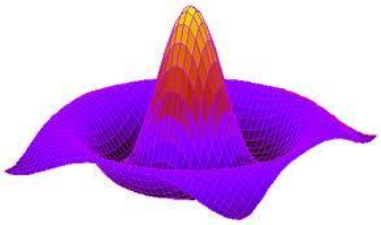
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

Μελέτη λογισμικών παράλληλης επεξεργασίας - Ανάπτυξη παράλληλων αλγορίθμων.



Πτυχιακή εργασία του

ΙΩΑΝΝΗ ΚΟΤΑΜΑΝΙΔΗ (1879)

Επιβλέπων: Δρ.Δημήτριος Ν. Βαρσάμης Καθηγητής Εφαρμογών

Απρίλιος 2013

Υπεύθυνη δήλωση

Υπεύθυνη Δήλωση: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Πληροφορικής & Επικοινωνιών του Τ.Ε.Ι. Σερρών.

ΠΕΡΙΛΗΨΗ

Παράλληλη επεξεργασία είναι μια μορφή υπολογισμού στην οποία πολλοί υπολογισμοί διεξάγονται ταυτόχρονα ,που λειτουργούν με την αρχή ότι τα μεγάλα προβλήματα μπορούν συχνά να διαιρεθούν σε μικρότερα, τα οποία στη συνέχεια επιλύονται ταυτόχρονα. Ο παραλληλισμός έχει χρησιμοποιηθεί εδώ και πολλά χρόνια, κυρίως σε υπολογιστές υψηλών επιδόσεων(high-performance computing) , αλλά έχει αυξηθεί τα τελευταία χρόνια λόγω των φυσικών περιορισμών πρόληψη της κλιμάκωσης συχνότητας . Δεδομένου ότι η κατανάλωση ενέργειας (και, κατά συνέπεια, την παραγωγή θερμότητας) από τους υπολογιστές έχει γίνει μια ανησυχία σε τα τελευταία χρόνια, ο παράλληλος προγραμματισμός έχει γίνει το κυρίαρχο παράδειγμα στην αρχιτεκτονική υπολογιστών , κυρίως με τη μορφή των πολυπύρηνων επεξεργαστών.

Στην εργασία αυτή υπάρχει ως πρώτο κεφάλαιο το θεωρητικό υπόβαθρο της πτυχιακής το όποιο εμβαθύνει και αναλύει την έννοια της παράλληλης επεξεργασίας ,τα ειδή της ,σε ποιους τομείς χρησιμοποιείται καθώς και τα πλεονεκτήματα της . Έπειτα παρουσιάζονται και αναλύονται λογισμικά παράλληλης επεξεργασίας ενώ στην τρίτη ενότητα παρουσιάζεται η εργαστηριακή υλοποίηση της πτυχιακής στο όποιο δείχνω αναλυτικά την δουλεία μου όσον αφορά την παράλληλη επεξεργασία, τα λογισμικά και τους παράλληλους αλγορίθμους. Στην τέταρτη ενότητα υπάρχει ένας αναλυτικός οδηγός ο όποιος δείχνει πως μπορεί κάποιος να κάνει εγκατάσταση το λειτουργικό σύστημα στο όποιο δούλεψα λογισμικό το όποιο χρησιμοποίησα καθώς και τα περαιτέρω πακέτα που χρειάζεται και τέλος να εκτελέσει παράλληλους αλγορίθμους .Τέλος στο πέμπτο κεφάλαιο είναι τα συμπεράσματα τα όποια πρόεκυψαν

Πίνακας περιεχομένων

Υπεύθυνη δήλωση

ΚΕΦΑΛΑΙΟ 1

Θεωρητική Ανάλυση

1.1 Παράλληλη και σειριακή επεξεργασία

Μετά από 40 χρόνια πλήρους ενασχόλησης με το σειριακό προγραμματισμό, η επιστήμη των υπολογιστών άρχισε να αναγνωρίζει τη σημασία του παράλληλου προγραμματισμού. Παρόλο που τα προγράμματα είναι σειριακά (μια ακολουθία πράξεων), τρομακτικές αυξήσεις στην υπολογιστική ισχύ επιτεύχθηκαν κάνοντας την εκτέλεση μιας ακολουθίας εντολών ολοένα και πιο γρήγορη. Το 1945 ο πρώτος ηλεκτρονικός επεξεργαστής στον υπολογιστή ENIAC μπορούσε να εκτελέσει 1000 αριθμητικές πράξεις ανά δευτερόλεπτο. Οι συμβατικοί υπολογιστές RISC (επεξεργαστές μειωμένου σετ εντολών) μπορούν σήμερα να εκτελέσουν 10 εκατομμύρια αριθμητικών πράξεων ανά δευτερόλεπτο. Αυτοί οι επεξεργαστές είναι κατά βάση ακόμα σειριακοί αλλά η ακολουθία είναι τώρα 10^4 πιο γρήγορη απ' ότι ήταν πριν 40 χρόνια. Η αύξηση της ταχύτητας των επεξεργαστών αναμένεται να συνεχιστεί, αλλά μάλλον με ρυθμούς μικρότερους σε σχέση με το παρελθόν.

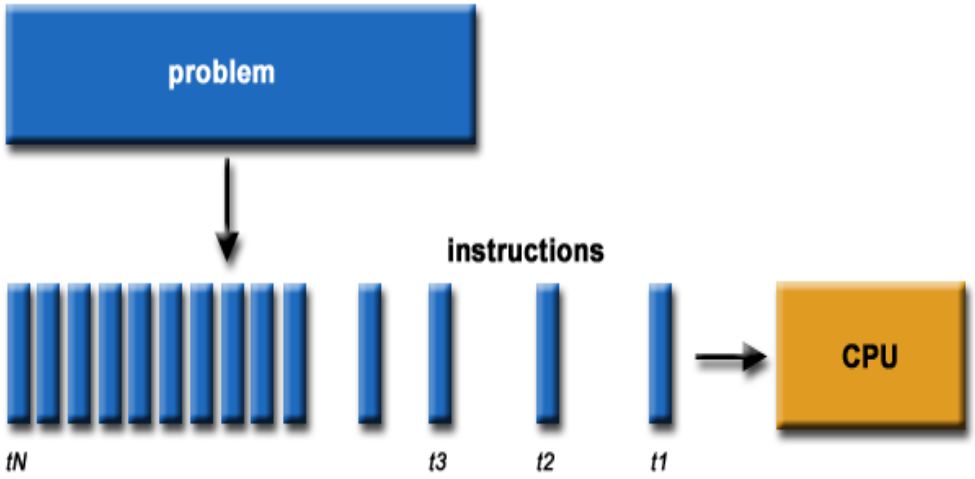
Παρόλη την συνεχώς αυξανόμενη υπολογιστική ισχύ των επεξεργαστών VLSI, υπάρχει μια ευρεία περιοχή σημαντικών επιστημονικών προβλημάτων που απαιτούν πολύ μεγαλύτερη υπολογιστική ταχύτητα. Μάλιστα σε ορισμένες περιπτώσεις η διαφορά της επιπλέον υπολογιστικής ισχύος που απαιτείται, αγγίζει τα όρια αρκετά μεγάλων τάξεων μεγέθους

1.1.1 Πως εκτελούνται σειριακοί αλγόριθμοι

Παραδοσιακά, το λογισμικό που γράφεται με στόχο την ακολουθιακή(σειριακή) εκτέλεση ακόλουθη κάποιους βασικούς κανόνες

- Εκτελείται σε έναν υπολογιστή που έχει μια ενιαία Κεντρική Μονάδα Επεξεργασίας(CPU)
- Το πρόβλημα (problem) επιλύεται με την εκτέλεση μιας σειράς (ακολουθίας) εντολών (instructions).
- Οι εντολές προυποθέτουν την ολοκλήρωση των προηγούμενων εντολών, και τελος

- Κάθε χρονική στιγμή (*t1, t2, t3, ..., tN*) εκτελείται μια εντολή.

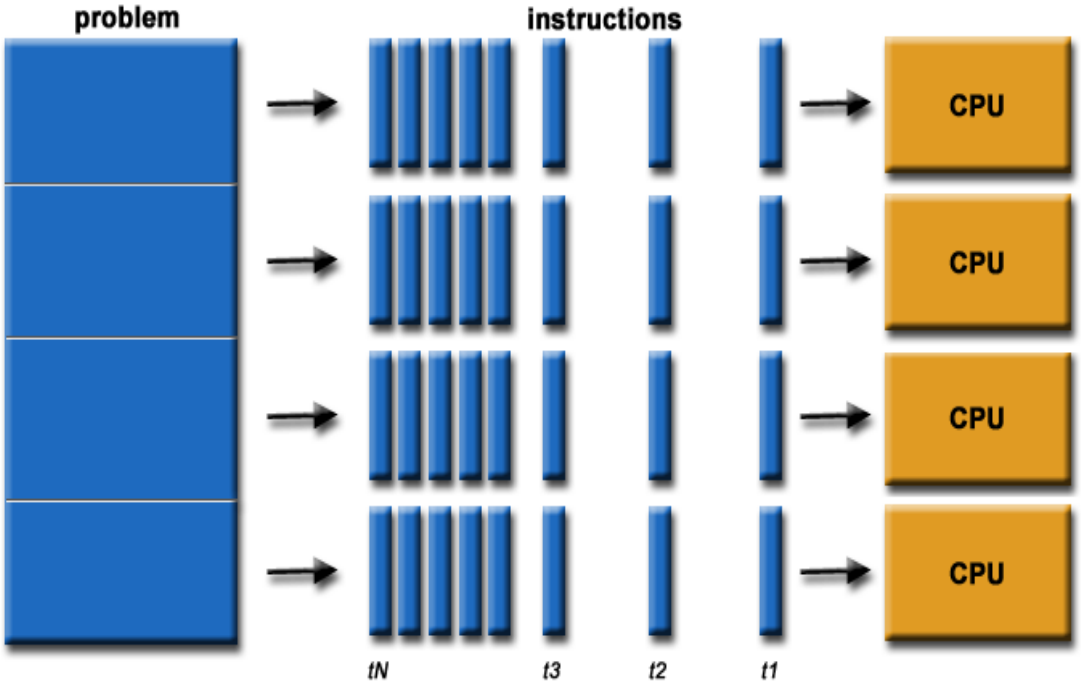


Εικόνα 1 Σειριακή επεξεργασία

1.1.2 Πως εκτελούνται παράλληλοι αλγόριθμοι

Σε αντίθεση με τον σειριακό υπολογισμό ,το λογισμικό που έχει γραφτεί για παράλληλη επεξεργασία μπορεί και κάνει ταυτόχρονη χρήση πολλών πόρων για να υπολογίζουν και τελικά να λύσουν ένα υπολογιστικό πρόβλημα. Οι βασικοί κανόνες που έχει ένα τέτοιο λογισμικό είναι οι εξής:

- Μπορεί να εκτελεστεί χρησιμοποιώντας πολλές CPU
- Ένα πρόβλημα είναι σπασμένο σε διακριτά μέρη που μπορούν να λυθούν ταυτόχρονα
- Κάθε διακριτό μέρος του προβλήματος αναλύεται περεταίρω σε μια σειρά από οδηγίες
- Εντολες από κάθε μέρος του προβληματος μπορουν να εκτελεστουν ταυτόχρονα σε διαφορετικές CPUs



Εικόνα 2 Παράλληλη επεξεργασία

Οι υπολογιστικοί πόροι που θα μπορούσαμε να εκτελέσουμε ένα λογισμικό παράλληλης επεξεργασίας είναι:

- Ένας υπολογιστής με πολλές CPUs (ή μια CPU με πολλούς πυρήνες -cores) ή μια CPU με πολλές ALUs. Η σύνδεση μεταξύ των CPUS ή ALUs καθώς και με τη κεντρική μνήμη επιτυγχάνεται μέσω είτε εξειδικευμένου δικτύου διασύνδεσης ή τυπικού διαύλου.
- Πολλοί υπολογιστές συνδεδεμένοι είτε σε συστοιχίες μέσω εξειδικευμένης διασύνδεσης υψηλής απόδοσης και κοινά προσπελάσιμους δίσκους, ή σε τυπικό τοπικό δίκτυο ή ακόμη και μέσω διαδικτύου.
- Συνδυασμός των δύο παραπάνω καταστάσεων (πχ υπολογιστές πολλαπλών πυρήνων σε τοπικό δίκτυο)

Ένα υπολογιστικό πρόβλημα συνήθως παρουσιάζει τα παρακάτω χαρακτηριστικά:

- Μπορεί να τεμαχιστεί σε διακριτά τμήματα - υποπροβλήματα- που μπορούν να επιλυθούν - εκτελεστούν - ταυτόχρονα.

- Η επίλυση του προβλήματος με τη χρήση πολλαπλών υπολογιστικών πόρων είναι αποδοτικότερη από ότι σε απλό υπολογιστή (αύξηση της ταχύτητας εκτέλεσης-speedup, αύξηση του μεγέθους του προβλήματος-scaleup).

1.2 Τομείς που χρησιμοποιείται η παράλληλη επεξεργασία

Παρόλη την συνεχώς αυξανόμενη υπολογιστική ισχύ των επεξεργαστών VLSI, υπάρχει μια ευρεία περιοχή σημαντικών επιστημονικών προβλημάτων που απαιτούν πολύ μεγαλύτερη υπολογιστική ταχύτητα. Μάλιστα σε ορισμένες περιπτώσεις η διαφορά της επιπλέον υπολογιστικής ισχύος που απαιτείται, αγγίζει τα όρια αρκετά μεγάλων τάξεων μεγέθους. Παραδείγματα τέτοιων γνωστικών περιοχών είναι η αεροδυναμική, η πρόγνωση καιρού, η μοριακή φυσική, η επεξεργασία εικόνων στα γραφικά υπολογιστών, η προσομοίωση χημικών αντιδράσεων και η προσομοίωση νευρωνικών δικτύων στη βιολογία. Ένα τυπικό παράδειγμα είναι οι εξισώσεις Navier-Stokes, που χρησιμοποιούνται στο σχεδιασμό αεροσκαφών. Η επίλυση τους τώρα καθίσταται δυνατή μόνο με τους πιο δυνατούς Η/Υ και η λύση που επιτυγχάνεται είναι προσεγγιστική. Αυτή τη στιγμή υπολογιστές από 10 φορές έως και 100 φορές πιο γρήγοροι απαιτούνται αυτή τη στιγμή στα παραπάνω επιστημονικά επίπεδα.

Στη δημιουργία κινούμενων εικόνων (animation), η τεχνική της ‘’ανίχνευσης ακτινών’’ (ray tracing), που παράγει τις καλύτερες ποιοτικά εικόνες, συνήθως δεν χρησιμοποιείται γιατί οι απαιτήσεις σε υπολογισμούς είναι μεγάλες ακόμα και για τα πιο γρήγορα συστήματα. Άλλο ένα παράδειγμα αυξημένης υπολογιστικής ισχύος μπορεί να βρεθεί στον κλάδο της μοριακής φυσικής. Οι σύγχρονες θεωρίες σχετικά με τα σωματίδια του πυρήνα περιέχουν εξισώσεις που προβλέπουν την ύπαρξη ενός συγκεκριμένου υποσωματιδίου που ονομάζεται ‘’κουάρκ’. Η επίλυση τέτοιων εξισώσεων απαιτεί τόσους πολλούς υπολογισμούς που ακόμα και ο πιο γρήγορος υπολογιστής διαθέσιμος σήμερα χρειάζεται ένα ολόκληρο χρόνο.

