των προμηθευτών που μπορούν να προμηθεύσουν εξαρτήματα τα οποία καλύπτουν τις απαιτήσεις κάποιου πελάτη. Τα αποτελέσματα παρουσιάζονται με φθίνουσα σειρά μετρητή και αύξουσα σειρά μάρκας (brand), τύπου (type) και μεγέθους (size).
17. Q17: Το query αυτό ονομάζεται Small-Quantity-Order Revenue Query. Το query υπολογίζει το μέσο όρο της απώλειας εσόδων αν δεν γίνονταν πλέον παραγγελίες για μικρές ποσότητες συγκεκριμένων εξαρτημάτων.Δίνεται ο τύπος κιβωτίου (container type) και η μάρκα (brand) και, εξετάζοντας όλες τις παραγγελίες που υπάρχουν μέσα στη βάση, υπολογίζεται ο μέσος όρος ποσότητας εξαρτημάτων της συγκεκριμένης μάρκας που έχουν ζητηθεί. Έπειτα υπολογίζεται η μέση ετήσια απώλεια ακαθάριστων εσόδων αν δεν γίνονταν οι παραγγελίες για εξαρτήματα με ποσότητα μικρότερη του 20% του μέσου όρου.
18. Q18: Το query αυτό ονομάζεται Large Volume Customer Query και επιστρέφει μια λίστα με τους πρώτους 100 πελάτες που έχουν πραγματοποιήσει παραγγελίες μεγάλων ποσοτήτων. Η λίστα περιλαμβάνει το όνομα του πελάτη (customer name), το κλειδί του πελάτη (customer key), το κλειδί της

παραγγελίας (order key), την ημερομηνία, την συνολική τιμή και την ποσότητα εξαρτημάτων στην παραγγελία.

19. Q19: Το query αυτό ονομάζεται Discount Revenued Query και υπολογίζει τα ακαθάριστα έσοδα από όλες τις παραγγελίες για τρεις διαφορετικούς τύπους εξαρτημάτων τα οποία αποστάληκαν αεροπορικώς και παραδόθηκαν αυτοπροσώπως. Τα εξαρτήματα επιλέγονται βάσει ενός συνδυασμού συγκεκριμένων μαρκών (brands), μιας λίστας από κιβώτια (container) και ενός εύρους τιμών.

Για να εκτελεστούν σωστά τα queries και να μπορέσουν να επιστρέψουν αποτελέσματα, κάποια δεδομένα πρέπει να πάρουν προκαθορισμένες τιμές. Οι τιμές αυτές κανονικά επιλέγονται τυχαία. Στην περίπτωση της εργασίας μου όμως, όπου θέλω να αναλύσω μετρήσεις που πήρα κατά τη διάρκεια της εκτέλεσης των queries, δεν μπορούσα να έχω τυχαίες τιμές για εκτελέσεις του ίδιου query, γιατί διαφορετικές τιμές συνεπάγονται διαφορετικό τρόπο εκτέλεσης κάθε φορά, διαφορετικό output και μη αντιπροσωπευτικά αποτελέσματα από τις μετρήσεις. Γι' αυτό το λόγο, το TPC-H Benchmark δίνει συγκεκριμένες προκαθορισμένες τιμές στα δεδομένα αυτά, ώστε κάθε εκτέλεση του query να δίνει πάντα το ίδιο output και να κάνει τα ίδια βήματα για να δώσει αυτό το output. Επίσης κατά τον ορισμό των queries στο documentation του TPC-H Benchmark[6] παρουσιάζεται το output αυτό, ώστε να μπορούμε να ελέγξουμε κατά πόσο το output που παίρνουμε είναι το σωστό. Έτσι, μπορούμε να είμαστε σίγουροι ότι τα αποτελέσματα που παίρνουμε από τις μετρήσεις για την εκτέλεση των queries είναι αξιόπιστα.

Το TPC-H Benchmark χρησιμοποιήθηκε για το πειραματικό στάδιο της εργασίας μου. Με τα αποτελέσματα από την εκτέλεση των queries του Benchmark και τη χρήση του perf, στο οποίο θα αναφερθώ στο επόμενο υποκεφάλαιο, μπόρεσα να πάρω διάφορες μετρήσεις σχετικά με τα χαρακτηριστικά που με ενδιέφεραν. Με τις μετρήσεις αυτές μπόρεσα να βγάλω κάποια συμπεράσματα για το πώς τα χαρακτηριστικά των μηχανών επηρεάζουν την εκτέλεση.

2.3 Linux perf

Πριν αναφερθώ στο perf θα μιλήσω για τους Hardware Performance Counters, των οποίων τις μετρήσεις αναλύει το perf.

Οι Hardware Performance Counters είναι ένα σύνολο από ειδικούς καταχωρητές, οι οποίοι έχουν σαν σκοπό την καταχώρηση μετρήσεων σε λειτουργίες που αφορούν το υλικό και έτσι βοηθούν τους χρήστες να διεξάγουν χαμηλού επιπέδου ανάλυση επίδοσης. Προσφέρουν πρόσβαση σε λεπτομερείς πληροφορίες για την επίδοση των λειτουργικών μονάδων της CPU, δηλαδή τις μνήμες cache και την κύρια μνήμη. Οι counters αποτελούν τη βάση για το profiling διάφορων εφαρμογών και για εντοπισμό των hotspots κατά την εκτέλεση της εφαρμογής, δηλαδή τα σημεία όπου υπάρχουν τα μεγαλύτερα ποσοστά εκτελέσιμων εντολών και σημεία της εφαρμογής τα οποία κάνουν το μεγαλύτερο χρόνο εκτέλεσης, άρα καθυστερούν την εκτέλεση.

Οι Hardware Performance counters που χρησιμοποιώ είναι καταχωρητές υλικού (hardware registers) της CPU και τους αναλύω μέσω του perf[7],[8], το οποίο είναι kernel-based υποσύστημα του λειτουργικού συστήματος Linux και βρίσκεται στο πακέτο linux-tools. Το perf παρέχει ένα πλαίσιο για συλλογή δεδομένων που αφορούν την απόδοση κατά την εκτέλεση και ανάλυσή αυτών των δεδομένων. Το perf χρησιμοποιείται για μέτρηση γεγονότων υλικού (hardware events), τα οποία μπορεί να ποικίλουν ανάλογα με το υλικό και λογισμικό του συστήματος.

Οι εντολές[9] που διαθέτει το perf για ανάλυση είναι οι εξης:

1. perf stat. Η εντολή αυτή δίνει γενικά στατιστικά στοιχεία για κοινά events απόδοσης, όπως το πόσες εντολές εκτελέστηκαν και πόσοι clock cycles χρειάστηκαν για την εκτέλεση. Ο χρήστης μπορεί να επιλέξει να πάρει μετρήσεις μόνο για συγκεκριμένα events, αντί για όλη τη λίστα των προκαθορισμένων events.
2. perf record. Η εντολή αυτή συλλέγει τα δεδομένα σχετικά με την απόδοση ανα thread, process ή cpu και τα καταγράφει στο δυαδικό αρχείο perf.data, το οποίο μπορούμε να αναλύσουμε χρησιμοποιώντας την εντολή perf report.

3. perf report. Η εντολή αυτή διαβάζει τα δεδομένα από το αρχείο perf.data, τα αναλύει και δημιουργεί ένα συνοπτικό προφιλ εκτέλεσης.
4. perf annotate. Η εντολή αυτή δίνει πληροφορίες για τις βασικές εντολές, όπως push, add, cmp κτλ. Για να πάρουμε τις πληροφορίες εκτελούμε την perf annotate για μια συγκεκριμένη εντολή και παρουσιάζεται το ποσοστό χρήσης των βασικών εντολών για κάθε συνάρτηση που καλείται με την εκτέλεση της εντολής. Επίσης, αν η εφαρμογή μεταγλωττιστεί με –ggdb, η perf annotate παράγει πληροφορίες για τον πηγαίο κώδικα.
5. perf list. Η εντολή αυτή παρουσιάζει μια λίστα με όλα τα events που είναι διαθέσιμα στην συγκεκριμένη μηχανή. Τα events αυτά μπορεί να διαφέρουν, ανάλογα με το υλικό που εποπτεύει την απόδοση και το configuration του λογισμικού του συστήματος.
6. perf top. Η εντολή αυτή παρουσιάζει πληροφορίες, κατά τη διάρκεια της εκτέλεσης, δηλαδή σε real time, για τις συναρτήσεις που χρησιμοποιούνται. Συγκεκριμένα, το προεπιλεγμένο event που εξετάζει η perf top είναι οι κύκλοι που χρειάζονται για την εκτέλεση της συνάρτησης και τυπώνονται οι εντολές και τα δείγματα ταξινομημένα σε φθίνουσα σειρά βάσει των κύκλων και άρα τυπώνονται πρώτες οι εντολές οι οποίες χρειάστηκαν τον περισσότερο χρόνο να εκτελεστούν.

Η εντολή που χρησιμοποίησα για την εργασία μου είναι η perf stat, με την οποία συνέλεξα μετρήσεις για τα Cache references, τα Cache misses και το Χρόνο εκτέλεσης. Ο τρόπος που εκτελείται η συγκεκριμένη εντολή φαίνεται πιο κάτω[10]:

```
perf stat [<options>] [<command>]
```

όπου `<options>` μπορεί να είναι κάποια από τις επιλογές που προσφέρει η εντολή perf stat και `<command>` είναι η εντολή για την οποία θέλουμε να συλλέξουμε μετρήσεις.