Γενικά η παράλληλη επεξεργασία αποτελεί μια εξέλιξη της ακολουθιακής επεξεργασίας που προσπαθεί να μιμηθεί αυτό που σχεδόν πάντα συμβαίνει στο φυσικό κόσμο: πολλά σύνθετα, αλληλοσχετιζόμενα γεγονότα συμβαίνουν ταυτόχρονα, το καθένα με τη δική του εσωτερική ακολουθία συμβάντων, και με κάποιας μορφής επικοινωνία ή συγχρονισμό μεταξύ των γεγονότων. Μερικά παραδείγματα:

- Τροχιές πλανητών, αστεριών ή γαλαξιών

- Μετεωρολογικά φαινόμενα, θαλάσσια ρεύματα
- Κίνηση τεκτονικών πλακών
- Κυκλοφορία οχημάτων σε μια πόλη
- Γραμμή παραγωγής σε ένα εργοστάσιο
- Ημερήσια δραστηριότητα σε μια επιχείρηση
- Οικοδόμηση ενός τεχνικού έργου
- Εξυπηρέτηση μιας παραγγελίας σε ένα κατάστημα.

Επίσης η παράλληλη επεξεργασία θεωρείται ως εξειδικευμένη τεχνική με βασικό σκοπό την αριθμητική επίλυση σύνθετων προβλημάτων επιστήμης και μηχανικής, όπως:

- Πρόβλεψη του καιρού, μακροπρόθεσμα κλιματικά μοντέλα
- Σύνθετες χημικές και πυρηνικές αντιδράσεις
- Αστρονομία και υπολογιστική φυσική
- Γονιδιωματική, υπολογιστική βιολογία
- Γεωλογία, σεισμολογία
- Υπολογιστική ρευστοδυναμική
- Μηχανική - από νανοεξαρτήματα έως αεροπλάνα
- Σύνθετα ηλεκτρικά και ηλεκτρονικά κυκλώματα
- Κατασκευαστικές διεργασίες
- Τεχνητή όραση, επεξεργασία φυσικής γλώσσας.

Σήμερα, οι επιχειρηματικές και οικονομικές εφαρμογές αποδεικνύονται εξ'ισου σημαντικές ή και σημαντικότερες από τις εφαρμογές επιστήμης και μηχανικής. Αυτές οι εφαρμογές απαιτούν την σύνθετες επεξεργασίες μεγάλων όγκων δεδομένων. Ορισμένα παραδείγματα:

- Παράλληλες βάσεις δεδομένων, εξόρυξη δεδομένων
- Αναζήτηση κοιτασμάτων πετρελαίου και φυσικού αερίου
- Μηχανές αναζήτησης ιστού, διακομιστές και υπηρεσίες ιστού
- Ιατρική διάγνωση με τη βοήθεια υπολογιστή
- Χρηματοοικονομική και τραπεζική πραγματικού χρόνου
- Γραφικά και εικονική πραγματικότητα
- Video, παιγνίδια, ψηφιακά μέσα

- Συστήματα ειδικού σκοπού, πραγματικού χρόνου
- Συνεργατικά περιβάλλοντα

1.3 Είδη παραλληλισμού

Στα σύγχρονα υπολογιστικά συστήματα συναντούμε πολλαπλά επίπεδα παραλληλισμού. Όσο "χαμηλότερο" είναι το επίπεδο προσέγγισης τόσο τα υπολογιστικά συστήματα είναι πιο **στενά συνδεδεμένα (tightly coupled)** ο δε παραλληλισμός που εφαρμόζεται είναι πιο **λεπτομερής (fine-grain)**. Οι υπολογιστικές μονάδες είναι απλές και πολλές, ενώ η επικοινωνία ταχύτατη. Αντίστοιχα, στα "υψηλά" επίπεδα τα συστήματα είναι **χαλαρά συνδεδεμένα (loosely coupled)** και ο παραλληλισμός **αδρομερής (coarse-grain)**. Οι υπολογιστικές μονάδες είναι ισχυρές και σχετικά λίγες ενώ η επικοινωνία σχετικά αργή.

- **Επίπεδο Δεδομένων** (Bit Level Parallelism): ταυτόχρονη εκτέλεση πράξεων σε ομοειδή δεδομένα, σε ένα επεξεργαστή ή και σε συνεπεξεργαστές ειδικού σκοπού (πχ bit-level παραλληλισμός, λειτουργίες επεξεργασίας γραφικών, FPGAs, ASICs).

- **Επίπεδο Εντολής** (Instruction Level Parallelism, ILP): ταυτόχρονη εκτέλεση εντολών ενός προγράμματο (κρυφή μνήμη, προ-προσκόμιση, διοχέτευση εντολών και πράξεων, υπερβαθμωτή εκτέλεση, πρόβλεψη διακλάδωσης)

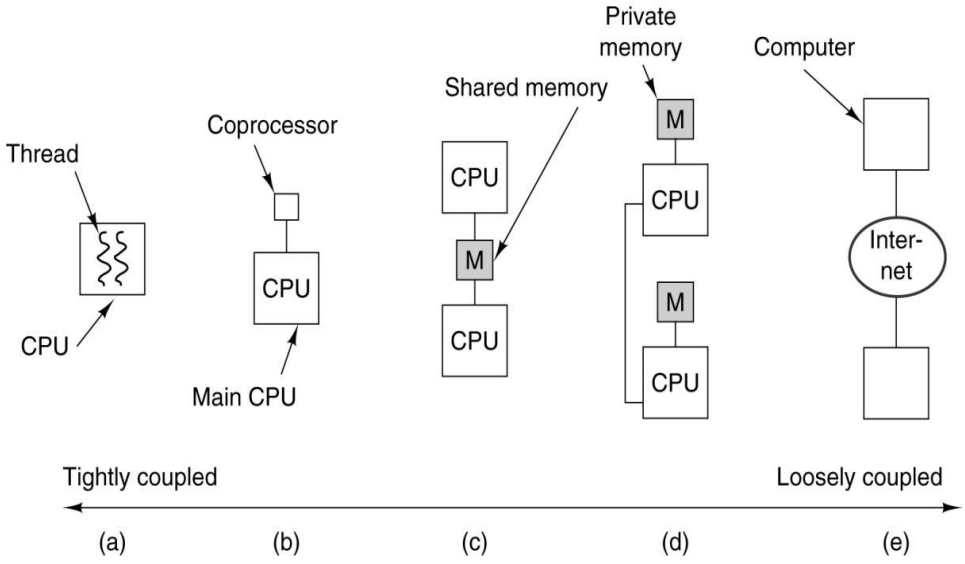
- **Επίπεδο Λογικού Πυρήνα** (Multithreading): ταυτόχρονη εκτέλεση πολλαπλών νημάτων (ελαφρών διεργασιών) σε ένα επεξεργαστή (a).

- **Επίπεδο Φυσικού Πυρήνα** (Multicore): ενσωμάτωση στο ίδιο ολοκληρωμένο κύκλωμα περισσότερων της μιας πλήρων CPU οι οποίες συνδέονται σε εσωτερικό δίαυλο και μοιράζονται υψηλότερα επίπεδα κρυφής μνήμης (c).

- **Επίπεδο Λειτουργιών** (Multitasking): ταυτόχρονη εκτέλεση λειτουργιών (διεργασιών) σε ένα υπολογιστή (πχ χρήση co-processors, DMA, I/O, κάρτες γραφικών) υπό τον έλεγχο του λειτουργικού συστήματος (b).

- **Επίπεδο Υπολογιστή/Υπολογισμού** (Multiprocessing, Multicomputing): ταυτόχρονη εκτέλεση ενός ή διαφορετικών προγραμμάτων σε πολλαπλούς επεξεργαστές.

- **Επίπεδο Δικτύου και Διαδικτύου** (Network / Distributed / Computing): ταυτόχρονη εκτέλεση διαφορετικών -συνήθως- προγραμμάτων σε πολύ χαλαρά συνδεδεμένους υπολογιστές. Εδώ η ταχύτητα δεν είναι το ζητούμενο όσο η κατανομή πόρων δεδομένων και εργασιών και η μεγάλη διακπεραιωτική ικανότητα.



Εικόνα 3

Συγγενικές έννοιες: ***Ταυτοχρονη (Concurrent), Παράλληλη (Parallel), Κατανεμημένη (Distributed) Επεξεργασία***.

Η **Ταυτοχρονη *(*Concurrent*)*:**

Επεξεργασία δίνει έμφαση στο διαμοιρασμό ενός ή περισσότερων πόρων μεταξύ πολλών ανεξάρτητων εργασιών που εκτελούνται ταυτόχρονα. Οι εργασίες είναι συνήθως σχετικά λίγες και ανόμοιες, και η επικοινωνία μεταξύ τους είναι μη-προκαθορισμένη και βασίζεται σε συμβάντα (events). Χαρακτηριστικό πεδίο η ανάπτυξη λειτουργικών συστημάτων και πολυνηματικών εφαρμογών σε ένα υπολογιστή.

Η **Παράλληλη *(*Parallel*)*:**

Επεξεργασία δίνει έμφαση στην ταυτόχρονη εκτέλεση πολλών αλληλοεξαρτώμενων εργασιών σε πολλούς όμοιους πόρους για την επίλυση ενός προβλήματος. Οι εργασίες είναι συνήθως σχετικά πολλές, σχετικά όμοιες μεταξύ τους, και η επικοινωνία ακολουθεί συνήθως κάποιο προκαθορισμένο ή κανονικό πρότυπο. Χαρακτηριστικό πεδίο η ανάπτυξη επιστημονικών και βιομηχανικών εφαρμογών για την επεξεργασία πολλών δεδομένων / εκτέλεση πολλών υπολογισμών.

Η **Κατανεμημένη *(*Distributed*)*:**

Επεξεργασία δίνει έμφαση στην επικοινωνία πολλών ανεξάρτητων και ανόμοιων διεργασιών που εκτελούνται ταυτόχρονα σε ξεχωριστά σύνολα πόρων και όπου η επικοινωνία είναι μη-ντετερμινιστική, ίσως αναξιόπιστη και προκαλεί σημαντική επιβάρυνση. Χαρακτηριστικό πεδίο η ανάπτυξη διαδικτυακών εφαρμογών μεγάλης κλίμακας, με χρήση πολλαπλών εξειδικευμένων διακομιστών ή και ομότιμης τεχνολογίας.

Ο παραλληλισμός επηρεάζει το σύνολο ενός υπολογιστικού σύστήματος: έχουμε παράλληλες αρχιτεκτονικές, λειτουργικά συστήματα / συστήματα εκτέλεσης που υποστηρίζουν τον παραλληλισμό, παράλληλες γλώσσες / βιβλιοθήκες προγραμματισμού, παράλληλους αλγορίθμους και μεθοδολογίες ανάπτυξης λογισμικού και εφαρμογών. Όμως ενώ τα ακολουθιακά αντίστοιχα είναι σχετικά καλά ορισμένα, οι παράλληλες εκδοχές επιτρέπουν αρκετούς βαθμούς ελευθερίας και συνακόλουθα εναλλακτικές προσεγγίσεις. Η υιοθέτηση ή μη μιας προσέγγισης δεν είναι μόνο θέμα ορθότητας αλλά και τεχνολογικής συγκυρίας.

Μια σημείωση σχετικά με το επίπεδο προσέγγισης της παράλληλης επεξεργασίας. Στο φυσικό κόσμο συνήθως διακρίνουμε τρία επίπεδα προσέγγισης ενός φαινομένου. Για παράδειγμα, στη Φυσική, έχουμε το μικρο-επίπεδο της Κβαντομηχανικής, το μεσο-επίπεδο, της κλασσικής Νευτώνιας Φυσικής, και το μακρο-επίπεδο της Θεωρίας της Σχετικότητας. Αντίστοιχα, η ανθρώπινη συμπεριφορά μπορεί να προσεγγιστεί σε μικρο-επίπεδο, δηλαδή σε επίπεδο βιολογικής και νευροφυσιλογικής λειτουργίας ενός προσώπου, σε μέσο-επίπεδο, δηλαδή σε επίπεδο συνειδητής καθημερινής συμπεριφοράς του ίδιου προσώπου, και σε μάκρο-επίπεδο, δηλαδή σε επίπεδο συμπεριφοράς κοινωνικών ομάδων ή και ολόκληρων πληθυσμών ή ειδών σε ιστορικό χρόνο. Συνήθως, η συμβατική κατανόηση της καθημερινότητάς μας και του κόσμου εστιάζεται στο προσωπικό μας μέσο-επίπεδο. Σε αυτό το επίπεδο οι σκέψεις και οι ενέργειές μας, όπως επίσης και η αλληλεπίδραση με το περιβάλλον είναι κατα βάση ακολουθιακή: σκεφτόμαστε ένα πράγμα κάθε φορά, εκτελούμε μια εργασία κάθε στιγμή. Η ακολουθιακή προσέγγιση είναι η πλέον εύλογη. Όμως, αν σκεφτούμε τον εαυτό μας σαν ένα οργανισμό με δισεκατομμύρια κύτταρα που συντονίζεται από έναν εγκέφαλο με δισεκατομμύρια νευρώνες οι οποίοι συνδέονται μεταξύ τους με δεκάδες χιλιάδες συνάψεις, και όλα αυτά λειτουργούν Παράλληληλα και Κατανεμημένα (Parallel Distributed Processing) για να επιτύχουν αυτό που ονομάζουμε συνείδηση, τότε κατανοούμε οτι υπάρχει ένα καλυμμένος παραλληλισμός στο μικρο-επίπεδο που είναι

απολύτως απαραίτητος για την επιτυχημένη ακολουθιακή λειτουργία του μέσου επιπέδου. Αντίστοιχα, αν σκεφτούμε τον εαυτό μας ως ένα μέρος ενός κοινωνικού συνόλου τότε η ακολουθιακή μας δραστηριότητα συντονίζεται με τις δραστηριότητες άλλων ανθρώπων για τη παραγωγή ενός ενιαίου αποτελέσματος. Αυτή η συνεργασία μπορεί να είναι συνειδητή ή μη, τοπική ή απομακρυσμένη, χρονικά περιορισμένη ή όχι, κεντρικά συντονισμένη ή όχι.

1.4 Πλεονεκτήματα παράλληλης επεξεργασίας

Οι κύριοι λόγοι για τους οποίους χρησιμοποιούμε την παράλληλη επεξεργασία είναι οι εξής:

- **Εξοικονομούμε χρόνο και χρήματα**

Τα παράλληλα υπολογιστικά συστήματα μπορούν να κατασκευαστούν από φτηνά συστατικά βασικών προϊόντων , αν συνδυάσουμε αυτό με την θεωρία ότι ρίχνοντας περισσότερους πόρους σε μια εργασία θα συντόμευση τον χρόνο της για να ολοκληρωθεί με πιθανή μείωση του κόστους τότε ως αποτέλεσμα έχουμε την εξοικονόμηση χρόνου και χρημάτων ταυτόχρονα.