Η perf stat, αν εκτελεστεί χωρίς καθορισμένα events, επιστρέφει στατιστικά στοιχεία για τα προκαθορισμένα events, όπως φαίνεται πιο κάτω[10]:

```
perf stat -B dd if=/dev/zero of=/dev/null count=1000000

 Performance counter stats for 'dd if=/dev/zero of=/dev/null
count=1000000':

            5,099 cache-misses             #      0.005 M/sec (scaled
from 66.58%)
          235,384 cache-references         #      0.246 M/sec (scaled
from 66.56%)
        9,281,660 branch-misses            #      3.858 %     (scaled
from 33.50%)
      240,609,766 branches                 #    251.559 M/sec (scaled
from 33.66%)
    1,403,561,257 instructions             #      0.679 IPC   (scaled
from 50.23%)
    2,066,201,729 cycles                   #   2160.227 M/sec (scaled
from 66.67%)
              217 page-faults              #      0.000 M/sec
                3 CPU-migrations           #      0.000 M/sec
               83 context-switches         #      0.000 M/sec
       956.474238 task-clock-msecs         #      0.999 CPUs

       0.957617512  seconds time elapsed
```

Για να πάρω μετρήσεις μόνο για τα events που ήθελα χρησιμοποίησα την επιλογή –e ακολουθούμενη από τα events που με ενδιέφεραν[10], όπως φαίνεται πιο κάτω:

```
perf stat –e cache-misses,cache-references,cycles dd if=/dev/zero
of=/dev/null count=1000000

Performance counter stats for 'dd if=/dev/zero of=/dev/null
count=1000000':

            5,099 cache-misses             #      0.005 M/sec (scaled
from 66.58%)
          235,384 cache-references         #      0.246 M/sec (scaled
from 66.56%)
    2,066,201,729 cycles                   #   2160.227 M/sec (scaled
from 66.67%)

        0.957617512  seconds time elapsed
```

Ο λόγος που επέλεξα να χρησιμοποιήσω το perf είναι ότι δε χρειάζονται μετατροπές στον κώδικα για να χρησιμοποιηθεί και άρα διευκολύνθηκα πολύ στη χρήση του. Ένα

μειονέκτημα των μετρήσεων με τη χρήση Performance counters είναι οτι ότι υπάρχει διαφορά στο είδος των μετρήσεων που παίρνουμε, γιατί για διαφορετικές αρχιτεκτονικές μπορεί να υπάρχουν διαφορετικές μετρήσεις.

Το perf επιλέχθηκε γιατί ήθελα να πάρω μετρήσεις κατά τη διάρκεια της εκτέλεσης των queries και συγκεκριμένα μετρήσεις για τα Cache references, τα Cache misses και το Χρόνο εκτέλεσης, για τα οποία θα αναφερθώ εκτενώς στο 4^{ο} κεφάλαιο.

Κεφάλαιο 3

Μηχανές και χαρακτηριστικά

3.1 Μηχανές

Οι μηχανές που χρησιμοποιώ για την εργασία μου είναι

- η μηχανή csteras, με επεξεργαστή Intel Xeon E5320
- η μηχανή cs6472, με επεξεργαστή AMD Opteron 2427 και
- η μηχανή csatom, με επεξεργαστή Intel Atom N270.

Στον πίνακα **Πίνακας 3.1** φαίνονται κάποια από τα χαρακτηριστικά των τριών μηχανών, τα οποία χρησιμοποίησα για να εξάγω τα συμπεράσματά μου[11],[12],[13].

Χαρακτηριστικά	Μηχανή csteras	Μηχανή cs6472	Μηχανή csatom
Συχνότητα CPU	1.86 GHz	2.2 GHz	1.60 GHz
Πυρήνες	8 (2x4)	12 (2x6)	1
L1 Data cache	4x32 KB, 8-way set associative	6 x 64 KB, 2-way set associative	24 KB
L1 Instruction	4x32 KB, 8-way	6 x 64 KB 2-way	32 KB

cache	set associative	associative	
L2 Cache	2x4 MB shared (per chip), 16-way set associative	6x512 KB shared (per chip), 16-way set associative	512 KB, 8-way set associative
L3 Cache	Δεν υπάρχει	6 MB shared, 48-way set associative	Δεν υπάρχει
Μνήμη RAM	17.7 GB	31.5 GB	1 GB
Μέγιστη TDP	80W	75 W	2.5W
Release Price	$690	$455	$44
Τύπος μηχανής	Server	Server	Laptop

Πίνακας 3.1 Χαρακτηριστικά μηχανών

Στο επόμενο υποκεφάλαιο γίνεται πιο αναλυτική εξήγηση αυτών των χαρακτηριστικών.

3.2 Χαρακτηριστικά μηχανών

3.2.1 Συχνότητα Επεξεργαστή (CPU frequency):

Η συχνότητα επεξεργαστή[14] είναι η συχνότητα με την οποία λειτουργεί η CPU. Όσο πιο μεγάλη είναι η συχνότητα ενός επεξεργαστή, τόσο πιο γρήγορη είναι η λειτουργία του. Οι σύγχρονοι επεξεργαστές δεν λειτουργούν πάντα με την ίδια συχνότητα, αλλά μπορεί να την μειώνουν ανά διαστήματα για εξοικονόμηση ενέργειας. Η συχνότητα επεξεργαστή μετριέται σε Hertz.

3.2.2 Πυρήνες (Cores):

Οι επεξεργαστές είχαν σχεδιαστεί αρχικά με μόνο ένα πυρήνα. Σήμερα όμως έχουμε επεξεργαστές με περισσότερους από έναν πυρήνες, τους multi-core processors, οι οποίοι έχουν 2 ή περισσότερες ανεξάρτητες κεντρικές μονάδες επεξεργασίας, που

ονομάζονται cores, οι οποίοι διαβάζουν και εκτελούν εντολές. Οι πολλαπλοί πυρήνες μπορούν να εκτελούν εντολές την ίδια ώρα, όμως η αύξηση της απόδοσης του υπολογιστή από αυτό το γεγονός εξαρτάται σε μεγάλο βαθμό από τους αλγορίθμους που χρησιμοποιούνται και από το βαθμό στον οποίο μπορεί να παραλληλιστεί το πρόγραμμα που εκτελείται. Για πλήρη εκμετάλλευση των πολυπύρηνων επεξεργαστών θα πρέπει να παρέχονται δεδομένα σε όλους, ώστε να μπορούν να δουλέυουν και να μπορεί να υπάρχει συνεργασία μεταξύ τους όταν δουλεύουν όλοι σε μία μόνο εργασία. Επίσης, η αυξημένη κατανάλωση ενέργειας και η μεγαλύτερη δημιουργία θερμότητας είναι θέματα που εγείρονται με τους επιπλέον πυρήνες. Γι'αυτό το λόγο οι πολυπύρηνοι επεξεργαστές και ο παραλληλισμός είναι θέματα που διερευνούνται πολύ τη σύγχρονη εποχή.

3.2.3 Cache

Μέχρι τις αρχές της δεκαετίας του 1990, οι επεξεργαστές και οι μνήμες των υπολογιστών, δούλευαν με σχετικά κοντινές ταχύτητες. Από το 1990 όμως, ενώ υπήρχε σημαντική και ραγδαία αύξηση στην ταχύτητα των επεξεργαστών, δεν υπήρχε αυτή η αύξηση και για τις μνήμες. Έτσι δημιουργήθηκε το «Processor – Memory Performance Gap», δηλαδή το μεγάλο χάσμα μεταξύ της συχνότητας λειτουργίας των επεξεργαστών και της μνήμης. Λόγω αυτού του χάσματος, οι επεξεργαστές έπρεπε να περιμένουν τις αργές μνήμες να μεταφέρουν τα δεδομένα, ώστε να μπορούν να εκτελέσουν τους υπολογισμούς τους και άρα η ταχύτητα των επεξεργαστών δεν αξιοποιούνταν. Για να λυθεί αυτό το πρόβλημα χρησιμοποιήθηκε η cache και έγινε αλλαγή στην ιεραρχία μνήμης των υπολογιστών.

Τί είναι Cache

Η cache[15] είναι μνήμη από υψηλής ταχύτητας στατική RAM (SRAM), αντί της πιο αργής και πιο φτηνής δυναμικής RAM (DRAM), η οποία χρησιμοποιείται για την κύρια μνήμη. Η cache χρησιμεύει στην CPU για να μειωθεί ο μέσος χρόνος πρόσβασης στη μνήμη, χρησιμοποιώντας το caching μνήμης. Με το caching μνήμης τα δεδομένα από τοποθεσίες της κύριας μνήμης, στις οποίες υπάρχει πιο συχνή πρόσβαση, αποθηκεύονται στην cache, ώστε να ανακτούνται από την cache, αντί από την αργή

δυναμική μνήμη. Άρα, ενόσο γίνονται αναφορές σε τοποθεσίες μνήμης που έχουν αποθηκευτεί στην cache, ο χρόνος καθυστέρησης για εξυπηρέτηση αυτών των αναφορών εξαρτάται από το χρόνο καθυστέρησης της cache αντί της κύριας μνήμης, και άρα είναι σημαντικά μικρότερος.

Λειτουργία και Οργάνωση της Cache

Η δομική μονάδα μιας μνήμης Cache, είναι η Cache Line (Γραμμή). Όταν ο επεξεργαστής ζητήσει να διαβάσει μια Word (η ελάχιστη μονάδα δεδομένων που διαβάζεται από τον επεξεργαστή) τότε το Block της κύριας μνήμης που περιέχει τη Word μεταφέρεται ολόκληρο σε μια Cache Line. Κάθε Cache Line έχει τη δομή που φαίνεται στο Σχήμα 3.2

Σχήμα 3.2 Δομή Cache Line

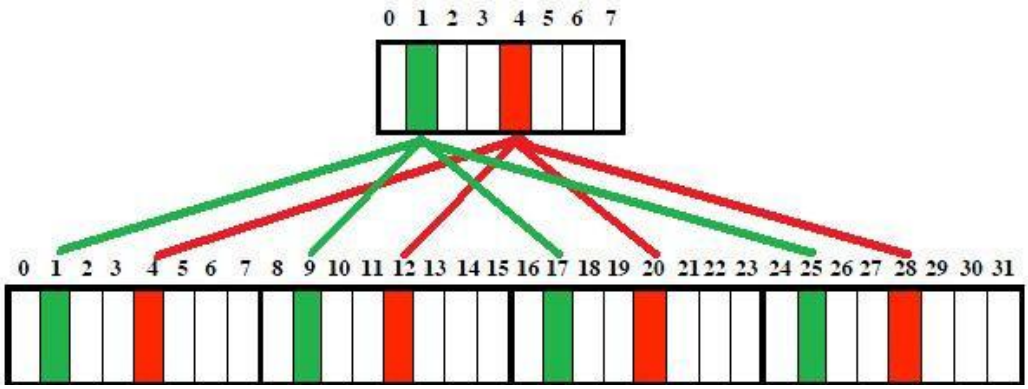
Το data block περιέχει τα δεδομένα που παίρνουμε από την κύρια μνήμη.
Το tag περιέχει μέρος της διεύθυνσης των δεδομένων που παίρνουμε από την κύρια μνήμη. Είναι το αναγνωριστικό για το ποιο Block καταλαμβάνει τον χώρο δεδομένων της Γραμμής.
Τα flag bits προστίθενται για βελτιστοποίηση του ελέγχου της Cache και αφορούν την κατάσταση των δεδομένων που περιέχονται στη Γραμμή. Τα συνήθη flag bits είναι τα εξής:

- Dirty Bit : Δείχνει αν η Γραμμή έχει έγκυρα δεδομένα.
- Shared Bit : Δείχνει αν υπάρχουν αντίγραφα αυτής της Γραμμής και σε άλλες Caches.
- Valid Bit : Δείχνει αν τα δεδομένα της Γραμμής έχουν τροποποιηθεί, από την στιγμή που μεταφέρθηκαν στην Cache.