- **Επίλυση μεγαλύτερων προβλημάτων:**

Πολλά προβλήματα είναι τόσο μεγάλα και πολύπλοκα που καθιστά ανέφικτη ή αδύνατη την επίλυσή τους σε έναν μόνο υπολογιστή, ειδικά δεδομένης της περιορισμένης μνήμη του υπολογιστή. Για παράδειγμα
μηχανές αναζήτησης στο Web καθως και την επεξεργασία δεδομένων

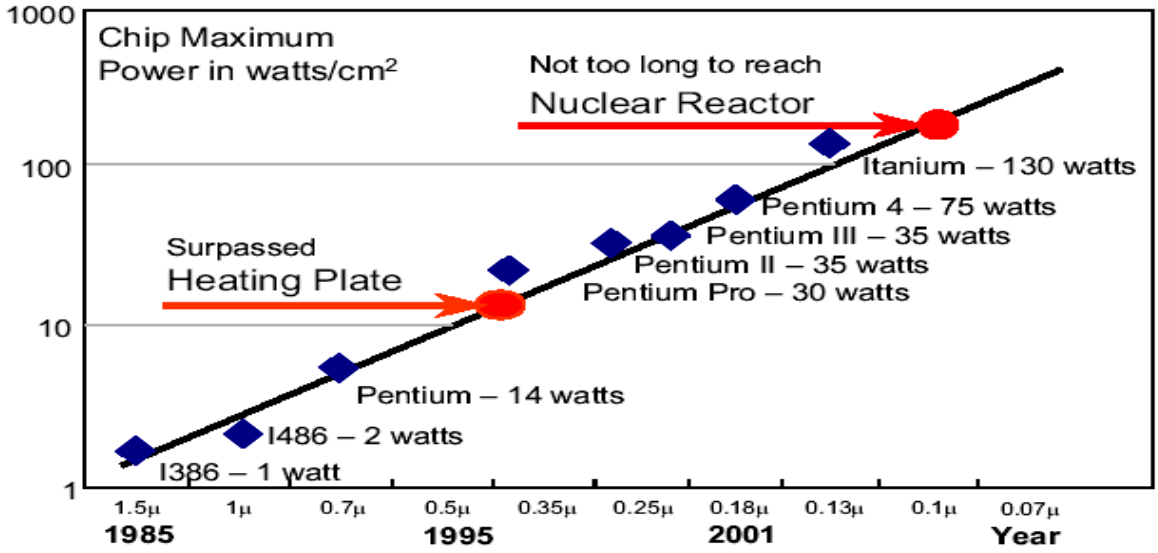
- **Παροχή συγχρονισμού :**

Ένας μοναδικός υπολογιστικός πόρος μπορεί να κάνει μόνο ένα πράγμα τη φορά. Πολλοί υπολογιστικοί πόροι μπορούν να κάνουν πολλά πράγματα ταυτόχρονα. Για παράδειγμα, το Acces Grid παρέχει ένα παγκόσμιο δίκτυο συνεργασίας, όπου οι άνθρωποι από όλο τον κόσμο να συναντηθούν και να διεξάγουν μια εργασία «εικονικά»

- **Όρια στον σειριακό υπολογισμό:**

Φυσικοί και πρακτικοί λόγοι θέτουν σημαντικούς περιορισμούς στην κατασκευή ταχύτερων υπολογιστών:

- **Ταχύτητα μετάδοσης** - η ταχύτητα ενός ακολουθιακού υπολογιστή εξαρτάται ευθέως από τη ταχύτητα μετάδοσης των δεδομένων στο υλικό. Απόλυτα όρια θέτουν οι ταχύτητα διάδοσης του φωτός (30 cm/nanosecond) και η ταχύτητα διάδοσης σήματος σε χαλκό (9 cm/nanosecond). Η αύξηση της ταχύτητας του επεξεργαστή απαιτεί διαρκώς πυκνότερη τοποθέτηση των κυκλωμάτων του.
- **Όρια στη σμίκρυνση** - η τεχνολογία ολοκληρωμένων κυκλωμάτων επιτρέπει διαρκώς μεγαλύτερους αριθμούς transistors ανά chip. Όμως, ακόμη και σε μοριακό ή ατομικό επίπεδο ολοκλήρωσης, υπάρχει ένα φυσικό όριο στη πυκνότητα των κυκλωμάτων και στην ενεργειακής κατανάλωση (watt/cm^2).
- **Ενεργειακοί περιορισμοί** - η κατανάλωση ισχύος των επεξεργαστών βαίνει διαρκώς αυξανόμενη αναλογικά με τη συχνότητα του ρολογιού ενός επεξεργαστή, πλησιάζοντας σε δύσκολα διαχειρίσιμα όρια. τόσο για τροφοδοσία όσο και για ψύξη (Power wall).

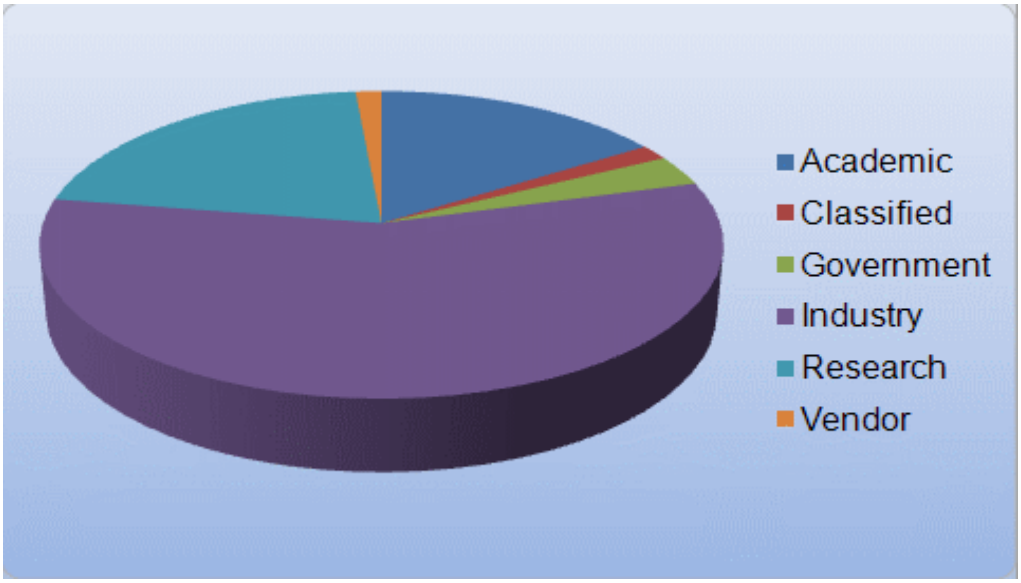


Στατιστικό διάγραμμα 1

1.5 Στατιστικά στοιχειά σχετικά με την παράλληλη επεξεργασία

Διαπιστώνουμε οτι πλέον οι 'παραδοσιακοί' τομείς είναι σχεδόν μειοψηφικοί, ενώ νέοι τομείς εμφανίζονται. Είναι πλέον φανερό οτι τόσο σε επίπεδο διακομιστών και data centers αλλά ακόμη και σε επίπεδο προσωπικών υπολογιστών η παράλληλη επεξεργασία κερδίζει συνεχώς έδαφος. Στο Top500.org δίνονται πολλές πληροφορίες σχετικά με τα ισχυρότερα συστήματα στο κόσμο, την αρχιτεκτονική τους, τις εφαρμογές που εκτελούν κλπ.

Τα παρακάτω διαγράμματα μας παρέχουν στατιστικά στοιχειά σχετικά με την παράλληλη επεξεργασία υπολογιστών



Στατιστικό διάγραμμα 2

1.6 Αρχιτεκτονική παράλληλων συστημάτων

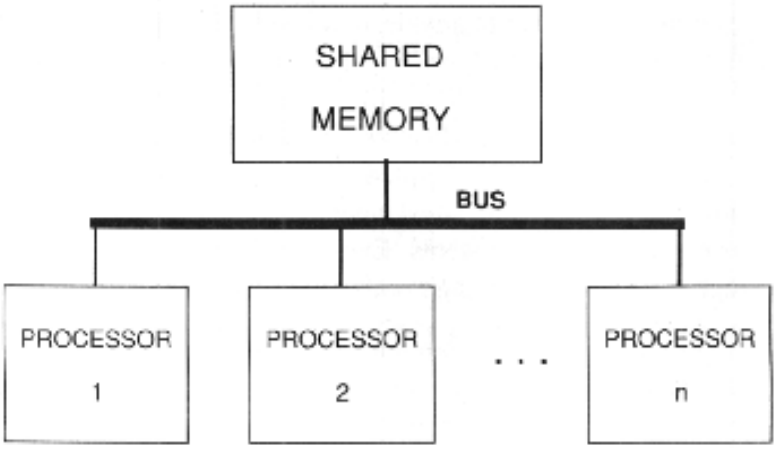
1.6.1 Κοινόχρηστης μνήμης

Η βασική ιδέα πίσω από ένα παράλληλο σύστημα είναι απλά να υπάρχει πάνω από ένας επεξεργαστής στον ίδιο υπολογιστή. Τα παράλληλα συστήματα μπορεί να έχουν πολύ λίγους επεξεργαστές για παράδειγμα 10 ή πάρα πολλούς π.χ. 50000. Το κλειδί για να χαρακτηριστεί ένα σύστημα σαν παράλληλο, είναι ότι όλοι οι επεξεργαστές του είναι ικανοί να λειτουργούν ταυτόχρονα. Αν ο κάθε επεξεργαστής μπορεί να εκτελέσει 1 εκατομμύριο

λειτουργίες ανά δευτερόλεπτο, τότε 10 επεξεργαστές μπορούν να εκτελέσουν 10 εκατομμύρια λειτουργίες ανά δευτερόλεπτο, οι 100 επεξεργαστές μπορούν να εκτελέσουν 100 εκατομμύρια λειτουργίες ανά δευτερόλεπτο κ.λ.π.

Η χρήση των πολλαπλών παράλληλων επεξεργαστών στο ίδιο υπολογιστικό σύστημα εισάγει μερικές επιπρόσθετες απαιτήσεις στην αρχιτεκτονική του συστήματος. Για να μπορέσουν πολλοί επεξεργαστές να δουλέψουν ταυτόχρονα πάνω στο ίδιο υπολογιστικό πρόβλημα, πρέπει να είναι ικανοί να μοιράζονται δεδομένα και να επικοινωνούν μεταξύ τους. Υπάρχουν, προς το παρόν, δύο βασικές αρχιτεκτονικές προσεγγίσεις για την εκπλήρωση αυτής της απαίτησης: η διαμοιραζόμενη μνήμη και το πέρασμα μηνυμάτων. Στα συστήματα κοινοχρηστης μνήμης, που συχνά ονομάζονται και συστήματα πολυεπεξεργασίας όλοι οι μεμονωμένοι επεξεργαστές έχουν να προσπελάσουν μία κοινή μνήμη, και τους επιτρέπεται η κοινή χρήση των ποικίλων, τιμών δεδομένων και δομών δεδομένων που είναι αποθηκευμένες στη μνήμη. Στα συστήματα αρχιτεκτονικής περάσματος μηνυμάτων, που συχνά ονομάζονται και παράλληλα συστήματα κατανεμημένης μνήμης, ο κάθε επεξεργαστής έχει τη δική του τοπική μνήμη, και οι επεξεργαστές μοιράζονται δεδομένα μεταξύ τους, με το μηχανισμό περάσματος μηνυμάτων μέσω ενός είδους δικτύου επικοινωνίας επεξεργαστών. Και οι δύο αυτοί τύποι αρχιτεκτονικής παράλληλων συστημάτων έχουν σίγουρα πλεονεκτήματα και μειονεκτήματα.

Στην εικόνα 4 βλέπουμε ένα διάγραμμα της οργάνωσης ενός συστήματος κοινόχρηστης μνήμης. Όλοι οι επεξεργαστές μπορούν να υπολογίσουν παράλληλα, και ο καθένας είναι δυνατόν να προσπελάσει την κεντρική διαμοιραζόμενη μνήμη μέσω μιας δομής κοινού διαύλου. Ο κοινός δίαυλος είναι υπεύθυνος για τη διαιτησία ανάμεσα στις ταυτόχρονες απαιτήσεις μνήμης των διαφόρων επεξεργαστών και για την εξασφάλιση της δίκαιας εξυπηρέτησης όλων των επεξεργαστών με την ελάχιστη καθυστέρηση προσπέλασης. Η λειτουργία του κάθε επεξεργαστή είναι ίδια με τη λειτουργία που εκτελείται από έναν επεξεργαστή σε ένα συνηθισμένο σειριακό σύστημα. Ο κάθε επεξεργαστής εξακολουθεί να διαβάζει τιμές δεδομένων από τη κοινή μνήμη, να υπολογίζει καινούργιες τιμές, και να τις γράφει πάλι πίσω στη κοινή μνήμη. Αυτή η υπολογιστική λειτουργία εκτελείται από όλους τους επεξεργαστές παράλληλα. Έτσι αν υπάρχουν n επεξεργαστές, τότε το υπολογιστικό σύστημα έχει μέγιστη υπολογιστική ικανότητα n φορές περισσότερη από ένα απλό σύστημα ενός επεξεργαστή.



Εικόνα 4 Οργάνωση Συστήματος Κοινόχρηστης Μνήμης με Κοινό Δίαυλο

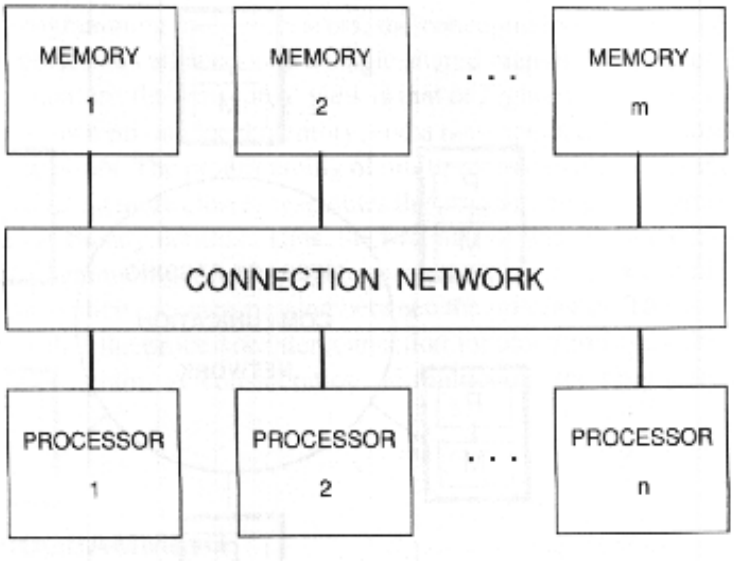
Ο αριθμός των επεξεργαστών που είναι διαθέσιμος σε τέτοια συστήματα σήμερα ποικίλει με μέγιστο από 40 έως 100. Ένα από τα βασικά προβλήματα των συστημάτων κοινόχρηστης μνήμης είναι ο ανταγωνισμός πρόσβασης στη μνήμη ανάμεσα στους επεξεργαστές, ο οποίος αυξάνεται όσο περισσότεροι επεξεργαστές προστίθενται. Όταν πολλοί από τους επεξεργαστές προσπαθούν να προσπελάσουν την κοινή μνήμη μέσα σε σύντομο χρονικό διάστημα, η μνήμη δεν θα είναι ικανή να εξυπηρετήσει όλες τις απαιτήσεις ταυτόχρονα, και μερικοί από τους επεξεργαστές πρέπει να περιμένουν ενώ άλλοι θα εξυπηρετούνται. Στο σχεδιασμό των συστημάτων κοινόχρηστης μνήμης, μεγάλη προσοχή πρέπει να δοθεί στη μείωση της πιθανότητας ανταγωνισμού των τμημάτων μνήμης.

Μια μεγάλη ποικιλία τεχνικών χρησιμοποιείται για να βοηθήσει τη μείωση του ανταγωνισμού πρόσβασης στη μνήμη και να κάνει το σύστημα πιο αποδοτικό. Μια τέτοια τεχνική είναι να επιτρέπεται σε κάθε επεξεργαστή να έχει μία τοπική κρυφή μνήμη, η οποία χρησιμοποιείται για να κρατάει αντίγραφα από τα πιο πρόσφατα χρησιμοποιημένα τμήματα μνήμης. Από τη στιγμή που ο κάθε επεξεργαστής έχει τη δική του τοπική κρυφή μνήμη, τα δεδομένα της μπορούν να προσπελαστούν πολύ γρήγορα χωρίς πιθανότητα ανταγωνισμού. Όμως έτσι εισάγεται το πρόβλημα της συνοχής των τμημάτων κρυφής μνήμης: είναι δυνατό να υπάρχουν πολλαπλά αντίγραφα της ίδιας μεταβλητής σε διαφορετικές τοπικές κρυφές μνήμες, γεγονός που οδηγεί στην πιθανότητα ύπαρξης μη ενημερωμένων τιμών μετά από κάποια ενημέρωση. Μια ποικιλία εξεζητημένων τεχνικών για τη συνοχή της κρυφής μνήμης έχει αναπτυχθεί σε διάφορα συστήματα για να λύσει αυτό το πρόβλημα. Μια τέτοια τεχνική

είναι ο κάθε επεξεργαστής να έχει μια κρυφή μνήμη με έλεγχο (snooping cache) η οποία να παρακολουθεί αδιάκοπα το κοινό δίαυλο και να καθιστά άκυρες τις τιμές μνήμης που έχουν ενημερωθεί (τροποποιηθεί) από άλλους επεξεργαστές.