Memory Mapping

Για να αποφασιστεί πού μπορεί ένα Block δεδομένων να τοποθετηθεί στην Cache (Block Placement) χρησιμοποιείται το Memory Mapping, το οποίο προσδιορίζει το

Associativity της Cache. Το Memory Mapping προσδιορίζει τη λογική με την οποία θα τοποθετηθεί ένα Block σε μία Cache Line, μέσω τη χρήση των Mapping Functions. Οι τρεις βασικές Mapping Functions φαίνονται και εξηγουνται πιο κάτω:



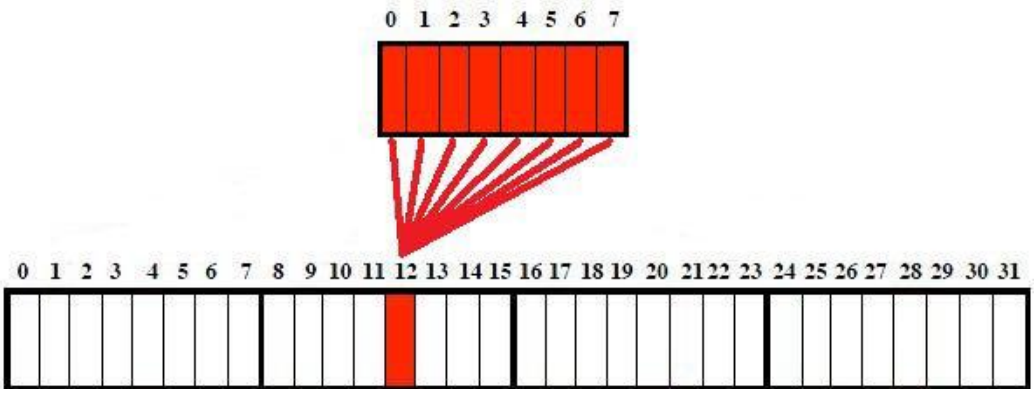
Σχήμα 3.3 Direct Mapping

Αν κάθε Block αντιστοιχεί μόνο σε μία θέση, τότε η Cache είναι direct mapped και η θέση του Block υπολογίζεται ως εξής

(Block address) mod (Αριθμός Blocks στην Cache)

Η Direct Mapped Cache είναι ο απλούστερος τύπος Cache, γιατί κατασκευάζεται εύκολα και παρέχει τον πιο εύκολο τρόπο για έλεγχο του Hit, δηλαδή του κατά πόσο βρέθηκε επιτυχώς το Block που αναζητήθηκε στην Cache. Εφόσον υπάρχει μόνο μια πιθανή θέση στην οποία μπορεί να βρίσκεται το Block, η συγκεκριμένη Cache line είτε θα περιέχει το Block είτε όχι.

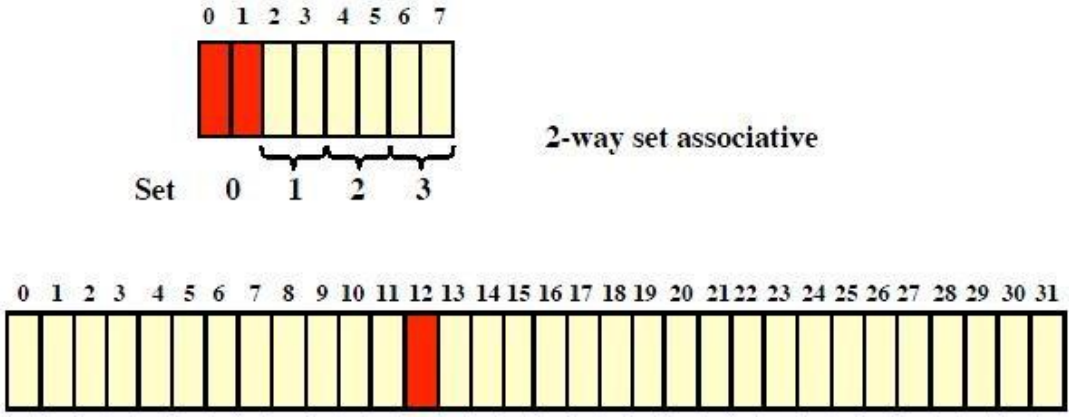
Δυστυχώς όμως, το Direct Mapping παρέχει την χειρότερη επίδοση από τις Mapping Functions, γιατί το σύνηθες είναι να ζητούνται συνεχόμενα Blocks από την Cache, τα οποία ανήκουν στην ίδια Cache Line και επειδή αυτό δεν λαμβάνεται υπόψιν από το Direct Mapping έχουμε μεγάλη καθυστέρηση στην εισαγωγή και αντικατάσταση του κάθε Block.



Σχήμα 3.4 Fully Associative Mapping

Αν ένα Block μπορεί να τοποθετηθεί σε οποιαδήποτε θέση της Cache, τότε η Cache είναι Fully Associative.

Η Fully Associative Cache έχει το καλύτερο Hit Ratio από τις τρεις Mapping Functions και είναι ευέλικτη, αλλά έχει σημαντικά προβλήματα απόδοσης, γιατί πρέπει να γίνει πολύ ψάξιμο για να βρεθεί το ζητούμενο Block, αφού μπορεί να βρίσκεται οπουδήποτε στην Cache. Αυτό καθιστά την Fully Associative Cache αρκετά χρονοβόρα.



Σχήμα 3.5 Set Associative Mapping

Αν ένα Block μπορεί να τοποθετηθεί σε ένα συγκεκριμένο Set θέσεων της Cache, τότε έχουμε Set Associative Cache. Ένα Set είναι μια ομάδα Blocks στην Cache. Αρχικά, το Block γίνεται mapped πάνω σε ένα Set και μετά μπορεί να τοποθετηθεί οπουδήποτε μέσα στο Set αυτό. Η επιλογή του Set γίνεται ως εξης

(Block address) mod (Αριθμός Set Cache στην Cache)

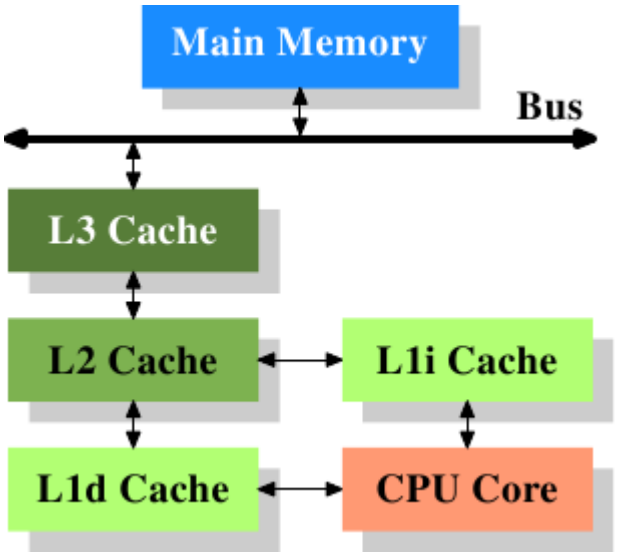
Αν υπάρχουν n Blocks σε ένα Set, τότε η Set Associative Cache ονομάζεται n-way Set Associative.

Η Set Associative Cache είναι ένας καλός συμβιβασμός ανάμεσα στην Direct Mapped και στην Fully Associative Cache, αφού διατηρεί όλα τα θετικά στοιχεία των δύο Mapping Functions, αφού έχει την ευελιξία της Fully Associative Cache και το καλό Hit Ratio της Direct Mapped Cache. Μεγαλύτερα Sets και ψηλότερη associativity οδηγούν σε χαμηλότερα miss rates, αλλά αυξάνουν και το κόστος του υλικού.

Η πολιτική που έχει επικρατήσει για τους σύγχρονους επεξεργαστές είναι Caches είτε Direct Mapped, είτε 2-way ή 4-way Set Associative. Οι 2 τελευταίες αποτελούν τον καλύτερο συμβιβασμό μεταξύ του υψηλού Hit Ratio και της γρήγορης ταχύτητας αναζήτησης των δεδομένων.

Ιεραρχία Cache:

Μαζί με την ευρεία χρήση της cache, κρίθηκε αναγκαίο να αλλάξει και η ιεραρχία μνήμης για την αντιμετώπιση του «Processor – Memory Performance Gap». Έτσι, χρησιμοποιήθηκαν περισσότερα από ένα επίπεδα cache, όπου όσο πιο μικρό ήταν το επίπεδο, τόσο πιο κοντά βρισκόταν η cache στον επεξεργαστή και τόσο πιο μικρές και γρήγορες ήταν. Σήμερα υπάρχουν μέχρι και 3 επίπεδα cache[17], με ονομασίες Level 1 (L1), Level 2 (L2) και Level 3 (L3), όπου η L1 είναι πλησιέστερη στον Επεξεργαστή και η L3 η πιο απομακρυσμένη από αυτόν, όπως φαίνεται στο Σχήμα 3.6. Στο σχήμα βλέπουμε τρία Level cache, όπου L1d είναι το level 1 data cache, το L1i είναι το level 1 instruction cache κτλ. Επισημαίνεται ότι αυτή είναι μια σχηματική αναπαράσταση και στην πραγματικότητα δε χρειάζεται τα δεδομένα να περάσουν από τις πιο ψηλού επιπέδου caches για να πάνε από τον πυρήνα στην κύρια μνήμη.



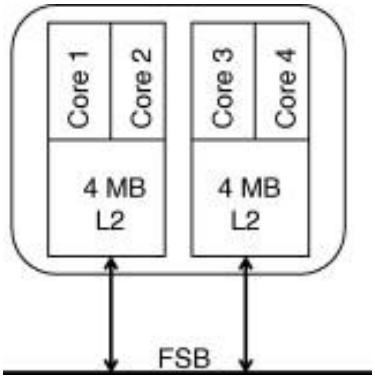
Σχήμα 3.6 Οργάνωση Multi-level Cache

Η διαδικασία που ακολουθείται όταν ο επεξεργαστής χρειάζεται ένα block είναι η εξης: Αρχικά ο επεξεργαστής θα αναζητήσει το block από την Level 1 cache και αν υπάρχει εκεί δεν υπάρχει καθυστέρηση στην απόδοση. Σε αντίθετη περίπτωση, το block πρέπει να ζητηθεί από την Level 2 cache. Αν υπάρχει στην level 2 cache θα υπάρξει σχετικά μικρή καθυστέρηση, αλλιώς η ίδια διαδικασία θα επαναληφθεί για την Level 3 cache. Αν το block δεν βρίσκεται ούτε στην cache αυτού του επιπέδου θα πρέπει να ζητηθεί από την κύρια μνήμη και αυτό θα προκαλέσει αρκετή καθυστέρηση. Ακόμη μεγαλύτερη καθυστέρηση θα προκληθεί αν το block δεν υπάρχει ούτε στη μνήμη και πρέπει να ζητηθεί από τα πιο αργά μέσα αποθήκευσης, τους σκληρούς δίσκους. Άρα η Cache ουσιαστικά προσπαθεί να προβλέψει ποια δεδομένα θα χρειαστεί στην συνέχεια ο επεξεργαστής και να του τα διαθέσει, ώστε να αποφευχθεί η μεγάλη καθυστέρηση.

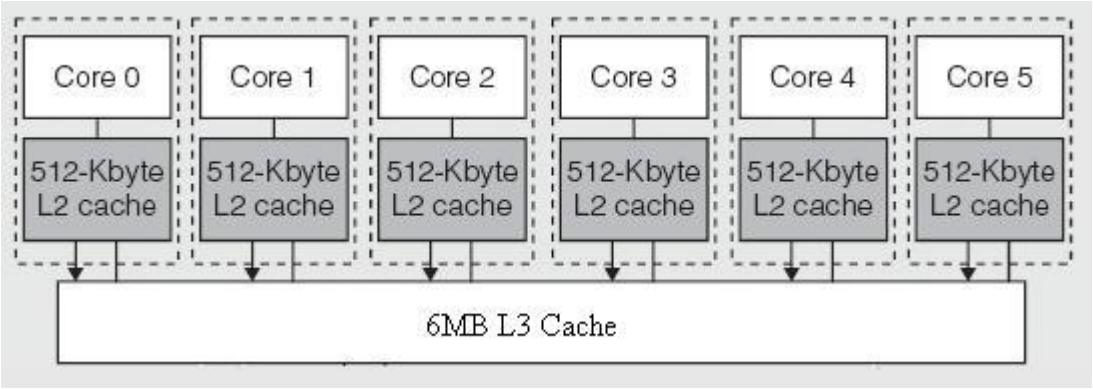
Η ύπαρξη των Multi-Level Caches λύνει και το πρόβλημα της αναλογίας μεταξύ hit ratio και καθυστέρησης (latency). Caches με σχετικά μεγάλο μέγεθος, έχουν μεγάλα ποσοστά hit ratio αλλά και μεγάλη καθυστέρηση, ενώ αντίστροφα, caches με μικρά μεγέθη έχουν μικρή καθυστέρηση αλλά και μικρό hit ratio. Με τα πολλαπλά επίπεδα Cache έχουμε εξομάλυνση του προβήματος αυτού αφού έχουμε σταδιακή αύξηση των μεγεθών και του latency των caches.

Επίσης, με την ύπαρξη πολλαπλών πυρήνων σε επεξεργαστές και πολλαπλών νημάτων (threads) σε κάθε πυρήνα ενθαρρύνεται η Multi-Level Οργάνωση των μνημών cache. Στην περίπτωση αυτή έχουμε κοινή χρήση της L1 cache μεταξύ των Threads, σε κάθε πυρήνα και όσον αφορά τους πυρήνες, έχουν γενικά κοινόχρηστες τις L2 και L3 caches, ενώ ο κάθε ένας έχει την δική του L1.

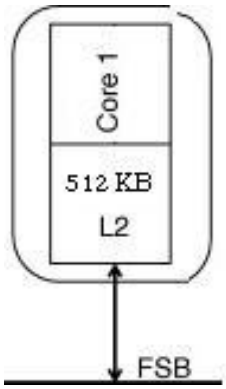
Στα πιο κάτω σχήματα φαίνεται η ιεραρχία Cache για τις τρεις μηχανές που χρησιμοποιώ για την εργασία μου. Στο Σχήμα 3.7 φαίνεται η ιεραρχία Cache της csteras, στο Σχήμα 3.8 φαίνεται η ιεραρχία Cache της cs6472 και στο Σχήμα 3.9 η ιεραρχία Cache της csatom.



Σχήμα 3.7 Ιεραρχία Cache της csteras



Σχήμα 3.8 Ιεραρχία Cache της cs6472



Σχήμα 3.9 Ιεραρχία Cache της csatom

Όπως φαίνεται από τα σχήματα, η csteras έχει 4 πυρήνες και κάθε 2 πυρήνες έχουν κοινόχρηστη L2 cache μεγέθους 4 MB. Η cs6472 έχει 6 πυρήνες και κάθε πυρήνας έχει L2 cache μεγέθους 512 KB, ενώ όλοι οι πυρήνες έχουν κοινόχρηστη L3 cache μεγέθους 6 MB. Η csatom έχει ένα πυρήνα με L2 cache μεγέθους 512 KB.

Μέγεθος Cache:

Όπως προαναφέρθηκε, caches με σχετικά μεγάλο μέγεθος, έχουν μεγάλα ποσοστά hit ratio αλλά και μεγάλη καθυστέρηση, ενώ αντίστροφα, caches με μικρά μεγέθη έχουν μικρή καθυστέρηση αλλά και μικρό hit ratio. Αυτό συμβαίνει γιατί όσο πιο μεγάλη είναι μια cache, τόσο πιο πολλά δεδομένα μπορεί να κρατήσει, και άρα τόσο πιο πολλές αναφορές θα μπορεί να εξυπηρετήσει. Όμως μεγάλες cache είναι και πιο αργές, άρα έχουν μεγαλύτερη καθυστέρηση.

Μνήμη RAM (RAM Memory):

Η μνήμη RAM είναι ένα μέσο αποθήκευσης του υπολογιστή, το οποίο αποθηκεύει προγράμματα για εκτέλεση και δεδομένα για επεξεργασία. Η χωρητικότητα της μνήμης RAM μετράται σε bytes και τα πολλαπλάσιά τους. Το μέγεθος της χωρητικότητας της μνήμης RAM είναι καθοριστικό για τη σταθερότητα και την ομαλή λειτουργία ενός υπολογιστικού συστήματος. Ανεπαρκές μέγεθος μνήμης μπορεί να μειώσει κατά πολύ την απόδοση ενός υπολογιστή.

Μέγιστη TDP:

Η μέγιστη TDP[18] αναφέρεται στη μέγιστη ποσότητα ενέγειας που χρειάζεται να διαλύει το cooling system μιας μηχανής. Είναι η μέγιστη ενέργεια που υπολογίζεται ότι θα αντλήσει η μηχανή κατά την εκτέλεση πραγματικών εφαρμογών.

Release Price:

Η τιμή πώλησης την στιγμή που βγήκε στην αγορά η μηχανή.

Κεφάλαιο 4

Υπό μελέτη μετρήσεις

4.1 Cache references

Τα Cache references είναι οι αναφορές στη μνήμη cache, δηλαδή πόσες φορές ζητήθηκαν δεδομένα από την cache.

Τα Cache references που συλλέγει η εντολή perf stat, την οποία χρησιμοποιώ, είναι οι αναφορές της cache τελευταίου επιπέδου (last level cache)[19], δηλαδή της cache που βρίσκεται πιο μακριά από τον επεξεργαστή σε σχέση με τις υπόλοιπες.

4.2 Cache Misses

Τα cache misses[20] είναι αναφορές (cache references) σε δεδομένα που δεν βρίσκονται στη μνήμη cache και έτσι τα δεδομένα που χρειάζονται πρέπει να προσκομιστούν από την κύρια μνήμη και να τοποθετηθούν στην cache.

Άρα, ένα cache miss συμβαίνει όταν υπάρξει αποτυχής προσπάθεια για διάβασμα ή γράψιμο στη μνήμη cache και κατά συνέπεια χρειάζεται να γίνει πρόσβαση στην κύρια μνήμη. Τα cache misses επηρεάζουν την απόδοση της εκτέλεσης, γιατί προκαλείται μεγαλύτερη καθυστέρηση όταν τα δεδομένα προσκομίζονται από την κύρια μνήμη αντί από την cache, γιατί ο επεξεργαστής πρέπει να περιμένει μέχρι τα δεδομένα να μεταφερθούν από τη μνήμη για να συνεχίσει την εκτέλεση του προγράμματος. Το πόσο

θα επηρεαστεί η επίδοση από ένα cache miss εξαρτάται από το μέγεθος της καθυστέρησης για να γίνει η ανάκτηση των δεδομένων από το επόμενο επίπεδο της cache ή από την κύρια μνήμη.

Υπάρχουν τρία είδη Cache Miss: το instruction read miss, το data read miss, και το data write miss.

1) Ένα instruction read miss, δηλαδή ένα cache miss σε εντολή fetch, συνήθως προκαλεί τη μεγαλύτερη καθυστέρηση, γιατί απαιτεί από τον επεξεργαστή να σταματήσει την εκτέλεση και να περιμένει μέχρι η εντολή να είναι διαθέσιμη από την κύρια μνήμη.
2) Ένα data read miss, δηλαδή ένα cache miss σε διάβασμα δεδομένων, συνήθως προκαλεί μικρότερη καθυστέρηση, λόγω του ότι οι εντολές που δεν θα διαβαστούν από την cache μπορούν να συνεχίσουν την εκτέλεσή τους μέχρι να έρθουν τα δεδομένα από την κύρια μνήμη και να συνεχίσουν την εκτέλεσή τους και οι εντολές που περιμένουν.
3) Ένα data write miss είναι το λιγότερο σοβαρό, γιατί συνήθως προκαλεί τη μικρότερη καθυστέρηση. Το γράψιμο μπορεί να μπει σε ουρά (queue) και προκαλεί ελάχιστους περιορισμούς σε επόμενες εντολές, αφού ο επεξεργαστής μπορεί να συνεχίσει την εκτέλεση μέχρι να γεμίσει η ουρά.