Μια άλλη τεχνική για τη μείωση του ανταγωνισμού των τμημάτων μνήμης στα συστήματα κοινόχρηστης μνήμης είναι η διαίρεση της κοινής μνήμης σε χωριστά τμήματα, τα οποία μπορούν να προσπελαστούν παράλληλα από διαφορετικούς επεξεργαστές. Τα διαμοιραζόμενα δεδομένα κατανέμονται σε πολλά χωριστά τμήματα μνήμης, έτσι ώστε να μειώνεται η πιθανότητα της ταυτόχρονης ζήτησης του ίδιου τμήματος μνήμης από διαφορετικούς επεξεργαστές. Η εικόνα 5 επεξηγεί την γενική οργάνωση ενός συστήματος με πολλαπλά τμήματα κοινόχρηστης μνήμης. Καθένας από τους n επεξεργαστές μπορεί να προσπελάσει οποιοδήποτε τα m τμήματα μνήμης, μέσω ενός δικτύου επικοινωνίας μνήμης. Αυτό το δίκτυο επικοινωνίας είναι ικανό ως ένα βαθμό για εσωτερικό παραλληλισμό, έτσι ώστε πολλοί επεξεργαστές να μπορούν να προσπελάσουν διαφορετικά τμήματα μνήμης ταυτόχρονα. Το κόστος και η εκτέλεση αυτού του μοντέλου εξαρτάται από τον εσωτερικό σχεδιασμό του δικτύου επικοινωνίας. Κάποια συνηθισμένα μοντέλα σχεδιασμού δικτύων όπως της πεταλούδας (butterfly), της διαπλοκής - εναλλαγής (shuffle-exchange), το ραβδεπαφικό (crossbar) και του omega εξετάζονται με λεπτομέρεια σε επόμενο κεφάλαιο.

Η ύπαρξη πολλαπλών τμημάτων μνήμης μπορεί να βελτιώσει την απόδοση του συστήματος κοινόχρηστης μνήμης επειδή πολλά τμήματα μνήμης είναι δυνατόν να προσπελαστούν παράλληλα. Ο παραλληλισμός στην προσπέλαση μνήμης βοηθάει την αύξηση του παραλληλισμού στη λειτουργία των επεξεργαστών: οι πιθανότητες ανταγωνισμού πρόσβασης στη μνήμη μειώνονται σημαντικά. Είναι σαν να συγκρίνεις μία τράπεζα που έχει πολλά ταμεία με μία άλλη που έχει μόνο ένα. Ο αυξημένος αριθμός ταμείων ότι μειώνει την πιθανότητα αναμονής του πελάτη. Με τον ίδιο τρόπο ο αυξημένος αριθμός τμημάτων μνήμης μειώνει την πιθανότητα ο επεξεργαστής να πρέπει να περιμένει έως ότου να προσπελάσει την μνήμη.



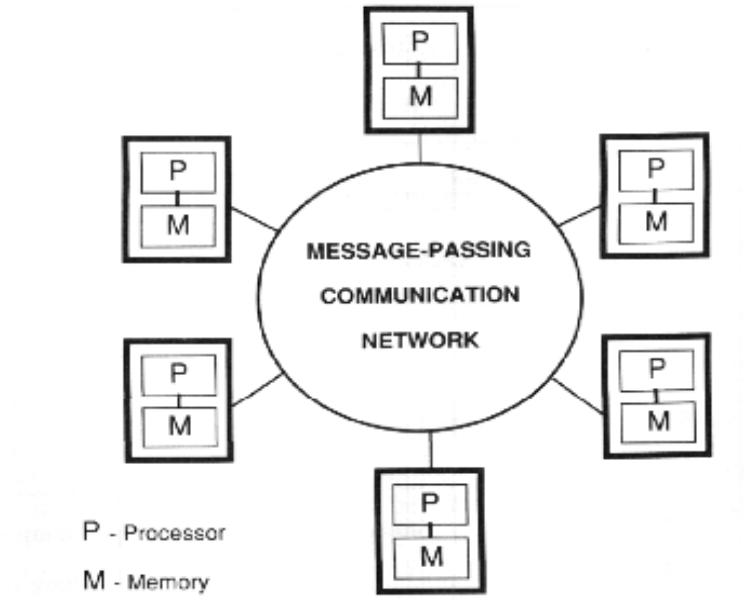
Εικόνα 5 Σύστημα Κατανεμημένης Μνήμης με Πολλαπλά Τμήματα

Είναι σημαντικό να θυμόμαστε γι΄ αυτό το μοντέλο κοινόχρηστης μνήμης, ότι τα πολλαπλά τμήματα μνήμης μαζί σχηματίζουν μια απλή διαμοιραζόμενη μνήμη, η οποία είναι προσπελάσιμη απ' όλους τους επεξεργαστές. Τη λειτουργία του δικτύου επικοινωνίας χειρίζεται εξ' ολοκλήρου το υλικό προσπέλασης μνήμης, και δεν είναι ορατό στον προγραμματιστή, ο οποίος απλά βλέπει μια ενιαία κεντρική διαμοιραζόμενη μνήμη. Έτσι το προγραμματιστικό μοντέλο είναι ίδιο για τις εικόνες 4 και 5. Υπάρχει ένας μοναδικός χώρος διεύθυνσης μνήμης που είναι ορατός πανομοιότυπα απ' όλους τους επεξεργαστές. Αυτές οι διευθύνσεις μνήμης στην πραγματικότητα κατανέμονται στα πολλαπλά τμήματα μνήμης, αλλά αυτό δεν είναι ορατό στους επεξεργαστές. Όταν ο επεξεργαστής διαβάζει ή γράφει μία συγκεκριμένη διεύθυνση μνήμης, η δραστηριότητα του δικτύου επικοινωνίας και η επιλογή του κατάλληλου τμήματος μνήμης ρυθμίζεται αυτόματα από το υλικό προσπέλασης μνήμης. Έτσι όταν κάποιος γράφει ένα πρόγραμμα για σύστημα κοινόχρηστης μνήμης, το απλό αρχιτεκτονικό μοντέλο της εικόνας 4 είναι επαρκές.

1.6.2 Παράλληλα συστήματα κατανεμημένης μνήμης

Μία άλλη προσέγγιση για τη μείωση του ανταγωνισμού των τμημάτων μνήμης στα συστήματα παράλληλης επεξεργασίας είναι η ολοκληρωτική εξάλειψη της κοινόχρηστης μνήμης και η εξασφάλιση μιας αρκετά μεγάλης τοπικής μνήμης για κάθε επεξεργαστή καθώς και ενός δικτύου επικοινωνίας για την αλληλεπίδραση και των επεξεργαστών μέσω

ενός μηχανισμού περάσματος μηνυμάτων. Αυτό το είδος κατανεμημένης μνήμης παράλληλου συστήματος επεξηγείται εικόνα 6. Ο κάθε επεξεργαστής να λειτουργεί αυτόνομα, χρησιμοποιώντας τα δεδομένα που είναι αποθηκευμένα στο δικό του τμήμα τοπικής μνήμης. Ο κάθε επεξεργαστής μπορεί επίσης να στέλνει και να λαμβάνει δεδομένα προς και από οποιονδήποτε άλλο επεξεργαστή, χρησιμοποιώντας το δίκτυο επικοινωνίας περάσματος μηνυμάτων.



Εικόνα 6 Συστήματος Κατανεμημένης Μνήμης

Η βασική οργάνωση του παράλληλου συστήματος κατανεμημένης μνήμης είναι ριζικά διαφορετική από την οργάνωση του συστήματος κοινόχρηστης μνήμης. Ένα παράλληλο σύστημα κατανεμημένης μνήμης συμπεριφέρεται με εντελώς διαφορετικό τρόπο απ' ότι ένα σύστημα κοινόχρηστης μνήμης και απαιτεί ένα διαφορετικό εννοιολογικό μοντέλο προγραμματισμού για την ανάπτυξη του λογισμικού. Τώρα ο επεξεργαστής γνωρίζει τη διαφορά ανάμεσα στα τοπικά και απομακρυσμένα τμήματα μνήμης. Κάθε ζευγάρι επεξεργαστή-μνήμης συμπεριφέρεται σαν ένα μικρό, αυτόνομα ενιαίο σύστημα υπολογιστή. Ο επεξεργαστής μπορεί να διαβάζει και να γράφει δεδομένα ελεύθερα, χρησιμοποιώντας την δική του τοπική μνήμη. Όταν οι επεξεργαστές θέλουν να ανταλλάξουν δεδομένα, τότε αυτό πρέπει να γίνει μέσω μιας ρητής ενέργειας ανταλλαγής μηνυμάτων χρησιμοποιώντας το δίκτυο επικοινωνίας. Ο επεξεργαστής έχει άμεση πρόσβαση μόνο στα δικά του τοπικά

τμήματα μνήμης και όχι στα τμήματα μνήμης που είναι συνδεδεμένα με άλλους επεξεργαστές. Όμως κάθε επεξεργαστής μπορεί να διαβάζει τιμές δεδομένων από τη δική του τοπική μνήμη και να στέλνει αυτά τα δεδομένα σε οποιονδήποτε άλλον επεξεργαστή. Έτσι αυτά τα δεδομένα μπορούν να μοιράζονται και να ανταλλάσσονται ελεύθερα ανάμεσα στους επεξεργαστές.

Υπάρχει μια μεγάλη ποικιλία διαφορετικών τοπολογιών δικτύου επικοινωνίας, που έχουν αναπτυχθεί για παράλληλα συστήματα κατανεμημένης μνήμης. Σκοπός αυτών των τοπολογιών, είναι να προσπαθήσουν να μειώσουν το κόστος και την πολυπλοκότητα του δικτύου, όταν η γρήγορη επικοινωνία μεταξύ των επεξεργαστών επιτρέπεται. Γενικά δεν είναι δυνατό σε μεγάλα παράλληλα συστήματα κατανεμημένης μνήμης να εξασφαλιστεί επικοινωνία ανάμεσα σε κάθε ζευγάρι επεξεργαστών, μια και αυτό θα απαιτούσε n^2 κανάλια επικοινωνίας για n επεξεργαστές. Για να διατηρηθεί ένα λογικό κόστος και να επιτραπεί στο σύστημα να αυξήσει εύκολα τον αριθμό των επεξεργαστών του, ο αριθμός των μονοπατιών επικοινωνίας που φτάνουν σε κάθε επεξεργαστή πρέπει να είναι σταθερός(ανεξάρτητος από τον συνολικό αριθμό επεξεργαστών) ή να αυξάνει λογαριθμικά με το συνολικό αριθμό επεξεργαστών.

Σε αυτό το κείμενο, γίνεται μια σημαντική διάκριση ανάμεσα στα συστήματα κοινόχρηστης μνήμης και στα συστήματα κατανεμημένης μνήμης. Ο προγραμματισμός αυτών των δύο κύριων κατηγοριών παράλληλων συστημάτων είναι αρκετά διαφορετικός, και απαιτεί διαφορετικές αντιλήψεις του υλικού του συστήματος. Για τον προγραμματισμό συστημάτων κοινόχρηστης μνήμης το μοντέλο είναι αυτό της εικόνας 4 πολλοί πανομοιότυποι επεξεργαστές προσπελαύνουν μια μοναδική διαμοιραζόμενη μνήμη παράλληλα. Για τον προγραμματισμό παράλληλων συστημάτων κατανεμημένης μνήμης, το μοντέλο είναι αυτό της εικόνας 5 πολλοί παράλληλοι επεξεργαστές, ο καθένας με τη δική του τοπική μνήμη, και ένα δίκτυο περάσματος μηνυμάτων για την ανταλλαγή μηνυμάτων μεταξύ των επεξεργαστών. Αρχίζοντας, ο προγραμματισμός των συστημάτων κοινόχρηστης μνήμης είναι μάλλον πιο εύκολος, γιατί μοιάζει περισσότερο με τον προγραμματισμό του απλού σειριακού συστήματος επεξεργασίας με το οποίο είμαστε ήδη εξοικειωμένοι. Έτσι, το πρώτο μισό αυτού του κειμένου επικεντρώνεται αποκλειστικά στο προγραμματισμό συστημάτων κοινόχρηστης μνήμης. Ο προγραμματισμός παράλληλων συστημάτων κατανεμημένης μνήμης, είναι περισσότερο πολύπλοκος γιατί το πρόγραμμα πρέπει να κάνει σαφή μεταφορά δεδομένων μεταξύ των επεξεργαστών. Αυτό γίνεται ακόμα πιο περίπλοκο από το γεγονός

ότι η τοπολογία διασύνδεσης των επεξεργαστών πρέπει επίσης να μελετηθεί για τον σχεδιασμό του προγράμματος

1.6.3 Παράλληλος προγραμματισμός

Αυτό το καινούργιο είδος αρχιτεκτονικής υπολογιστών με μεγάλο αριθμό επεξεργαστών εισάγει καινούργιες απαιτήσεις στο λογισμικό. Ένα πρόγραμμα για ένα κοινό σειριακό σύστημα πρέπει να διαγράψει μια σειρά από λειτουργίες που ο επεξεργαστής έχει να ακολουθήσει. Ένα πρόγραμμα για ένα παράλληλο σύστημα πρέπει να διαγράψει μια σειρά από λειτουργίες που κάθε επεξεργαστής έχει να ακολουθήσει παράλληλα, περιλαμβάνοντας λειτουργίες που συντονίζουν τους χωριστούς επεξεργαστές στην εκτέλεση μιας ενιαίας διεργασίας. Αυτή η αναγκαιότητα της δημιουργίας και του συντονισμού πολλών παράλληλων διεργασιών προσθέτει μια καινούργια διάσταση στη διαδικασία του προγραμματισμού. Αλγόριθμοι για συγκεκριμένα προβλήματα πρέπει να σχεδιαστούν με τέτοιο τρόπο ώστε να παράγεται ένας μεγάλος αριθμός παράλληλων λειτουργιών που να εκτελούνται από διαφορετικούς επεξεργαστές. Έτσι, παρόλο που οι αρχιτεκτονικές παράλληλων συστημάτων κοινόχρηστης και κατανεμημένης μνήμης έχουν εξασφαλίσει τη δυνατότητα τεράστιας αύξησης της υπολογιστικής δύναμης με λογικό κόστος, αυτή η δυνατότητα μπορεί να πραγματοποιηθεί μόνο από την πλήρη κατανόηση των παράλληλων γλωσσών προγραμματισμού και του σχεδιασμού παράλληλων αλγορίθμων.

Ένα χρήσιμο εννοιολογικό εργαλείο, για την δημιουργία προγραμμάτων για παράλληλα συστήματα είναι η κατανόηση της έννοιας της *διεργασίας*, που είναι ουσιαστικά η σειρά (διαδοχή) των λειτουργιών που μπορούν να εκτελεστούν από έναν απλό επεξεργαστή. Η *διεργασία* μπορεί να χρησιμοποιηθεί σαν η βασική δομή για το χτίσιμο παράλληλων προγραμμάτων: κάθε επεξεργαστής εκτελεί μια συγκεκριμένη διεργασία σε οποιαδήποτε στιγμή. Άτυπα, η διεργασία μπορεί να νοηθεί σαν μια υπορουτίνα ή διαδικασία που εκτελείται από ένα συγκεκριμένο φυσικό επεξεργαστή. Η διαθεσιμότητα μεγάλου αριθμού φυσικών επεξεργαστών σημαίνει ότι ένας μεγάλος αριθμός διεργασιών λογισμικού μπορεί να εκτελείται παράλληλα από το υλικό του υπολογιστή. Υποθέτοντας, ότι η δραστηριότητα της κάθε διεργασίας συμβάλλει στην ολοκλήρωση ενός απλού υπολογισμού, τότε η εκτέλεση αυτού του υπολογισμού μπορεί να είναι πολύ πιο γρήγορη απ’ ότι σε ένα σύστημα ενός επεξεργαστή. Η έννοια της διεργασίας για να είναι χρήσιμη στη δημιουργία προγραμμάτων για παράλληλα συστήματα, πρέπει να προστεθεί σαν επιπλέον

χαρακτηριστικό στις παράλληλες γλώσσες προγραμματισμού. Μετά ο προγραμματιστής μπορεί να σχεδιάσει και να κωδικοποιήσει τους παράλληλους αλγόριθμους που είναι ικανοί να εκμεταλλευτούν τη διαθεσιμότητα πολλών επεξεργαστών στο υλικό των παράλληλων συστημάτων κατανεμημένης και κοινόχρηστης μνήμης.