Επίσης υπάρχει και μία ειδική περίπτωση των cache misses, τα cache misses που προκαλούνται από prefetch οδηγίες και δεν προκαλούν καθόλου καθυστέρηση, όμως πυροδοτούν ένα fetch των δεδομένων έτσι ώστε επόμενες προσβάσεις να μην έχουν cache miss.

Τα Cache misses που συλλέγει η εντολή perf stat, την οποία χρησιμοποιώ, είναι τα misses της cache τελευταίου επιπέδου (last level cache)[19], δηλαδή της cache που βρίσκεται πιο μακριά από τον επεξεργαστή σε σχέση με τις υπόλοιπες και άρα προκαλεί και τη μεγαλύτερη καθυστέρηση.

Είναι δύσκολο να αποφανθούμε μόνο από τον αριθμό των cache misses αν αυτές δημιουργούν τα προβλήματα επίδοσης σε μια εφαρμογή. Μια πιο χρήσιμη μετρική είναι η cache miss rate για το οποίο αναφέρομαι στην επόμενη υποενότητα.

4.3 Cache miss rate

Το cache miss rate[20] είναι η αναλογία μεταξύ των cache misses και του ολικού αριθμού προσβάσεων στη μνήμη (cache references). Για να υπολογίσουμε το cache miss rate διαιρούμε τον αριθμό των cache misses δια τον αριθμό των cache references. Από τη μετρική αυτή μπορούμε να καταλάβουμε αν οι cache misses προκαλούν πρόβλημα στην επίδοση της εφαρμογής, αφού όσο πιο μεγάλο είναι cache miss rate, σημαίνει ότι τόσο πιο πολλά είναι τα cache misses και άρα τόσο μεγαύτερη καθυστέρηση προκαλείται στη λειτουργία του επεξεργαστή.

Το cache miss rate μιας εφαρμογής εξαρτάται από το μέγεθος της cache. Όσο μεγαλύτερη είναι η cache, τόσο περισσότερα δεδομένα μπορεί να κρατήσει και άρα τόσο μικρότερος αναμένεται να είναι ο αριθμός του cache miss rate.

4.4 Χρόνος εκτέλεσης

Είναι ο χρόνος που χρειάζεται για την εκτέλεση κάθε query. Είναι μια μετρική που αφορά άμεσα ένα χρήστη, γιατί είναι σημαντικό γι’ αυτόν να μπορεί η μηχανή να τελειώσει μια εργασία γρήγορα. Μηχανές με πιο μεγάλη ταχύτητα επεξεργαστή αναμένονται να έχουν μικρότερους χρόνους εκτέλεσης από μηχανές με πιο αργούς επεξεργαστές.

Κεφάλαιο 5

Μεθοδολογία και Αποτελέσματα

5.1 Μεθοδολογία

Για να γίνει η σύγκριση των μηχανών χρειαζόταν να πάρω αρχικά τις μετρήσεις που προαναφέρθηκαν. Για τη μέτρηση αυτή χρησιμοποίησα το σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL, τα queries που προσφέρουν τα TPC-H benchmarks και το perf του λειτουργικού συστήματος Linux. Συγκεκριμένα, δημιούργησα μια βάση, η οποία προσφέρεται από του TPC-H benchmark και στη συνέχεια έτρεξα τα queries του TPC-H benchmark στην PostgreSQL, συλλέγοντας ταυτόχρονα τις μετρήσεις που δίνουν οι performance counters με τη χρήση του perf. Στα επόμενο κεφάλαιο θα αναφερθώ στο τί είναι η PostgreSQL, το TPC-H benchmark, οι performance counters και το Linux perf.

Η μεθοδολογία που ακολούθησα για την εργασία μου είναι η εξης: Αρχικά έκανα εγκατάσταση της ίδιας έκδοσης της PostgreSQL (έκδοση 8.4) σε όλες τις μηχανές με την πιο κάτω εντολή.

```
apt-get install postgresql-8.4
```

Η PostgreSQL χρησιμοποιήθηκε για την αποθήκευση και επεξεργασία της βάσης που δημιούργησα με τη χρήση των συστατικών στοιχείων που προσφέρει το TPC-H benchmark, το οποίο κατέβασα σε έκδοση 2.14.0. Συγκεκριμένα, με τις εντολές[1]

που φαίνονται πιο κάτω δημιούργησα μια κενή βάση με scale factor 1, δηλαδή μεγέθους 1GB και όνομα <name>.

```
./dbgen -s 1

postgres createdb <name>
```

Έπειτα, για να μπορέσω να γεμίσω τη βάση με δεδομένα ενώθηκα με τον server της PostgreSQL με την εξης εντολή

```
postgresql-8.4 start
```

Στη συνέχεια, άνοιξα τη βάση <name>

```
postgres psql <name>
```

και χρησιμοποιώντας τα .tbl αρχεία που αντιπροσωπεύουν τα tables της βάσης και το αρχείο dss.ddl, που περιέχει εντολές για τη δημιουργία των tables, δημιούργησα το σχήμα της βάσης, με την εντολή που φαίνεται πιο κάτω.

```
\i /home/casper/tpch_2_14_0/dbgen/dss.ddl
```

Έπειτα δημιούργησα τα tables μέσα στη βάση με την εντολή

```
COPY <tablename> FROM '<filename>' WITH DELIMITER AS '<delimiter>';
```

όπου <tablename> είναι το όνομα του table, <filename> το όνομα του tbl. αρχείου που περιέχει τον ορισμό του table και <delimeter> είναι το διαχωριστικό μεταξύ των

στηλών του table. Σύμφωνα με το documentation TPC-H benchmark[1] χρησιμοποίησα το σύμβολο '|' σαν delimeter.

Για την ολοκλήρωση της βάσης, εκτέλεσα το αρχείο dss.ri, για δημιουργία των constraints, δηλαδή των primary keys, foreign keys και indexes

```
\i /home/casper/tpch_2_14_0/dbgen/dss.ri
```

Μετά τη δημιουργία της βάσης εκτέλεσα κάθε query των TPC-H benchmarks ξεχωριστά σε κάθε μηχανή και με τη χρήση των performance counters μπόρεσα να πάρω μετρήσεις για τα χαρακτηριστικά που ήθελα να μελετήσω. Για πιο αντιπροσωπευτικά αποτελέσματα, εκτέλεσα το κάθε query 5 φορές, αφαίρεσα την μεγαλύτερη και τη μικρότερη από τις 5 τιμές και πήρα το μέσο όρο από τις τρεις τιμές που έμειναν. Συγκεκριμένα, η εντολή που χρησιμοποίησα για να πάρω τα αποτελέσματά μου βρισκόταν σε ένα bash αρχείο και είχε την εξης μορφή

```
for i in 1 2 3 4 5
do
perf stat -e cache-misses,cache-references psql test1 -f <filename> 1>
<outputfile> 2> <perf_results_file>
done
```

όπου test1 είναι το όνομα της βάσης, το –f χρησιμοποιείται για να δηλώσει ότι οι εντολές πρέπει να εκτελεστούν από το αρχείο filename, το οποίο είχε extension .sql και το αρχείο outputfile αποθηκεύει το αποτέλεσμα του query, ενώ το perf_results_file αποθηκεύει τις μετρήσεις που έλαβε το perf.

Μετά από την εξαγωγή των αποτελεσμάτων αυτών και μετά από μελέτη των διαφορετικών χαρακτηριστικών κάθε μηχανής, έφτασα σε κάποια συμπεράσματα σχετικά με το πώς τα χαρακτηριστικά κάθε μηχανής επηρεάζουν την εκτέλεση των queries.

5.2 Αποτελέσματα

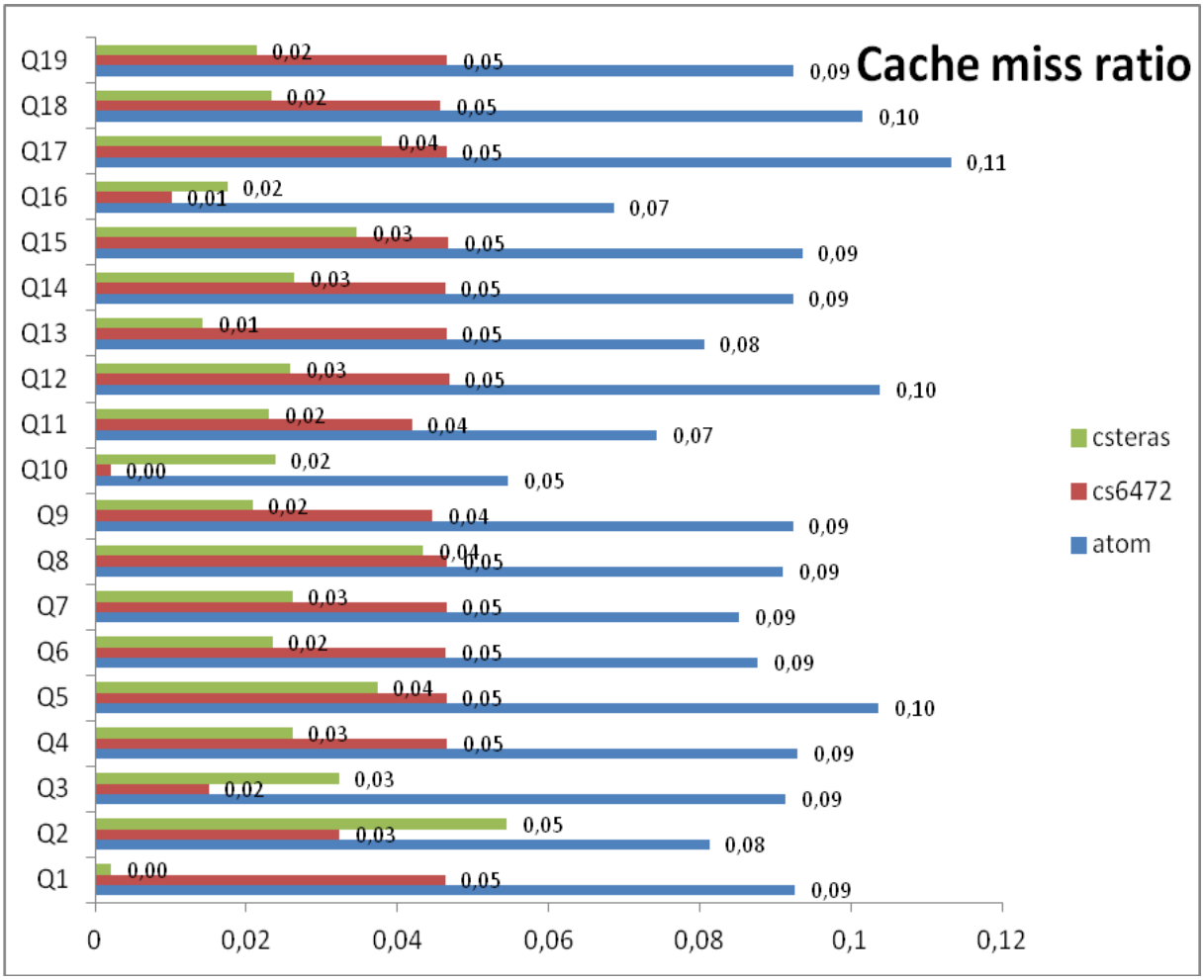
Χρησιμοποιώντας τις μετρήσεις που πήρα για τα Cache misses, Cache References και το Χρόνο εκτέλεσης, δημιούργησα κάποιες γραφικές για εύκολη ανάλυση των μετρήσεων και σύγκριση των αποτελεσμάτων για κάθε μηχανή. Οι μετρήσεις των Cache misses και Cache References χρησιμοποιήθηκαν για τη δημιουργία της γραφικής του Cache miss ratio.