Για την ενσωμάτωση της έννοιας της διεργασίας σε μια παράλληλη γλώσσα, πρέπει να υπάρχουν μερικοί μηχανισμοί στη γλώσσα για τον ορισμό και τη δημιουργία καινούργιων διεργασιών. Πρέπει επίσης να υπάρχουν μερικά χαρακτηριστικά στη γλώσσα που να επιτρέπουν την διαθεσιμότητα των δεδομένων στις παράλληλες διεργασίες, έτσι ώστε να μπορούν οι διεργασίες να αλληλεπιδρούν η μια με την άλλη, καθώς δουλεύουν για τον υπολογισμό τελικών αποτελεσμάτων. Από τη στιγμή που οι διεργασίες μπορούν να εκτελούνται με μεταβαλλόμενες ταχύτητες σε διαφορετικούς φυσικούς επεξεργαστές, η γλώσσα πρέπει επίσης να διαθέτει μηχανισμούς για τον συγχρονισμό των διεργασιών. Η έννοια των παράλληλων διεργασιών και της ελεγχόμενης αλληλεπίδρασης διεργασιών επινοήθηκε στα τέλη της δεκαετίας του 1960, όταν τα συστήματα πολλαπλών χρηστών διαμοιραζόμενου χρόνου είχαν μόλις αναπτυχθεί. Από τη στιγμή που πολλαπλά προγράμματα μπορούσαν να λειτουργήσουν ταυτόχρονα σε συστήματα διαμοιραζόμενου χρόνου, η έννοια των παράλληλων διεργασιών ήταν μια χρήσιμη οργανωτική αρχή για το σχεδιασμό των λειτουργικών συστημάτων αυτών των υπολογιστών, παρόλο που το υλικό συχνά περιείχε ένα μόνο φυσικό επεξεργαστή.

Κατά τη διάρκεια της δεκαετίας του 1970-80, είχε αναπτυχθεί ένας αριθμός γλωσσών προγραμματισμού που περιλάμβανε χαρακτηριστικά για τη δημιουργία και αλληλεπίδραση των παράλληλων διεργασιών, με εφαρμογή στην ανάπτυξη λογισμικού για λειτουργικά συστήματα. Σε αυτές περιλαμβάνονταν η Pascal , Modula-2, Communication Sequential Processes, Distributed Procceses ,PLITS και η Occam. Η έννοια της διεργασίας υπήρξε χρήσιμη και για τα ένθετα συστήματα πραγματικού χρόνου και έτσι περιλήφθηκε και στη γλώσσα ADA. Η γλώσσα Argus χρησιμοποιεί παράλληλες διεργασίες για τη δημιουργία κατανεμημένων συστημάτων βάσεων δεδομένων για δίκτυα υπολογιστών.

Καθώς, τα παράλληλα συστήματα κατανεμημένης και κοινόχρηστης μνήμης έγιναν δημοφιλή στη δεκαετία του 1980, η έννοια της διεργασίας προστέθηκε σε έναν αριθμό εφαρμοσμένων γλωσσών προγραμματισμού ευρείας χρήσης όπως η Fortran, C, Pascal και Lisp. Για τα περισσότερα εμπορικά διαθέσιμα παράλληλα συστήματα, αυτό επιτεύχθηκε με το να επιτραπούν ειδικές κλήσεις συστήματος μέσα στη γλώσσα για την δημιουργία,

επικοινωνία και συγχρονισμό των διεργασιών. Λόγω της απουσίας καθιερωμένων παράλληλων γλωσσών προγραμματισμού, ο κάθε κατασκευαστής συστήματος δημιουργούσε την δική του ποικιλία από τέτοιες κλήσεις λειτουργικού συστήματος. Δύο τέτοιες γλώσσες, που έχουν εφαρμογή σε πολλά εμπορικά συστήματα είναι η Linda και Multilisp. Παρότι, υπάρχει μεγάλη ποικιλία στην ακριβή σύνταξη και σημασιολογία των χαρακτηριστικών της παράλληλης γλώσσας στα σύγχρονα παράλληλα συστήματα κατανεμημένης και κοινόχρηστης μνήμης, όλα αυτά διαπερνούνται από μερικές απλές έννοιες, με κυριότερη αυτή της διεργασίας.

Στη προσπάθεια να αποκτήσουμε μια πρώτη επαφή για μερικές από τις αρχές του παράλληλου προγραμματισμού, θα είναι χρήσιμο να μελετήσουμε σύντομα ένα απλό παράλληλο πρόγραμμα. Η ταξινόμηση είναι μια ιδιαίτερα έντονη υπολογιστική δραστηριότητα η οποία πραγματοποιείται συχνά σε πολλά συστήματα. Εφόσον είναι ένα οικείο θέμα για στους περισσότερους προγραμματιστές, αποτελεί ένα καλό σημείο εκκίνησης για τον παράλληλο προγραμματισμό. Η ταξινόμηση στα παράλληλα συστήματα έχει μελετηθεί από πολλούς ερευνητές και υπάρχει μια ευρεία ποικιλία παράλληλων αλγόριθμων ταξινόμησης που εκτελούνται αποτελεσματικά σε πολλούς τύπους παράλληλης αρχιτεκτονικής. Ένας σχετικά απλός αλγόριθμος που βασίζεται σε μια τεχνική σειριακής ταξινόμησης και που δεν είναι ιδιαίτερα αποδοτικός ονομάζεται αλγόριθμος Ταξινόμησης Σειράς (Rank Sort).

Ας υποθέσουμε ότι έχουμε ένα πρόβλημα ταξινόμησης μιας λίστας που περιέχει n αριθμούς διαφορετικούς μεταξύ τους. Σ΄ ένα πρόγραμμα που υλοποιείται με τον παραπάνω αλγόριθμο, η σειρά κάθε αριθμού μέσα στη λίστα ορίζεται ως ο συνολικός αριθμός των στοιχείων της λίστας που είναι μικρότερα από αυτόν. Ο αλγόριθμος για να υπολογίσει τη σειρά ενός τυχαίου αριθμού x , τον συγκρίνει με κάθε στοιχείο της λίστας και κρατά ένα τρέχον σύνολο του αριθμού των στοιχείων της λίστας που είναι μικρότερα του x. Σε μια ταξινομημένη λίστα, η σειρά κάθε στοιχείου θα είναι η πραγματική θέση του μέσα στη λίστα (υποθέτοντας ότι η λίστα έχει ταξινομηθεί με αύξουσα σειρά). Ο αλγόριθμος Ταξινόμησης Σειράς υπολογίζει την σειρά κάθε στοιχείου και μετά χρησιμοποιεί την πληροφορία αυτή ώστε να τοποθετήσει τα στοιχεία στη σωστή τους θέση.

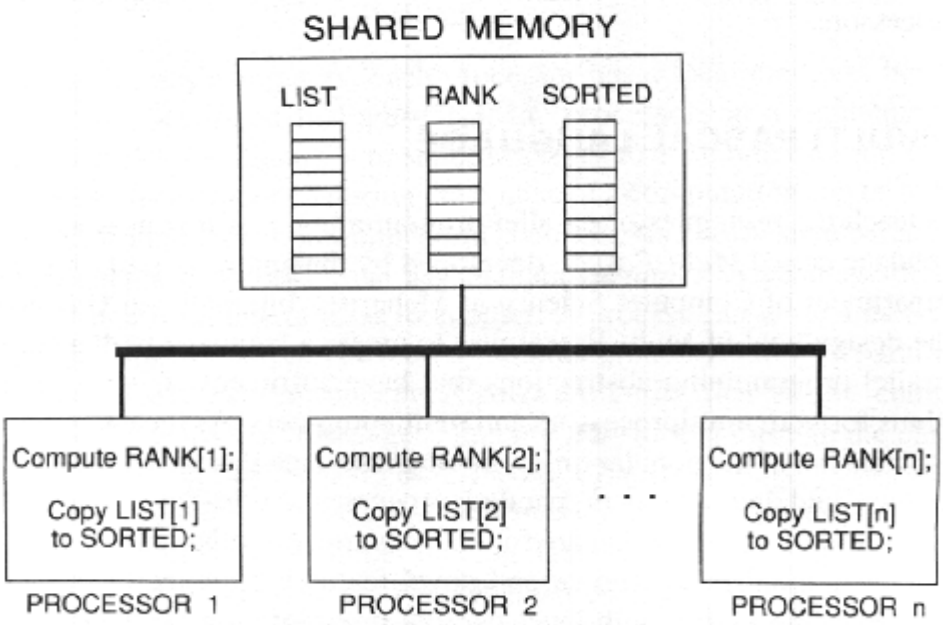
Για να γίνει πιο κατανοητός ο αλγόριθμος, δίδεται το παρακάτω παράδειγμα μιας μη ταξινομημένης λίστας και της σειράς κάθε στοιχείου εντός της λίστας:

(Μη ταξινομημένη)

LIST	RANK
15	4
10	3
39	7
8	2
22	6
4	0
19	5
6	1

Προσέξτε ότι ο πίνακας Rank δίνει ακριβώς τη θέση των στοιχείων στην ταξινομημένη λίστα. Από την στιγμή που θα υπολογιστεί ο πίνακας που περιέχει τη σειρά των στοιχείων, ο αλγόριθμος μπορεί εύκολα να τοποθετήσει τα στοιχεία στις τελικές ταξινομημένες θέσεις, χρησιμοποιώντας την τιμή που περιέχει ο πίνακας RANK στη θέση i (RANK[i]) σαν δείκτης για την τοποθέτηση του στοιχείου που βρίσκεται στην i θέση του πίνακα LIST (LIST[i]).

Ο αλγόριθμος ταξινόμησης σειράς μπορεί εύκολα να λειτουργήσει παράλληλα γιατί η σειρά κάθε στοιχείου της λίστας μπορεί να υπολογιστεί ανεξάρτητα από διαφορετικούς επεξεργαστές. Ο επεξεργαστής 1 μπορεί να υπολογίσει την τιμή του στοιχείου της θέσης 1 του πίνακα RANK (RANK[1]) παίρνοντας την τιμή που υπάρχει στη θέση 1 του πίνακα LIST(LIST[1])και συγκρίνοντας την με οποιοδήποτε άλλο στοιχείο της λίστας. Όταν αυτός ο υπολογισμός γίνεται στον επεξεργαστή 1 ο επεξεργαστής 2 μπορεί ταυτόχρονα να υπολογίζει την τιμή της θέσης 2 του πίνακα RANK (RANK[2]) συγκρίνοντας την τιμή που βρίσκεται στη θέση 2 του πίνακα LIST(LIST[2]) με κάθε άλλο στοιχείο της λίστας. Αν υπάρχουν n στοιχεία στη LIST και n επεξεργαστές, τότε κάθε επεξεργαστής i μπορεί να αναλάβει τον υπολογισμό του RANK[i] χρησιμοποιώντας το στοιχείο LIST[i]. Αυτό επεξηγείται στην εικόνα 7 για ένα παράλληλο σύστημα κοινόχρηστης μνήμης.



Εικόνα 7 Παράλληλη Ταξινόμηση Σειράς

Οι τρεις πίνακες LIST, RANK, και SORTED βρίσκονται στη διαμοιραζόμενη μνήμη, και έτσι είναι άμεσα προσπελάσιμοι απ' όλους τους επεξεργαστές. Αρχικά, τα μη ταξινομημένα στοιχεία τοποθετούνται στον πίνακα LIST. Ο επεξεργαστής 1 διαβάζει το LIST[1] από τη διαμοιραζόμενη μνήμη και μετά συνεχίζει στον υπολογισμό του RANK[1] συγκρίνοντας το LIST[1] με όλα τα άλλα στοιχεία του πίνακα LIST. Όταν υπολογιστεί το RANK(1), ο επεξεργαστής 1 μπορεί να γράψει το LIST[1] άμεσα στην κατάλληλη τελική ταξινομημένη θέση του στον πίνακα SORTED. Από τη στιγμή που και οι n επεξεργαστές υπολογίζουν παράλληλα, και τα n στοιχεία της λίστας θα τοποθετηθούν στην τελική ταξινομημένη τους θέση.

Πολλές από τις σημαντικές ιδιότητες των παράλληλων προγραμμάτων επεξηγούνται από αυτό το παράδειγμα Ταξινόμησης Σειράς όπως φαίνεται στην εικόνα 7. Η υπολογιστική δραστηριότητα του κάθε επεξεργαστή καλείται *διεργασία*. Η *διεργασία* είναι κατά βάση διαφορετική από τον επεξεργαστή, ο *επεξεργαστής* είναι η φυσική συσκευή υλικού, που έχει την δυνατότητα προσπέλασης της μνήμης και υπολογισμού νέων τιμών δεδομένων, ενώ η *διεργασία* είναι πολύ πιο αφηρημένη: η *διεργασία* είναι μια υπολογιστική δραστηριότητα. Για να έχει ο φυσικός επεξεργαστής δικαίωμα να κάνει οτιδήποτε πρέπει να του έχουν ανατεθεί κάποιες διεργασίες - αυτό σημαίνει ότι ο επεξεργαστής πρέπει να έχει μερικές

υπολογιστικές δραστηριότητες να εκτελέσει. Στην εικόνα 7 η διεργασία που εκτελείται από τον επεξεργαστή 1 είναι ‘’Να υπολογίσει το RANK(1)και να αντιγράψει το LIST(1) στο SORTED‘’. Η διεργασία που ο κάθε επεξεργαστής εκτελεί ορίζεται από το πρόγραμμα. Έτσι ο επεξεργαστής είναι συσκευή υλικού, ενώ η διεργασία μια γενική ιδέα λογισμικού. Προς το παρόν, η διεργασία μπορεί άτυπα να οριστεί σαν τον *κώδικα του προγράμματος που εκτελείται* από έναν επεξεργαστή.

Η απόδοση των παράλληλων προγραμμάτων εκτιμάται κυρίως από δύο σημαντικούς παραμέτρους: τον Χρόνο Εκτέλεσης και την Επιτάχυνση. Ο Χρόνος Εκτέλεσης που καλείται και Παράλληλος Χρόνος Εκτέλεσης, είναι απλά ο χρόνος εκτέλεσης του παράλληλου προγράμματος στην αρχιτεκτονική του παράλληλου συστήματος που θέλουμε να προσομοιώσουμε. Ο Σειριακός Χρόνος Εκτέλεσης ορίζεται σαν ο χρόνος εκτέλεσης της σειριακής έκδοσης του ίδιου αλγορίθμου, που εκτελείται με έναν μόνο επεξεργαστή. Η Επιτάχυνση του παράλληλου προγράμματος είναι ο λόγος του Σειριακού Χρόνου Εκτέλεσης προς τον Παράλληλο Χρόνο Εκτέλεσης. Ένα άλλο σημαντικό μέτρο της απόδοσης του προγράμματος είναι η Απόδοση του Επεξεργαστή. Για κάθε επεξεργαστή, αυτό ορίζεται σαν το ποσοστό του χρόνου που ο επεξεργαστής πραγματικά λειτουργεί κατά τη διάρκεια εκτέλεσης του προγράμματος. Αν αναφερθούμε στο σύνολο του προγράμματος, η Απόδοση του Συστήματος για τον συγκεκριμένο αλγόριθμο ορίζεται σαν η μέση απόδοση όλων των επεξεργαστών.