Οι γραφικές για τα Cache misses και Cache References δεν θα παρουσιαστούν, λόγω του ότι οι performance counters κάθε επεξεργαστή δίνουν αποτελέσματα σχετικά με τον συγκεκριμένο επεξεργαστή και άρα δεν μπορούμε εύκολα να συγκρίνουμε τις τιμές των Cache misses και Cache References από μετρητές διαφορετικών επεξεργαστών. Γι'αυτό είναι προτιμότερο να πάρουμε τη σχετική μετρική Cache miss ratio, η οποία μπορεί να χρησιμοποιηθεί για σύγκριση μεταξύ των τριών διαφορετικών μηχανών.

Για πιο αξιόπιστα αποτελέσματα, από τις μετρήσεις που πήρα αφαίρεσα την μικρότερη και τη μεγαλύτερη από τις πέντε τιμές, ώστε να αποφύγω μεγάλες διαφορές στις τιμές και πήρα το μέσο όρο για τις εναπομείναντες 3 τιμές. Πιο κάτω φαίνονται οι γραφικές, με κάποια σχόλια.

5.2.1 Cache miss rate

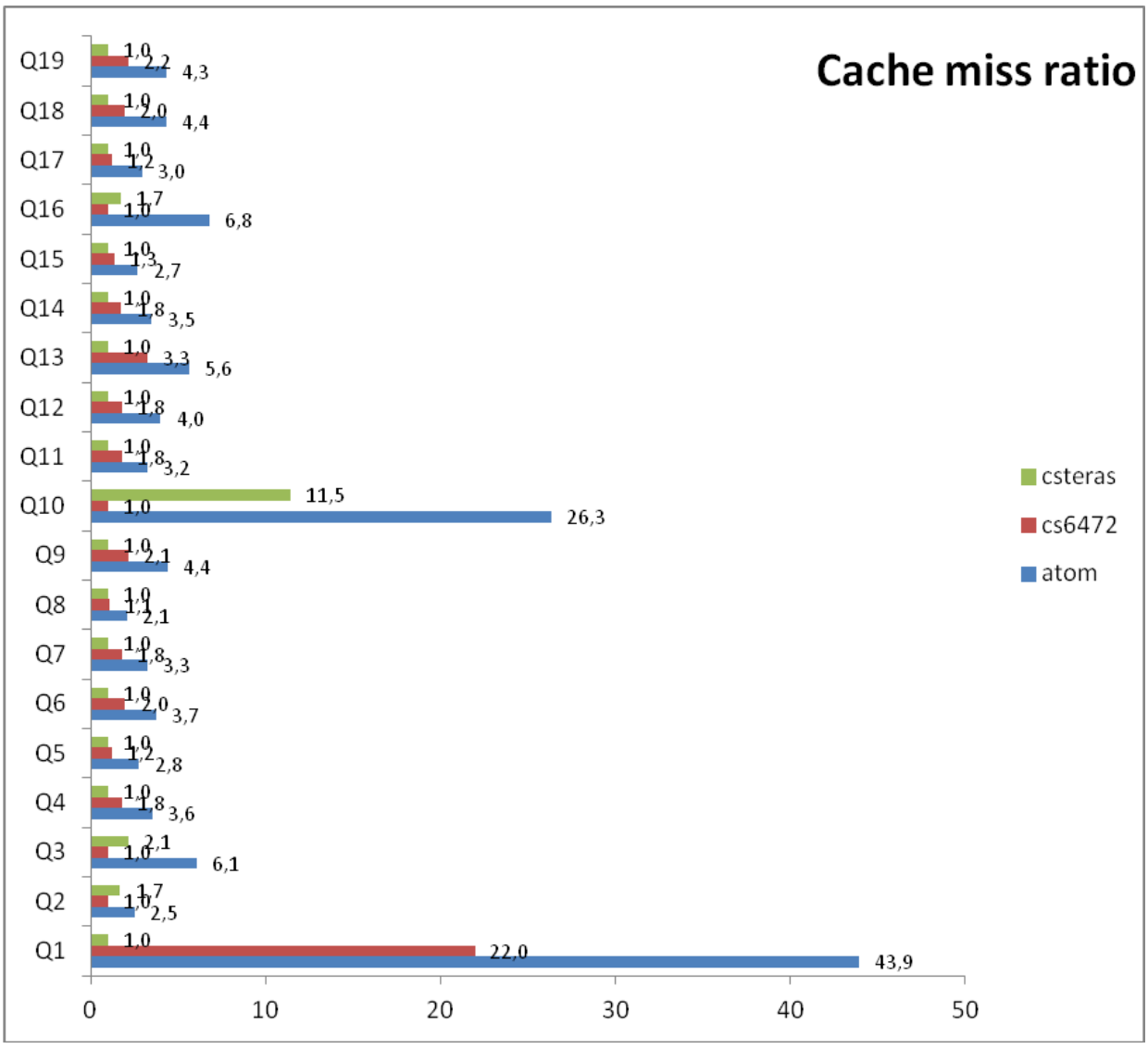
Στη γραφική παράσταση του σχήματος Σχήμα 5.1 φαίνονται τα αποτελέσματα του cache miss ratio, δηλαδή της αναλογίας των Cache misses προς τα Cache references. Ο υπολογισμός του cache miss ratio έγινε με διαίρεση των cache misses δια τα cache references.



Σχήμα 5.1 Cache miss rate μηχανών για όλα τα queries

Από τη γραφική φαίνεται ότι η μηχανή csteras έχει για όλα τα queries, εκτός των Q2, Q3, Q10 και Q16, το μικρότερο cache miss ratio. Αντίθετα, η μηχανή atom, σε σχέση με τις άλλες μηχανές έχει πάντα το μεγαλύτερο cache miss rate, με τις τιμές του rate να είναι αρκετά μεγαλύτερες από το cache miss rate των άλλων δύο μηχανών.

Η γραφική παράσταση του σχήματος Σχήμα 5.2 είναι η γραφική του Σχήματος 5.1 τροποποιημένη έτσι ώστε να μην φαίνονται οι τιμές του cache miss ratio, αλλά η αναλογία μεταξύ των τιμών για τις τρεις μηχανές για κάθε query. Συγκεκριμένα, για κάθε query, βρήκα τη μικρότερη (min) από τις τρεις τιμές για τις διαφορετικές μηχανές και διαίρεσα κάθε μία από τις τιμές αυτές δια τη min, ώστε να φαίνεται αναλογικά η διαφορά μεταξύ των τιμών.