1.6.4 Παράλληλοι αλγόριθμοι

Για να εκμεταλλευτούμε τις τεράστιες υπολογιστικές δυνατότητες που προσφέρονται από την ανάπτυξη συστημάτων παράλληλης επεξεργασίας, είναι αναγκαίο να επινοηθούν παράλληλοι αλγόριθμοι που μπορούν να διαχειριστούν ένα μεγάλο αριθμό επεξεργαστών που να δουλεύουν παράλληλα για την ολοκλήρωση ενός συνολικού υπολογισμού. Σε ορισμένες περιπτώσεις, απλοί σειριακοί αλγόριθμοι μπορούν εύκολα να προσαρμοστούν για να χρησιμοποιηθούν σε παράλληλα προβλήματα. Ωστόσο στις περισσότερες περιπτώσεις, το υπολογιστικό πρόβλημα πρέπει να αναλυθεί ξανά από την αρχή και να αναπτυχθούν νέοι παράλληλοι αλγόριθμοι. Τα 15 τελευταία χρόνια, η έρευνα στο χώρο της σχεδίασης παράλληλων αλγορίθμων έχει προχωρήσει σημαντικά, καλύπτοντας ένα ευρύ φάσμα πρακτικών προβλημάτων όπως η ταξινόμηση, η επεξεργασία γραφημάτων, η επίλυση γραμμικών και διαφορικών εξισώσεων και η προσομοίωση. Επίσης, σημαντική πρακτική

εμπειρία έχει αποκτηθεί πρόσφατα με την ανάπτυξη αποδοτικών παράλληλων προγραμμάτων σε συστήματα κατανεμημένης και κοινόχρηστης μνήμης.Παρακάτω βρίσκεται μια λίστα με μερικές κατηγορίες τεχνικών με μια σύντομη περιγραφή της κάθε κατηγορίας:

Παραλληλισμός δεδομένων (Data Parallelism)

Ένας μεγάλος αριθμός δεδομένων υπόκεινται στην ίδια ή παρόμοια επεξεργασία παράλληλα. Αυτός ο τύπος παραλληλισμού είναι ιδιαίτερα χρήσιμος στους αριθμητικούς αλγορίθμους που ασχολούνται με μεγάλους πίνακες και διανύσματα.

Ο αλγόριθμος παράλληλης σειριακής ταξινόμησης που επεξήγεται στην εικόνα 7 χρησιμοποιεί τον παραλληλισμό δεδομένων: κάθε στοιχείο από την μη κατανεμημένη λίστα επεξεργάζεται με παρόμοιο τρόπο από διαφορετικό επεξεργαστή. Στην πραγματικότητα, ο παραλληλισμός δεδομένων, είναι μια μεγάλη γενική κατηγορία τεχνικών. Τα περισσότερα παράλληλα προγράμματα που επιτυγχάνουν καλή απόδοση χρησιμοποιούν κάποια μορφή παραλληλισμού δεδομένων.

Κατάτμηση δεδομένων (Data Partitioning)

Αυτός είναι ένας ιδιαίτερος τύπος παραλληλισμού δεδομένων στον οποίο ο χώρος αποθήκευσης των δεδομένων είναι φυσικά διαχωρισμένος σε παρακείμενες περιοχές, κάθε μία από τις οποίες υφίστανται επεξεργασία παράλληλα μέσω διαφορετικού επεξεργαστή. Μπορεί να υπάρξει σποραδική ανταλλαγή τιμών δεδομένων διαμέσου των ορίων της κάθε περιοχής. Αυτός ο τύπος αλγορίθμου είναι ιδιαίτερα κατάλληλος για παράλληλα συστήματα κατανεμημένης μνήμης γιατί η υπολογιστική δραστηριότητα του κάθε επεξεργαστή αφορά κυρίως την δική του περιοχή τοπικών δεδομένων και έτσι η επικοινωνία των αυτόνομων ενιαίων συστημάτων επεξεργαστών είναι σχετικά σπάνια.Για παράδειγμα ένας επαναληπτικός αριθμητικός αλγόριθμος που χρησιμοποιεί ένα διδιάστατο πίνακα δεδομένων 80x80 θέσεων μπορεί να έχει εφαρμογή σε ένα παράλληλο σύστημα κατανεμημένης μνήμης, τμηματοποιώντας τα δεδομένα του πίνακα σε 64 μη επικαλυπτόμενες περιοχές, που η κάθε μία αποτελείται από ένα πίνακα αριθμών 10x10 θέσεων. Σε ένα παράλληλο σύστημα κατανεμημένης μνήμης με 64 επεξεργαστές, η κάθε μία από τις 64 περιοχές μπορεί να αποθηκευτεί στην τοπική μνήμη ενός από τους επεξεργαστές. Στη συνέχεια οι επεξεργαστές μπορούν να συγκεντρώσουν την υπολογιστική τους δραστηριότητα στην δική τους περιοχή

τοπικών δεδομένων, ενώ σποραδικά ανταλλάσσουν πληροφορίες με άλλους επεξεργαστές με τον μηχανισμό περάσματος μηνυμάτων.

Ασύγχρονος αλγόριθμος (Asychronous Algorithm)

Κάθε παράλληλη διεργασία λειτουργεί αυτόνομα, χωρίς καθόλου συγχρονισμό ή επικοινωνία με τις άλλες διεργασίες. Οι ασύγχρονοι αλγόριθμοι αποδίδουν τόσο στα συστήματα κοινόχρηστης μνήμης όσο και στα συστήματα κατανεμημένης μνήμης και μπορεί να έχουν σαν αποτέλεσμα πολύ μεγάλες επιταχύνσεις. Η παράλληλη ταξινόμηση σειράς είναι ένα παράδειγμα ασύγχρονου αλγόριθμου. Παρατηρούμε ότι παρότι που οι επεξεργαστές προσπελαύνουν μερικά κοινά δεδομένα, ο κάθε επεξεργαστής μπορεί να υπολογίσει με ένα απόλυτα αυτόνομο τρόπο, χωρίς καμιά βοήθεια, τις ενδιάμεσες τιμές δεδομένων που παράγονται από άλλους επεξεργαστές. Αυτή η απόλυτη αυτονομία του κάθε επεξεργαστή είναι ένα από τα γνωρίσματα-κλειδιά των ασύγχρονων αλγορίθμων που καθιστά εύκολο τον προγραμματισμό τους.

Σύγχρονη επανάληψη (Synchonous Iteration)

Ο κάθε επεξεργαστής εκτελεί την ίδια επαναληπτική λειτουργία σε διαφορετικό τμήμα δεδομένων. Όμως, οι επεξεργαστές πρέπει να συγχρονίζονται στο τέλος κάθε επανάληψης. Αυτό εξασφαλίζει ότι δεν επιτρέπεται σε κανένα επεξεργαστή να αρχίσει την επόμενη επανάληψη, προτού ολοκληρώσουν και οι υπόλοιποι την τρέχουσα επανάληψη. Ο συγχρονισμός της κάθε επανάληψης είναι απαραίτητος επειδή τα δεδομένα που παράγονται από ένα συγκεκριμένο επεξεργαστή κατά τη διάρκεια της επανάληψης i χρησιμοποιούνται από πολλούς επεξεργαστές κατά την επανάληψη i+1. Έτσι, είναι απαραίτητο όλοι οι επεξεργαστές να ολοκληρώσουν την επανάληψη i, πριν κάποιους από αυτούς ξεκινήσει την επανάληψη i+1.

Πολλοί από τους κλασσικούς αριθμητικούς αλγόριθμους που χρησιμοποιούνται στις φυσικές επιστήμες όταν μετατρέπονται σε παράλληλη μορφή καταλήγουν σε σύγχρονη επανάληψη. Αυτό το είδος παράλληλου προγράμματος αποδίδει ικανοποιητικά σε παράλληλα συστήματα κοινόχρηστης μνήμης επειδή ο χρόνος που απαιτείται για τον συγχρονισμό των επεξεργαστών είναι σχετικά μικρός. Έτσι, το μειονέκτημα του συγχρονισμού δεν μειώνει πολύ την απόδοση του προγράμματος. Όμως το τίμημα του συγχρονισμού στα παράλληλα συστήματα κατανεμημένης μνήμης είναι πολύ μεγαλύτερο

εξαιτίας της κατανεμημένης φύσης των επεξεργαστών. Γι΄ αυτό το λόγο η σύγχρονη επανάληψη πρέπει να χρησιμοποιείται με προσοχή στα παράλληλα συστήματα κατανεμημένης μνήμης προκειμένου να γραφτούν προγράμματα με καλή απόδοση.

Πολλαπλοί εργαζόμενοι (Multiple Workers)

Εδώ διατηρείται μια κεντρική δεξαμενή παρόμοιων υπολογιστικών εργασιών. Υπάρχει ένας μεγάλος αριθμός από διεργασίες-εργαζόμενοι που ανακτούν εργασίες από την παραπάνω δεξαμενή, εκτελούν τους απαιτούμενους υπολογισμούς και πιθανώς προσθέτουν νέες εργασίες στην δεξαμενή. Όλη η υπολογιστική λειτουργία τερματίζει όταν η κεντρική δεξαμενή αδειάσει εντελώς. Αυτή η τεχνική των πολλαπλών εργαζομένων είναι χρήσιμη με προβλήματα εκθετικής πολυπλοκότητας, όπως είναι η αναζήτηση σε δένδρο ή σε γράφημα. Σε τέτοια προβλήματα, ένας μεγάλος αριθμός από μικρές υπολογιστικές εργασίες παράγεται δυναμικά ως αποτέλεσμα της συνολικής υπολογιστικής διαδικασίας. Από την στιγμή που δεν γνωρίζουμε εκ των προτέρων πότε ή πόσες τέτοιες εργασίες θα παραχθούν, είναι βολικό να οργανώσουμε το παράλληλο πρόγραμμα με μια δεξαμενή εργασίας (work-pool). Έτσι, όποτε μια καινούρια εργασία παράγεται, απλά προστίθεται στη δεξαμενή εργασίας, απ΄ όπου μπορεί αργότερα να ανακτηθεί από οποιαδήποτε από τις πανομοιότυπες διεργασίες-εργάτες.

Υπολογιστική διαδικασία διασωλήνωσης (Pipeline Computation)

Οι διεργασίες οργανώνονται σε μερικές κανονικές δομές όπως είναι αυτή του δακτυλίου ή του δισδιάστατου πλέγματος. Διαμέσου αυτής της κανονικής δομής τα δεδομένα, μετακινούνται μέσα στη δομή, με κάθε διεργασία να εκτελεί ένα συγκεκριμένο τμήμα της συνολικής υπολογιστικής διαδικασίας. Αυτοί οι αλγόριθμοι έχουν καλή εφαρμογή στα παράλληλα συστήματα κατανεμημένης μνήμης, λόγω του μεθοδικού τρόπου ροής των δεδομένων και της μη αναγκαιότητας για γενική προσπέλαση των διαμοιραζόμενων δεδομένων.

Προβλήματα που περιορίζουν την απόδοση των παράλληλων αλγοριθμων

Ακολουθεί μια λίστα που περιέχει μερικά από τα προβλήματα που ενδέχεται μερικές φορές να περιορίσουν την απόδοση των παράλληλων προγραμμάτων:

1. **Ο ανταγωνισμός μνήμης (Memory Contention).** . Η εκτέλεση του σε ένα επεξεργαστή καθυστερεί όταν αυτός περιμένει για να αποκτήσει πρόσβαση σ΄ ένα μια θέση μνήμης που χρησιμοποιείται εκείνη τη χρονική στιγμή από κάποιον άλλο επεξεργαστή. Το πρόβλημα μπορεί να παρουσιαστεί όταν δεδομένα διαμοιράζονται σε ένα μεγάλο αριθμό παράλληλων επεξεργαστών. Ο ανταγωνισμός της μνήμης τυπικά είναι ένα πρόβλημα που παρουσιάζεται μόνο όταν υπάρχει διαμοιραζόμενη μνήμη.
2. **Εκτεταμένος Σειριακός Κώδικας (Extensive Sequencial Code)**. Σε οποιοδήποτε παράλληλο αλγόριθμο, θα υπάρχουν πάντα τμήματα που να περιέχουν καθαρά σειριακό κώδικα στα οποία εκτελούνται συγκεκριμένα είδη κεντρικών λειτουργιών όπως είναι η αρχικοποίηση των μεταβλητών. Το αποτέλεσμα του σειριακού κώδικα είναι να στερεί από μερικούς αλγόριθμους αρκετό ποσοστό από τη μέγιστη συνολική επιτάχυνση που μπορούν να πετύχουν.
3. **Χρόνος δημιουργίας των διεργασιών (Process Creation Time).**.Σε οποιοδήποτε πραγματικό σύστημα, η δημιουργία των παράλληλων διεργασιών απαιτεί ένα συγκεκριμένο τμήμα του συνολικού χρόνου εκτέλεσης. Αν οι διεργασίες είναι σχετικά μικρής διάρκειας είναι δυνατόν σε μερικούς αλγόριθμους, ο χρόνος δημιουργίας τους να ξεπεράσει το χρόνο που εξοικονομείται από τον παραλληλισμό. Συχνά αυτός ο παράγοντας αγνοείται στα εγχειρίδια και τα βιβλία για παράλληλους αλγόριθμους.
4. **Καθυστέρηση Επικοινωνίας (Communication Delay).** Αυτό γενικά συμβαίνει μόνο στα παράλληλα συστήματα κατανεμημένης μνήμης επειδή οι επεξεργαστές επικοινωνούν με τον μηχανισμό περάσματος μηνυμάτων. Σε ορισμένες περιπτώσεις, η επικοινωνία ανάμεσα σε δύο επεξεργαστές μπορεί να χρειαστεί την προώθηση του μηνύματος σε ενδιάμεσους επεξεργαστές του δικτύου επικοινωνίας. Οι επακόλουθες καθυστερήσεις επικοινωνίας μπορεί να μειώσουν αισθητά την ταχύτητα εκτέλεσης ορισμένων παράλληλων αλγόριθμων.
5. **Καθυστέρηση Συγχρονισμού (Sychronization Delay).** Όταν οι παράλληλες διεργασίες συγχρονίζονται, αυτό σημαίνει ότι μια διεργασία μπορεί να αναγκαστεί να περιμένει μια άλλη. Σε μερικά παράλληλα προγράμματα, οι επακόλουθες καθυστερήσεις μπορεί να προκαλέσουν ‘λειτουργική συμφόρηση’ και μείωση στη συνολική επιτάχυνση.
6. **Ανισορροπία Φορτίου (Load Imballance)**. Σε ορισμένα παράλληλα προγράμματα, οι υπολογιστικές εργασίες παράγονται δυναμικά και με απρόβλεπτο τρόπο και

πρέπει να μεταβιβάζονται στους επεξεργαστές αμέσως μόλις παράγονται. Έτσι όμως είναι δυνατόν, ορισμένοι επεξεργαστές να παραμένουν αδρανείς ενώ άλλοι να έχουν περισσότερες υπολογιστικές εργασίες από αυτές που μπορούν να χειριστούν.

ΚΕΦΑΛΑΙΟ 2

Μελέτη λογισμικών παράλληλης επεξεργασίας

2.1 Λογισμικά παράλληλης επεξεργασίας

Τα λογισμικά παράλληλης επεξεργασίας διαχειρίζονται την εκτέλεση ενός προγράμματος το όποιο είναι γραμμένο για παράλληλη επεξεργασία, με τους στόχους της απόκτησης απεριόριστης επεκτασιμότητας (να είναι σε θέση να χειριστεί μια αύξηση του αριθμού των αλληλεπιδράσεων ταυτόχρονα) και τη μείωση του χρόνου εκτέλεσης. Οι εφαρμογές που επωφελούνται από παράλληλη επεξεργασία χωρίζονται περίπου σε δυο κατηγορίες

1. Για επεξεργασία επιχειρηματικών δεδομένων(business data processing and technical)
2. Για επιστημονική επεξεργασία(scientific processing.).

Στην παρακάτω ενότητα θα αναφέρουμε μερικά λογισμικά παράλληλης επεξεργασίας τα όποια επί των πλείστων κατατάσσονται στην δεύτερη κατηγορία ,δηλαδή για επιστημονική επεξεργασία.