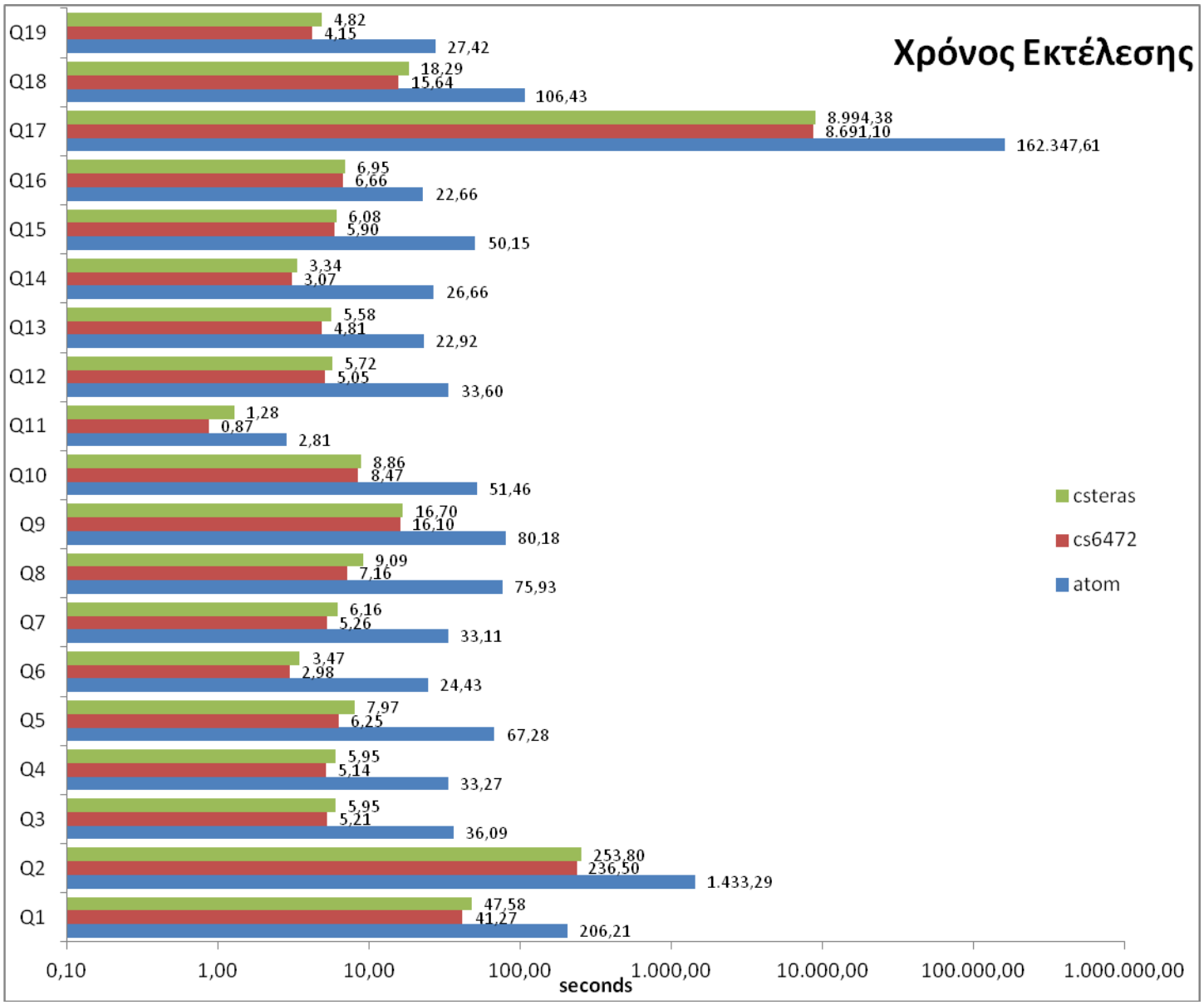
Σχήμα 5.2 Cache miss ratio σε αναλογία

Από την πιο πάνω γραφική φαίνεται ότι, εκτός από κάποιες εξαιρέσεις (Q1 και Q10), η μηχανή cs6472 έχει αναλογικά 0,8 με 2 φορές πιο μεγάλο cache miss ratio από την

csteras, ενώ η atom κυμαίνεται μεταξύ 2 με 7 φορών πιο μεγάλου cache miss ratio από την csteras και πάλι με τις ίδιες εξαιρέσεις (Q1 και Q10).

5.2.2 Χρόνος εκτέλεσης

Στη γραφική παράσταση του σχήματος Σχήμα 5.3 φαίνονται τα αποτελέσματα για το χρόνο εκτέλεσης των queries για κάθε μηχανή, σε δευτερόλεπτα (seconds).

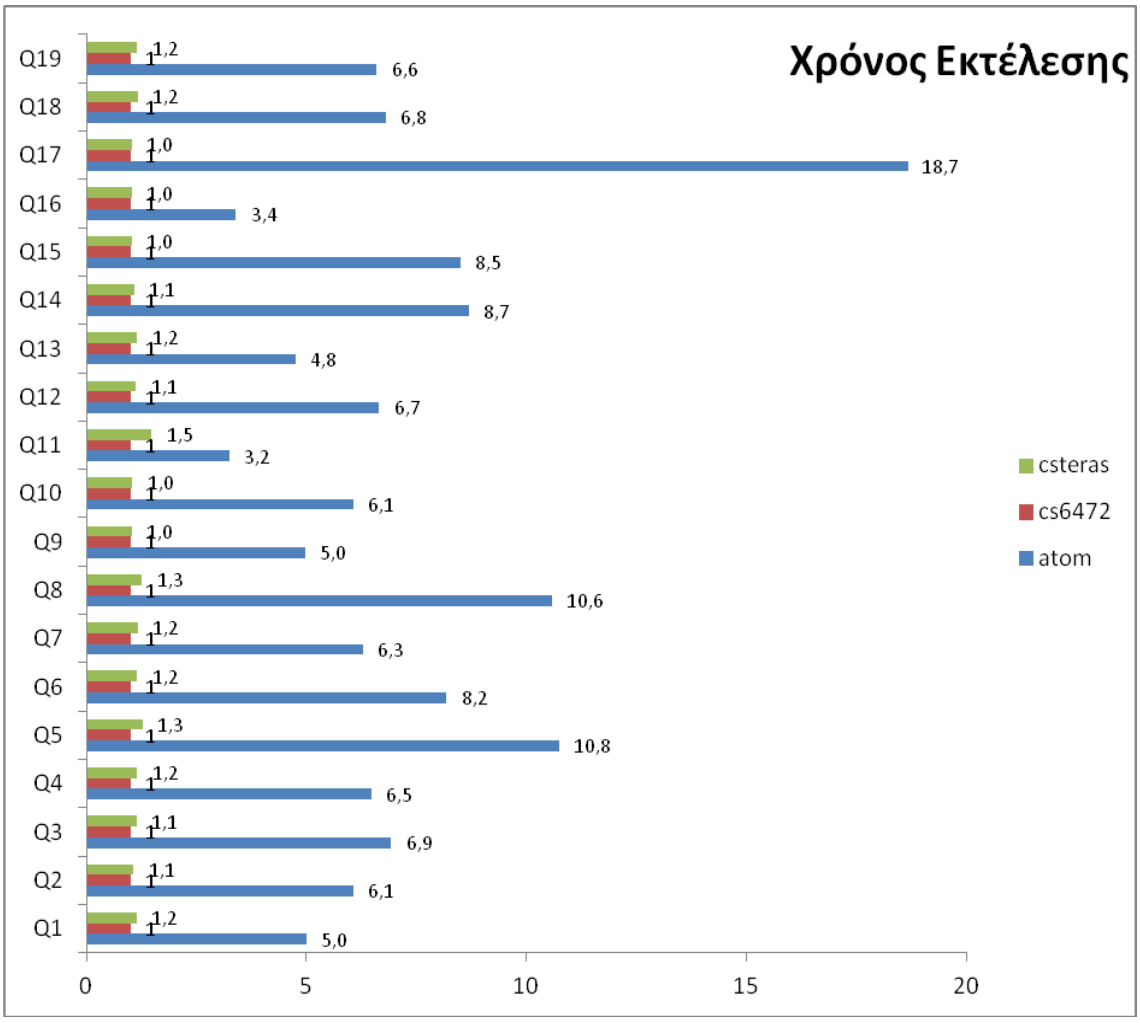


Σχήμα 5.3 Χρόνος εκτέλεσης για όλα τα queries

Από τη γραφική παράσταση φαίνεται ότι η μηχανή που δίνει πάντα το μεγαλύτερο χρόνο εκτέλεσης είναι η atom, ενώ η cs6472 δίνει πάντα το μικρότερο χρόνο. Επίσης, φαίνεται ότι η διαφορά μεταξύ των χρόνων εκτέλεσης της cs6472 και της csteras είναι

πολύ μικρή. Αξιοσημείωτη είναι η μεγάλη διαφορά στο χρόνο εκτέλεσης για το Q17, το οποίο είναι το μόνο query που απαιτεί να γίνει ανάγνωση δεδομένων από όλη τη βάση και άρα ζητά το διάβασμα ενός μεγάλου αριθμού δεδομένων. Για την εκτέλεση του query αυτού φαίνεται ότι η atom χρειάζεται πολύ περισσότερο χρόνο από τις άλλες μηχανές.

Η γραφική παράσταση του σχήματος Σχήμα 5.4 είναι η γραφική παράσταση του Σχήματος 5.3 τροποποιημένη ώστε να μη φαίνονται οι τιμές του χρόνου που έδωσαν οι counters, αλλά να βλέπουμε το χρόνο σε αναλογία. Συγκεκριμένα, για κάθε query διαιρέθηκαν οι τιμές των τριών μηχανών δια τη μικρότερη τιμή από τις τρεις, ώστε να φαίνεται πόσες φορές πιο αργή ή πιο γρήγορη ήταν η μία μηχανή από την άλλη.



Σχήμα 5.4 Χρόνος εκτέλεσης για όλα τα queries σε αναλογία

Από τη γραφική αυτή φαίνεται ότι η αναλογία μεταξύ των χρόνων εκτέλεσης των μηχανών csteras και cs6472 κυμαίνεται μεταξύ του 0,1 με 0,3 δηλαδή η csteras είναι 0,1 με 0,3 φορές πιο αργή από την cs6472. Αντίθετα, η atom είναι συνήθως 7 με 8 φορές πιο αργή από την cs6472 και για το Q17, όπου έχουμε δραματική διαφορά στο χρόνο, η μηχανή atom είναι 18 φορές πιο αργή από τις δύο άλλες μηχανές.

Κεφάλαιο 6

Συμπεράσματα

6.1 Συμπεράσματα για εκτέλεση queries

Πιο κάτω θα αναφερθώ στα συμπεράσματα που εξήγαγα από τα αποτελέσματα για κάθε μηχανή ξεχωριστά. Θα αναφερθώ στο πώς συμπέρανα ότι επηρεάζουν τα χαρακτηριστικά κάθε μηχανής τις διάφορες μετρήσεις που πήρα.

6.1.1 Μηχανή csteras

Η μηχανή csteras διαθέτει μνήμη cache μεγέθους 4096 KB, η οποία φαίνεται ότι της δίνει τη δυνατότητα να έχει μικρό αριθμό cache misses σε σχέση με τα cache references της. Άρα, το μεγάλο μέγεθος μνήμης cache βοηθά τη μηχανή να τρέχει πιο αποδοτικά, αφού δεν χρειάζεται να περιμένει πολλές φορές για δεδομένα που βρίσκονται εκτός μνήμης cache και να σπαταλά χρόνο μέχρι τα δεδομένα αυτά να παρθούν από την κύρια μνήμη.Επίσης, η csteras διαθέτει αρκετά γρήγορο επεξεργαστή, ταχύτητας 1.86 GHz, κάτι που τη βοηθά να εκτελεί τα queries σε μικρούς σχετικά χρόνους.

Συνδυαστικά, τα χαρακτηριστικά της μηχανής αυτής φαίνεται να την καθιστούν αρκετά αποτελεσματική στην εκτέλεση των queries.