2.1.1 Matlab

Η MATLAB είναι μια υψηλού επιπέδου τεχνική γλώσσα υπολογισμού και ένα διαλογικό περιβάλλον για την ανάπτυξη αλγορίθμων, την απεικόνιση στοιχείων, την ανάλυση στοιχείων, και τον αριθμητικό υπολογισμό. Χρησιμοποιώντας το προϊόν MATLAB, υπάρχει η δυνατότητα επίλυσης τεχνικών προβλημάτων υπολογισμού γρηγορότερα από ότι με τις παραδοσιακές γλώσσες προγραμματισμού, όπως η C, η C++, και η Fortran.

Το MATLAB μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα εφαρμογών όπως στην επεξεργασία σημάτων, ήχου και εικόνας, χρηματοοικονομικών επικοινωνιών, Συστημάτων Αυτομάτου Ελέγχου, βελτιστοποίησης της υπολογιστικής βιολογίας κ.α.. Οι πρόσθετες εργαλειοθήκες (συλλογές έτοιμου κώδικα) επεκτείνουν το περιβάλλον MATLAB για να λύσουν τις ιδιαίτερες κατηγορίες προβλημάτων σε αυτούς τους τομείς εφαρμογής.
Το MATLAB παρέχει διάφορα χαρακτηριστικά γνωρίσματα για την τεκμηρίωση και τη διανομή μίας εργασίας. Παρέχει, ακόμα, τη δυνατότητα ενσωμάτωσης του κώδικα

MATLAB σε άλλες γλώσσες και εφαρμογές, και τη δυνατότητα διανομής των αλγορίθμων και των εφαρμογών του.

Παράλληλα Υπολογιστικά Συστήματα του MATLAB

Τα παράλληλα υπολογιστικά συστήματα επιτρέπουν στον χρήστη της εφαρμογής MATLAB να μεταφέρει το φόρτο εργασίας από μία σύνοδο του MATLAB (τον πελάτη) σε άλλες συνόδους του MATLAB, οι οποίες ονομάζονται εργάτες (workers) ή εργαστήρια (labs). Ο χρήστης έχει τη δυνατότητα να χρησιμοποιήσει μεγάλο πλήθος εργατών ή εργαστηρίων ώστε να επωφεληθεί από την κατανεμημένη ή την παράλληλη επεξεργασία, αντίστοιχα. Έχει τη δυνατότητα να χρησιμοποιήσει έναν μόνο εργάτη ή εργαστήριο για να επωφεληθεί από την υπολογιστική ισχύ ενός επιπλέον υπολογιστή, είτε απλά για να διατηρήσει την αρχική σύνοδο πελάτη του MATLAB ελεύθερη.
Η εργαλειοθήκη Distributed Computing Toolbox επιτρέπει στον χρήστη να χρησιμοποιήσει στο τοπικό του μηχάνημα, εκτός από τη αρχική σύνοδο πελάτη του MATLAB, τέσσερις επιπλέον εργάτες ή εργαστήρια. Το λογισμικό MATLAB Distributed Computing Server επιτρέπει στον χρήστη να χρησιμοποιήσει όσους περισσότερους εργάτες ή εργαστήρια, από μία απομακρυσμένη συστοιχία υπολογιστών, του επιτρέπει η άδεια του.

Τυπικές Περιπτώσεις Χρήσης των Παράλληλων Υπολογιστικών Συστημάτων του MATLAB

Παράλληλοι βρόχοι-for (parfor)

Πολλές εφαρμογές εμπεριέχουν τμήματα κώδικα τα οποία επαναλαμβάνονται. Συχνά σε τέτοιες περιπτώσεις για να αποφευχθεί η επανάληψη χρησιμοποιείται ένας βρόχος-for. Η δυνατότητα της παράλληλης εκτέλεσης κώδικα, σε έναν υπολογιστή ή σε μία συστοιχία υπολογιστών, μπορεί να βελτιώσει σημαντικά την απόδοση σε πολλές περιπτώσεις. Οι σημαντικότερες περιπτώσεις όπου υπάρχει η δυνατότητα να παρουσιαστεί σημαντική βελτίωση είναι οι εξής:

- **Εφαρμογές σάρωσης παραμέτρων**

Πολλαπλές επαναλήψεις: Ένας βρόχος υπάρχει περίπτωση να απαιτεί πολύ ώρα για την εκτέλεση του, επειδή συνίσταται από πολλαπλές επαναλήψεις. Κάθε επανάληψη είναι

πιθανό να εκτελείται σε πολύ μικρό χρονικό διάστημα, αλλά για την ολοκλήρωση χιλιάδων ή εκατομμυρίων επαναλήψεων σειριακά απαιτείται αρκετός χρόνος.

Μακροσκελής επαναλήψεις: Ένας βρόχος υπάρχει πιθανότητα να μην εμπεριέχει μεγάλο πλήθος επαναλήψεων, αλλά η κάθε επανάληψη του να απαιτεί σημαντικό χρόνο για την εκτέλεση της.

Ακολουθίες από ελέγχους με ανεξάρτητα τμήματα κώδικα

Εφαρμογές, οι οποίες τρέχουν μία ακολουθία από ασυσχέτιστες διεργασίες, έχουν την δυνατότητα να εκτελεστούν ταυτόχρονα σε ξεχωριστούς πόρους του συστήματος. Σε περιπτώσεις όπου συνδυάζονται ασυσχέτιστες διεργασίες ο βρόχος-for υπάρχει περίπτωση να μην χρησιμοποιείται, αλλά ένας παράλληλος βρόχος-for πιθανότατα μπορεί να αποτελέσει την κατάλληλη λύση.

Η εργαλειοθήκη Distributed Computing Toolbox βελτιώνει την απόδοση εκτέλεσης ενός τέτοιου βρόχου επιτρέποντας διάφορους εργάτες του MATLAB να εκτελούν μεμονωμένες επαναλήψεις του βρόχου ταυτόχρονα (Parallel for-Loops). Για παράδειγμα, ένας βρόχος που αποτελείται από 100 επαναλήψεις μπορεί να εκτελεστεί ταυτόχρονα από μία συστοιχία με 20 εργάτες MATLAB, ώστε κάθε εργάτης να εκτελεί μόνο 5 επαναλήψεις του βρόχου. Η βελτίωση, όμως, που παρατηρείται στην ταχύτητα της εκτέλεσης δεν είναι ίση με 20 φορές και αυτό οφείλεται στις επικεφαλίδες επικοινωνίας και στην δικτυακή συμφόρηση. Παρόλα αυτά, η βελτίωση στην ταχύτητα εκτέλεσης είναι σημαντική. Ακόμα και εάν ο χρήστης διαθέτει ένα πολυπύρηνο μηχάνημα, όπου ο κάθε πυρήνας είναι ένας εργάτης, μπορεί να παρατηρήσει αξιόλογη βελτίωση της απόδοσης του προγράμματος του.

Μεταφορά Φόρτου Εργασίας

Δουλεύοντας διαδραστικά σε μία σύνοδο MATLAB, μπορεί κανείς να μεταφέρει το φόρτο εργασίας στη σύνοδο ενός MATLAB εργάτη. Η εντολή για να γίνει αυτή η εργασία είναι ασύγχρονη, που σημαίνει ότι η τρέχουσα σύνοδος MATLAB δεν έχει μπλοκάρει και μπορεί κανείς να συνεχίσει τη δική του διαδραστική σύνοδο ενώ ο εργάτης MATLAB είναι απασχολημένος εκτελώντας τον κώδικα. Ο εργάτης MATLAB μπορεί είτε να εκτελεστεί

στο ίδιο μηχάνημα με τον πελάτη, είτε σε μία απομακρυσμένη συστοιχία υπολογιστών (Evaluating Functions in a Cluster).

Τεράστια Σύνολα Δεδομένων

Εάν ένας πίνακας είναι υπερβολικά μεγάλος για τη μνήμη ενός υπολογιστή, δεν είναι δυνατόν να τον διαχειριστεί εύκολα μία μοναδική σύνοδος του MATLAB. Η εργαλειοθήκη Distributed Computing Toolbox επιτρέπει την κατανομή του πίνακα σε ένα πλήθος εργαστηρίων του MATLAB, έτσι ώστε κάθε εργαστήριο να περιλαμβάνει μόνο ένα τμήμα του πίνακα. Παρόλα αυτά, υπάρχει η δυνατότητα να αντιμετωπιστεί ολόκληρος ο πίνακας ως μία οντότητα. Κάθε εργαστήριο χειρίζεται μόνο ένα μέρος από τον πίνακα και όλα τα εργαστήρια έχουν τη δυνατότητα να μεταφέρουν δεδομένα μεταξύ τους όποτε είναι απαραίτητο, όπως για παράδειγμα στον πολλαπλασιασμό πινάκων. Ένας τεράστιος αριθμός από λειτουργίες και συναρτήσεις πινάκων έχουν αναπτυχθεί και βελτιωθεί ώστε ο χρήστης του MATLAB να έχει τη δυνατότητα να δουλεύει με κατανεμημένους πίνακες.

2.1.2 GNU Octave

Εισαγωγή

Η GNU Octave είναι μια υψηλού επιπέδου γλώσσα, η οποία προορίζεται κυρίως για αριθμητικούς υπολογισμούς. Παρέχει μια διεπαφή με κυρίαρχο στοιχείο το Παράθυρο Εντολών. Χρησιμοποιείται, κυρίως, για την επίλυση γραμμικών και μη γραμμικών αριθμητικών προβλημάτων. Αλλά και για την εκτέλεση άλλων αριθμητικών πειραμάτων χρησιμοποιώντας μια γλώσσα που είναι συμβατή επί το πλείστον με το MATLAB.

Σχετικά με την Octave

Η αρχική ιδέα για το πρόγραμμα Octave συλλήφθηκε περίπου το 1988. Στην αρχή προοριζόταν να χρησιμοποιηθεί για τις απαιτήσεις μίας σειράς μαθημάτων που αφορούσαν το σχεδιασμό χημικών αντιδραστήρων. Η ανάπτυξη του προγράμματος με τη μορφή που

έχει σήμερα άρχισε από John W. Eaton το 1992. Η πρώτη λειτουργικά ολοκληρωμένη έκδοση του Octave παρουσιάστηκε στις 17 Φεβρουαρίου 1994.

Η GNU Octave, σήμερα, διαθέτει εκτεταμένα εργαλεία για την επίλυση κοινών προβλημάτων που αφορούν την αριθμητική γραμμική άλγεβρα, για την ανεύρεση των ριζών από μη γραμμικές εξισώσεις, για την ενσωμάτωση συνηθισμένων συναρτήσεων, για τη χρήση πολυωνύμων και για την ενσωμάτωση συνηθισμένων διαφορικών – αλγεβρικών εξισώσεων. Είναι εύκολα επεκτάσιμη και προσαρμόσιμη μέσω συναρτήσεων, οι οποίες καθορίζονται από το χρήστη της Octave και είναι γραμμένες είτε στη γλώσσα Octave, είτε σε άλλες γλώσσες (MATLAB).

Τα τεχνικά χαρακτηριστικά της γλώσσας Octave είναι τα εξής:

- Έχει γραφτεί σε γλώσσα C++ και χρησιμοποιεί τις βιβλιοθήκες STL.
- Έχει ένα μεταγλωττιστή που χρησιμοποιεί για την μεταγλώττιση των προγραμμάτων.
- Είναι επεκτάσιμη χρησιμοποιώντας δυναμικά φορτώσιμες ενότητες

Ο μεταγλωττιστής της Octave χρησιμοποιεί την gnuplot και το λογισμικό Grace για να μπορέσει να δημιουργήσει εικόνες, γραφικές παραστάσεις και διαγράμματα.
Τέλος, πρέπει να αναφερθεί ότι είναι ελεύθερο λογισμικό, το οποίο μπορεί να διανεμηθεί και/ή να τροποποιηθεί υπό τους όρους της General Public Licence (GPL) .

Παράλληλη Επεξεργασία μέσω του Octave

Τα εργαλεία για παράλληλη επεξεργασία που διαθέτει το Octave είναι, συνήθως, βασισμένα σε εργαλεία του MATLAB.
Τα σημαντικότερα εργαλεία που διαθέτει το Octave για τον παραλληλισμό προγραμμάτων είναι το πακέτο Multicore και το MPI Toolbox. Περιληπτικά, το πακέτο Multicore του Octave χρησιμοποιεί κάποιες συναρτήσεις για την πραγματοποίηση παράλληλης επεξεργασίας σε πολλαπλούς πυρήνες σε ένα μόνο μηχάνημα ή με πολλούς υπολογιστές που έχουν πρόσβαση σε έναν κοινό κατάλογο. Το MPI Toolbox του Octave, το οποίο ουσιαστικά αποτελεί το σημαντικότερο εργαλείο του Octave για παράλληλη επεξεργασία,

χρησιμοποιεί το πρότυπο MPI. Το πρότυπο MPI (Message Passing Interface) είναι ένα πρωτόκολλο επικοινωνίας που χρησιμοποιεί μία σειρά συναρτήσεων οι οποίες μέσω της ανταλλαγής μηνυμάτων μεταξύ υπολογιστών δίνουν τη δυνατότητα παράλληλης εκτέλεσης μίας εφαρμογής.

Τυπικές Περιπτώσεις Χρήσης της Παράλληλης Επεξεργασίας μέσω του Octave

Όπως έχει αναφερθεί στην προηγούμενη ενότητα το πρωτόκολλο επικοινωνίας MPI είναι το σημαντικότερο εργαλείο του Octave για παράλληλη επεξεργασία. Το πρότυπο αυτό είναι κατάλληλο για την υλοποίηση εφαρμογών, οι οποίες είτε περιέχουν μεγάλο πλήθος δεδομένων, είτε περιέχουν μεγάλο πλήθος εντολών, είτε έναν συνδυασμό των παραπάνω. Στη συνέχεια αναπτύσσονται κάποια μοντέλα στα οποία έχει τη δυνατότητα να χρησιμοποιηθεί το πρότυπο MPI και μέσω της παράλληλης επεξεργασίας να προσφέρει σημαντική ελάττωση στο χρόνο εκτέλεσης τους.

Το μοντέλο Απλή Εντολή Πολλαπλά Δεδομένα (ΑΕΠΔ)

Το μοντέλο αυτό (Single Instruction Multiple Data, SIMD) αναφέρεται σε μία απλή εντολή, η οποία εκτελείται σε διαφορετικά δεδομένα. Μία τυπική περίπτωση, όπου κάποιος μπορεί να συναντήσει το μοντέλο αυτό είναι ένας βρόχος που η επαναλήψεις του είναι ανεξάρτητες μεταξύ τους.
Ένας βρόχος, όπως έχει αναφερθεί και στην αντίστοιχη ενότητα του MATLAB, μπορεί είτε να έχει πολλαπλές επαναλήψεις, είτε να έχει μακροσκελής επαναλήψεις. Σε κάθε περίπτωση το πρότυπο MPI δίνει τη δυνατότητα βελτίωσης τέτοιων βρόχων, επιτρέποντας διάφορες διεργασίες να εκτελούν μεμονωμένες επαναλήψεις του βρόχου ταυτόχρονα. Η βελτίωση, δηλαδή στο μοντέλο αυτό πραγματοποιείται με τον παραλληλισμό των δεδομένων.

Το μοντέλο Πολλαπλές Εντολές Πολλαπλά Δεδομένα (ΠΕΠΔ)

Το μοντέλο αυτό (Multiple Instructions Multiple Data, MIMD) αναφέρεται σε πολλαπλές εντολές, οι οποίες εκτελούνται σε διαφορετικά δεδομένα. Υπάρχει περίπτωση, σε μία εφαρμογή, να βρίσκεται μία σειρά διαφορετικών εντολών οι οποίες εκτελούνται σε διαφορετικά δεδομένα. Όπως και στην περίπτωση των βρόχων, μπορεί είτε το πλήθος των εντολών αυτών να είναι μεγάλο, είτε κάθε μία από τις εντολές να απαιτεί πολύ χρόνο για να

εκτελεστεί. Και στις δύο περιπτώσεις η σειριακή εκτέλεση των εντολών αυτών απαιτεί σημαντικό χρόνο για να ολοκληρωθεί. Το πρότυπο MPI δίνει τη δυνατότητα βελτίωσης τέτοιων τμημάτων κώδικα, επιτρέποντας διάφορες διεργασίες να εκτελούν ταυτόχρονα τέτοιες εντολές. Η βελτίωση, δηλαδή στο μοντέλο αυτό πραγματοποιείται με τον παραλληλισμό των εντολών.