6.2.2 Μηχανή cs6472

Η μηχανή csteras διαθέτει μνήμη cache μικρού μεγέθους 512 KB και γρήγορο επεξεργαστή, ταχύτητας 2.2 GHz, κάτι που φαίνεται ότι τη βοηθά να κρατά το cache miss ratio της σε χαμηλά επίπεδα, ενώ τρέχει με γρήγορους χρόνους εκτέλεσης.

Συνδυαστικά, τα χαρακτηριστικά της μηχανής αυτής φαίνεται να την καθιστούν αρκετά αποτελεσματική στην εκτέλεση των queries.

6.3.3 Μηχανή csatom

Η μηχανή csatom διαθέτει μνήμη cache μικρού μεγέθους 512 KB και επεξεργαστή ταχύτητας 1.6 GHz και φαίνεται ότι ο συνδυασμός των δύο αυτών χαρακτηριστικών δεν βοηθούν τη μηχανή με το χρόνο εκτέλεσής της, ειδικά για απαιτητικά queries. Όσον αφορά το cache miss ratio της, δεν βρίσκεται σε πολύ ψηλά επίπεδα.

Συνδυαστικά, τα χαρακτηριστικά της μηχανής αυτής φαίνεται να την περιορίζουν λίγο στην αποτελεσματικότητά της σε σχέση με την εκτέλεση των queries.

6.2 Σύγκριση μηχανών

Λαμβάνοντας υπόψιν την συχνότητα επεξεργαστή των μηχανών, παρατηρούμε ότι η cs6472 έχει περίπου 20% μεγαλύτερη συχνότητα επεξεργαστή από την csteras και περίπου 40% μεγαλύτερη συχνότητα από την csatom. Αυτό μας οδηγεί στο συμπέρασμα ότι επεξεργαστής της μπορεί να λειτουργεί με μεγαλύτερη ταχύτητα από τις άλλες μηχανές και είναι ένας από τους λόγους που η cs6472 έχει πάντα καλύτερους χρόνους από τις άλλες μηχανές.

Όσον αφορά τις cache των μηχανών, η csteras έχει ένα μεγάλο μέγεθος L2 cache, περισσότερο από 2.5 φορές πιο μεγάλη από την L2 cache της cs6472 και 16 φορές πιο μεγάλη από της csatom. Επίσης, έχει 4 φορές περισσότερη L2 cache διαθέσιμη ανά πυρήνα σε σχέση με τις άλλες δύο μηχανές. Αυτό της δίνει τη δυνατότητα να

εξυπηρετεί περισσότερες αναφορές από τις άλλες δύο μηχανές, όπως φαίνεται από το cache miss ratio της. Όμως, η κοινόχρηστη L3 cache μεγέθους 6 MB, της δίνει ένα cache miss ratio πολύ κοντινό και κάποιες φορές μικρότερο από αυτό της csteras και η ιεραρχία μνήμης της, την καθιστά πιο γρήγορη από την csteras, έστω κι αν έχει μεγαλύτερο cache miss ratio. Είναι φανερό ότι οι μικρού μεγέθους L2 caches, όπως αυτές της cs6472 και της csatom (512 KB), αν και γρήγορες λόγω μικρού μεγέθους, χρειάζεται να συνοδεύονται και από L3 cache, η οποία μειώνει σημαντικά το cache miss ratio και το χρόνο εκτέλεσης. Επίσης, μπορούμε να συμπεράνουμε ότι το set-associativity των cs6472 και csteras τις βοηθά στο να διατηρούν το cache miss ratio τους χαμηλό.

Με βάση τους πυρήνες, η cs6472 υπερτερεί έναντι των άλλων 2 μηχανών γιατί έχει τους περισσότερους πυρήνες, κάτι που την ευνοεί στην εκτέλεση περισσότερων εντολών ή και εφαρμογών ταυτόχρονα, ενώ ακολουθεί η csteras, με 2 λιγότερους πυρήνες από την cs6472 και μετά έχουμε την csatom, η οποία έχει μόνο ένα πυρήνα. Κατά την εκτέλεση των πειραμάτων μου δεν χρησιμοποιήσα παραλληλισμό και άρα τα αποτελέσματα δεν βασίζονται στον αριθμό των πυρήνων των μηχανών, αλλά αυτό δεν αναιρεί το γεγονός ότι η cs6472 και η csteras είναι πολυπύρηνες και με τις κατάλληλες ρυθμίσεις μπορούν να εκτελέσουν πολύ περισσότερη δουλειά από την csatom σε μικρότερους χρόνους.

Συγκρίνοντας τις μηχανές με βάση την κύρια μνήμη τους βλέπουμε ότι η cs6472 έχει περίπου διπλάσιο μέγεθος μνήμης από την csteras και 30 φορές πιο πολλή μνήμη από την csatom. Επειδή η βάση που δημιούργησα για την εκτέλεση των queries είχε μέγεθος 1 GB και εφόσον η csatom έχει 1 GB μνήμη, είναι λογικό ότι κάποιες αναφορές θα έγιναν στον σκληρό δίσκο της μηχανής και αυτό εξηγεί περισσότερο και τους μεγάλους χρόνους εκτέλεσης που έχει η csatom.

Η μηχανή csatom έχει συγκριτικά πολύ χαμηλότερο TDP από τις άλλες δύο μηχανές, καθώς και πολύ χαμηλότερη τιμή, αλλά αυτό οφείλεται στο γεγονός ότι η csatom είναι Laptop, ενώ οι άλλες 2 μηχανές είναι Servers.

Η μηχανή csteras είναι η ακριβότερη από τις τρεις και λαμβάνοντας υπόψιν τα χαρακτηριστικά που προαναφέραμε και το ότι έχει λιγότερα συστατικά στοιχεία (δεν έχει L3 cache και έχει λιγότερα Sets και χαμηλότερο associativity) από την cs6472, θεωρώ ότι είναι αρκετά ακριβή.

6.3 Μελλοντική εργασία

Η εργασία αυτή μελέτησε τα χαρακτηριστικά τριών μηχανών και το πώς επηρεάζουν ένα αριθμό μετρικών, οι οποίες υποδυκνείουν ένα μέρος της απόδοσης των μηχανών αυτών. Ενδιαφέρον θα ήταν στο μέλλον, η εργασία αυτή να επεκταθεί, με την συμπερίληψη περισσότερων μετρικών για αποτελέσματα που θα δίνουν μια πιο ολοκληρωμένη εικόνα για την συνολική απόδοση των μηχανών. Επίσης, καλό θα ήταν να μελετηθούν κι άλλες μηχανές και να ληφθούν υπόψιν κι άλλα χαρακτηριστικά των μηχανών, ώστε να είναι πιο μεγάλη και περιεκτική η έρευνα.

Βιβλιογραφία

[1] *PostgreSQL 8.4.11 Documentation*, The PostgreSQL Global Development Group, 1996-2009.

[2] Peter Zaitsev. (2008, Apr.). TPC-H Run on MySQL 5.1 and 6.0. [Online]. Available: http://www.mysqlperformanceblog.com/2008/04/10/tpc-h-run-on-mysql-51-and-60/

[3] Vadim Tkachenko. (2008, Mar.). MySQL 6.0 vs 5.1 in TPC-H queries. [Online]. Available: http://www.mysqlperformanceblog.com/2008/03/25/mysql-60-vs-51-in-tpc-h-queries/

[4] Pilho Kim. (2011, Mar.). Project/EFIM/TPC-H. [Online]. Available: http://www.pilhokim.com/index.php?title=Project/EFIM/TPC-H

[5] Hennie de Nooijer. (2012, Sep.). Using the TPC-H Benchmark. [Online]. Available: http://bifuture.blogspot.com/2012/09/using-tpc-h-benchmark.html

[6] *TPCH Benchmark™ H Standard Specification Revision 2.14.0*, Transaction Processing Performance Council (TPC), Presidio of San Fransisco, 1993-2011.

[7] Konstantin Boyanov. (2011, Nov.). [Online]. Available: https://dvinfo.ifh.de/perf

[8] Red Hat, Inc. (2013). Performance Counters for Linux (PCL) Tools and perf. [Online]. Available: https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Developer_Guide/perf.html

[9] (2013, May). Linux kernel profiling with perf. [Online]. Available: https://perf.wiki.kernel.org/index.php/Tutorial

[10] (2013, May). Counting with perf stat. [Online]. Available: https://perf.wiki.kernel.org/index.php/Tutorial#Counting_with_perf_stat

[11] Intel Corporation. Intel® Xeon® Processor E5320. [Online]. Available: http://ark.intel.com/products/28031/Intel-Xeon-Processor-E5320-8M-Cache-1_86-GHz-1066-MHz-FSB

[12] Advanced Micro Devices, Inc.(2013). AMD Opteron™ Processor Solutions. [Online]. Available: http://products.amd.com/en-us/OpteronCPUDetail.aspx?id=553&f1=Six-Core+AMD+Opteron%E2%84%A2&f2=&f3=Yes&f4=&f5=512&f6=Socket+F+(1207)&f7=&f8=45nm+SOI&f9=&f10=4800&f11=6&

[13] Intel Corporation. Intel® Atom™ Processor N270. [Online]. Available: http://ark.intel.com/products/36331/Intel-Atom-Processor-N270-512K-Cache-1_60-GHz-533-MHz-FSB

[14] Gennadiy Shvets. (2011, Mar.). CPU Frequency. [Online]. Available: http://www.cpu-world.com/Glossary/C/CPU_Frequency.html

[15] QuinStreet Inc. (2013). Cache. [Online]. Available: http://www.webopedia.com/TERM/C/cache.html

[16] L. Hennessy John, A. Patterson David, *Computer Architecture A Quantitive Approach*, 4th ed. San Fransisco, CA: Morgan Kauffman Publishers, 2007.

[17] Ulrich Drepper. (2007, Oct.). Memory part 2: CPU caches. [Online]. Available: http://lwn.net/Articles/252125/

[18] Matt Smith. (2011, Jul.). What Is Thermal Design Power? [Online]. Available: http://www.makeuseof.com/tag/thermal-design-power-technology-explained/

[19] (2013, Feb.). PERF_EVENT_OPEN. [Online]. Available: http://web.eece.maine.edu/~vweaver/projects/perf_events/perf_event_open.html

[20] *ThreadSpotter™ Manual*, Rogue Wave Software, Inc., 2011.

[21] L. Henning John, "Performance Counters and Development of SPEC CPU2006", Vol. 35, No. 1, March 2007.