Το μοντέλο Απλό Πρόγραμμα Πολλαπλά Δεδομένα (ΑΠΠΔ)

Το μοντέλο αυτό (Single Program Multiple Data, SPMD) αναφέρεται στο ίδιο πρόγραμμα, το οποίο εκτελείται σε διαφορετικά δεδομένα, χωρίς όμως να απαιτείται συγχρονισμός σε επίπεδο εντολής. Υπάρχει περίπτωση, σε μία εφαρμογή να καλείται πολλές φορές μία συνάρτηση που εκτελείται σε διαφορετικά δεδομένα.
Η σειριακή εκτέλεση μίας τέτοιας συνάρτησης είναι χρονοβόρα. Το πρότυπο MPI δίνει τη δυνατότητα βελτίωσης μίας τέτοιας εφαρμογής, εκτελώντας παράλληλα το πρόγραμμα που εκτελείται σε διαφορετικά δεδομένα. Η βελτίωση, δηλαδή στο μοντέλο αυτό πραγματοποιείται με τον παραλληλισμό του προγράμματος.

2.1.3 Scilab

Το Scilab είναι ελεύθερο και ανοιχτού κώδικα λογισμικό το όποιο είναι χρήσιμο για αριθμητικούς υπολογισμούς αλλά και για παράλληλη επεξεργασία και παρέχει ένα ισχυρό υπολογιστικό περιβάλλον για τους μηχανικούς αλλά και για επιστημονικές εφαρμογές
Το Scilab περιλαμβάνει εκατοντάδες μαθηματικές συναρτήσεις. Έχει μια υψηλού επίπεδου γλωσσά προγραμματισμού που επιτρέπει την πρόσβαση σε προηγμένες δομές δεδομένων καθώς και σε 2-D 3-D γραφικά
Εν συντομία οι λειτουργιές του Scilab είναι οι εξής:

- Μαθηματικά και προσομοίωση
- 2-D και 3-D γραφική απεικόνιση
- Βελτιστοποίηση αλγορίθμων
- Στατιστικά
- Σχεδιασμός Συστήματος Έλεγχου και ανάλυση
- Επεξεργασία Σήματος
- Ανάπτυξη εφαρμογών
- Υβριδικά δυναμικά συστήματα και μοντελοποίηση προσομοιωτή

2.1.4 MapReduce/Hadoop

Το MapReduce είναι ένα προγραμματιστικό μοντέλο και μια εφαρμογή για την επεξεργασία και την παραγωγή μεγάλου όγκου δεδομένων. Τα τελευταία χρόνια παρατηρείται η ανάγκη διαχείρισης όλο και μμεγαλύτερου όγκου δεδομένων πληροφοριών(άνθρωποι ανεβάζουν βίντεο στο διαδίκτυο, βγάζουν φωτογραφίες μέσω των κινητών τους τηλεφώνων, αναβαθμίζουν τη σελίδα τους στο Facebook, αφήνουν σχόλια στο διαδίκτυο και πολλά ακόμη) γεγονός που ανάγκασε μεγάλες εταιρίες του χώρου όπως οι Google, Yahoo!, Amazon και Microsoft να αναζητήσουν νέα εργαλεία για την επεξεργασία τέτοιων μεγάλων συνόλων δεδομένων. Η Google πρώτη δημοσίευσε το Map Reduce ένα παράλληλο προγραμματιστικό μοντέλο για την κλιμάκωση των δεδομένων και αμέσως προσέλκυσε το ενδιαφέρον αρκετών εταιριών που αντιμετώπιζαν παρόμοια προβλήματα. Ο Doug Cutting ανέπτυξε μια ανοιχτού κώδικα έκδοση του Map Reduce και την ονόμασε Hadoop. Αμέσως μετά η Yahoo και οι υπόλοιποι κινητοποιήθηκαν και υποστήριξαν αυτή την προσπάθεια.
Σήμερα, το Hadoop είναι ένα σημαντικό κομμάτι της υπολογιστικής υποδομής για πολλές web εταιρίες, όπως η Yahoo, Facebook, LinkedIn και Twitter. Πολύ περισσότερες παραδοσιακές εταιρίες media και τηλεπικοινωνιών αρχίζουν να καταφεύγουν σε αυτό το σύστημα. Αρκετές εταιρίες όπως οι New York Times, China Mobile και IBM χρησιμοποιούν το Hadoop.
Αυτό που έχει σημασία είναι ότι ένας προγραμματιστής σήμερα, πρέπει να γνωρίζει σχεσιακές βάσεις δεδομένων, δικτύωση και ασφάλεια, πράγματα που θεωρούνταν προαιρετικές δεξιότητες δυο δεκαετίες νωρίτερα. Παρόμοια, καλή κατανόηση της κατανεμημένης επεξεργασίας δεδομένων θα αποτελέσει σημαντικό κομμάτι της εργαλειοθήκης του προγραμματιστή στο μέλλον. Πρωτοπόρα Πανεπιστήμια όπως το Stanford και το CMU, έχουν ήδη εντάξει το Hadoop στο πρόγραμμα μαθημάτων της επιστήμης υπολογιστών.

2.1.4.1 Ιστορια Hadoop

Το Hadoop ξεκίνησε ως ένα κομμάτι του project του Nutch το οποίο με τη σειρά του ήταν ένα υπό project του Apache Lucene. Ο Doug Cutting δημιούργησε όλα αυτά τα τρία projects και καθένα από αυτά ήταν μια λογική ακολουθία των προηγούμενων.
Το Lucene είναι μια βιβλιοθήκη για ευρετηριοποίηση πλήρους κειμένου και για αναζήτηση. Αν μας δοθεί μια συλλογή κειμένων ο προγραμματιστής μπορεί πολύ εύκολα να προσθέσει

τη δυνατότητα αναζήτησης στο έγγραφο χρησιμοποιώντας τη μηχανή Lucene. Desktop search, enterprise search και πολλές domain-specific μηχανές αναζήτησης έχουν δημιουργηθεί χρησιμοποιώντας το Lucene. Το Nutch είναι η πιο φιλόδοξη επέκταση του Lucene. Αυτό που κάνει είναι να προσπαθεί να χτίσει μια ολοκληρωμένη μηχανή αναζήτησης στο διαδίκτυο χρησιμοποιώντας το Lucene ως βασικό του συστατικό. Το Nutch περιέχει συντακτικούς αναλυτές για HTML, ένα web crawler, μια βάση δεδομένων συνδεδεμένου γράφου και διάφορα επιπλέον συστατικά απαραίτητα σε μια μηχανή αναζήτησης στο διαδίκτυο. Ο Doug Cutting οραματίστηκε το Nutch σαν ένα εναλλακτικό μέσο σε σχέση με τις εμπορικές τεχνολογίες όπως της Google.

Ακόμη, έχοντας προσθέσει συστατικά όπως ένα crawler και ένα parser, μια μηχανή αναζήτησης στο διαδίκτυο διαφέρει από μια βασική μηχανή αναζήτησης εγγράφων όσον αφορά την κλιμάκωση. Ενώ το Lucene στοχεύει στο να ευρετηριοποιήσει εκατομμύρια έγγραφα, το Nutch πρέπει να είναι ικανό να χειριστεί δισεκατομμύρια από ιστοσελίδες χωρίς να γίνεται υπερβολικά ακριβό στη λειτουργία. Το Nutch τρέχει σε ένα κατανεμημένο cluster εμπορικού hardware. Η πρόκληση για την ομάδα δημιουργίας του Nutch είναι να διευθετήσει θέματα κλιμάκωσης του λογισμικού. Το Nutch χρειάζεται ένα επίπεδο για να χειριστεί τις κατανεμημένες διεργασίες, την αφθονία, το αυτόματο failover και το load balancing.Αυτές οι προκλήσεις είναι συνηθισμένες.

Γύρω στο 2004, η Google εξέδωσε δύο papers που περιέγραφαν το Google File System(GFS) και το πλαίσιο του MapReduce . Η Google ισχυρίστηκε ότι χρησιμοποίησε αυτές τις δύο τεχνολογίες για να κλιμακώσει το δικό της σύστημα αναζήτησης. Ο Doug Cutting αμέσως αντιλήφθηκε ότι μπορούσε να εφαρμόσει ιδανικά αυτές τις τεχνολογίες στο Nutch, και έτσι η ομάδα του ανέπτυξε το νέο πλαίσιο και εισήγαγε το Nutch σε αυτό. Η νέα υλοποίηση ενίσχυσε άμεσα την κλιμάκωση του Nutch. Ξεκίνησε να χειρίζεται μια σειρά από εκατοντάδες εκατομμύρια ιστοσελίδες και μπορούσε να τρέξει σε clusters από δωδεκάδες κόμβων. Ο Doug συνειδητοποίησε ότι ένα αφιερωμένο project ειδικά για να διανθίσει τις δύο τεχνολογίες ήταν επιβεβλημένο για να έχουμε κλιμάκωση στο διαδίκτυο, και έτσι το Hadoop γεννήθηκε. Η Yahoo προσέλαβε τον Doug τον Ιανουάριο του 2006 για να εργαστεί με μια εξειδικευμένη ομάδα στην ανάπτυξη του Hadoop σαν ένα ανοιχτού κώδικα Project. Δύο χρόνια αργότερα, το Hadoop πέτυχε την κατάσταση ενός Apache Top Level Project. Λίγο αργότερα στις 19 Φεβρουαρίου του 2008 η Yahoo ανακοίνωσε ότι το Hadoop που έτρεχε σε μια 10000+ συστάδα του πυρήνα Linux ήταν το σύστημα παραγωγής της για ευρετηριοποίηση του Web. Το Hadoop είχε πραγματικά χτυπήσει τη web κλιμάκωση.

ΚΕΦΑΛΑΙΟ 3

Εργαστηριακή υλοποίηση

3.1Θεωρητικό υπόβαθρο

Σ αυτό το κεφάλαιο παρουσιάζεται το εργαστηριακό κομμάτι της πτυχιακής εργασίας.Σκοπός της εργασίας ήταν να μελετηθούν λογισμικά παράλληλης επεξεργασίας και παράλληλοι αλγόριθμοι καθώς και η μέθοδος κατανεμημένης επεξεργασίας.Για να επετεύχθη αυτό έπρεπε να γίνει η επιλογή ενός λογισμικού το όποιο να πληροί τις προϋποθέσεις για εκτέλεση παράλληλων αλγορίθμων, έπειτα γίνει η εγκατάσταση σε τρεις υπολογιστές που να είναι μεταξύ τους συνδεδεμένοι σε ένα τοπικό δίκτυο και να μελετηθούν οι παράλληλοι αλγόριθμοι καθώς και το ίδιο το λογισμικό.

Τι χρησιμοποιήθηκε

Για αυτήν την εργασία χρησιμοποιήθηκε το λειτουργικό σύστημα Linux και η έκδοση Ubuntu 12.4 και το λογισμικό GNU Octave. Επίσης τρεις υπολογιστές συνδεδεμένοι μεταξύ τους σε ένα τοπικό δίκτυο με τα εξής επεξεργαστικά χαρακτηριστικά:

- Ταχύτητα επεξεργαστή 2,66 GH
- Μέγεθος μνήμης 4 GB
- Τετραπυρηνοι (4cores) 32bit επεξεργαστές

Για να πραγματοποιηθεί η εγκαταστήσει του Linux χρησιμοποιήθηκε το λογισμικό Wubi

3.2 Λογισμικό Wubi

Το **Wubi** (αρχικά του *Windows-based Ubuntu Installer*, ελεύθερα *Πρόγραμμα εγκατάστασης του Ubuntu βασισμένο στα Windows*) είναι μία εφαρμογή η οποία επιτρέπει την εγκατάσταση της διανομής Linux Ubuntu μέσα από το περιβάλλον των Windows.

Ο στόχος του Wubi είναι να επιτρέψει σε χρήστες χωρίς προηγούμενη εμπειρία με Linux να δοκιμάσουν τη διανομή Ubuntu χωρίς να είναι απαραίτητη η κατάτμηση του σκληρού τους δίσκου, ενέργεια που θεωρείται η δυσκολότερη μιας εγκατάστασης Linux. Συγκεκριμένα, το Ubuntu εγκαθίσταται μέσα στο σύστημα αρχείων των Windows, χωρίς να δημιουργηθεί

ξεχωριστή διαμέριση (partition) για αυτό. Τα Windows "βλέπουν" το φάκελο στον οποίο βρίσκεται το Ubuntu σαν ξεχωριστό δίσκο. Το Wubi επιτρέπει επίσης την *απεγκατάσταση* του Ubuntu μέσα από τα Windows.

Δεν είναι μια εικονική μηχανή, αλλά δημιουργεί μια αυτόνομη εγκατάσταση γνωστή και ως εικόνα δίσκου. Δεν είναι μια διανομή Linux από μόνη της, αλλά μάλλον ένα πρόγραμμα εγκατάστασης για το Ubuntu.

Το Wubi προσθέτει μια καταχώρηση στο μενού εκκίνησης των Windows η οποία επιτρέπει στο χρήστη να τρέχει Linux. Το Ubuntu είναι εγκατεστημένο σε ένα αρχείο στο σύστημα αρχείων των Windows (c:\ubuntu\disks\root.disk), σε αντίθεση με την εγκατάσταση στο δικό του διαμέρισμα. Αυτό το αρχείο είναι ορατό από το Linux ως ένας πραγματικός σκληρός δίσκος. Το Wubi δημιουργεί επίσης ένα αρχείο εικονικής μνήμης στο σύστημα αρχείων των Windows (c:\ubuntu\disks\swap.disk). Αυτό το αρχείο είναι ορατό από το Ubuntu ως πρόσθετη μνήμη RAM.

3.3 Πακέτο multicore του octave

Το πακέτο αυτό επιτρέπει στον χρήστη του Octave να υπολογίσει μία συνάρτηση παράλληλα χρησιμοποιώντας διάφορες διεργασίες.

3.3.1 Συναρτήσεις που περιέχονται στο πακέτο Multicore

Χρήση διάφορων διεργασιών με σκοπό την παράλληλη εκτέλεση ενός κώδικα με τη χρήση του πακέτου Multicore πραγματοποιείται με τη χρήση των κατάλληλων συναρτήσεων. Οι συναρτήσεις αυτές είναι οι εξής:

startmulticoremaster.m

Λαμβάνει τη συνάρτηση και τις παραμέτρους της και υπολογίζει το αποτέλεσμα. Τα αρχεία που περιέχουν πληροφορίες σχετικά με το ποια συνάρτηση είναι για να εκτελεστεί και με ποιες παραμέτρους, αποθηκεύονται σε ένα φάκελο και διαβάζονται από τις διεργασίες εργάτες. Τα αποτελέσματα

σώζονται από τη διεργασία εργάτη με σκοπό να διαβαστούν αργότερα.

startmulticoreslave.m

Φορτώνει τα αρχεία που δημιουργούνται από την συνάρτηση startmulticoremaster.m και αξιολογεί τη συγκεκριμένη συνάρτηση συνάρτηση. Τα αποτελέσματα σώζονται σε αρχεία στον ίδιο κατάλογο

setfilesemaphore.m , removefilesemaphore.m , **deletewithsemaphores.m**

Είναι πολύ σημαντικό να αποφευχθεί το γεγονός ότι διάφορες διεργασίες MATLAB μπορεί να προσπαθήσουν να ανοίξουν/διαγράψουν/ εγγράψουν σε ένα αρχείο ταυτόχρονα. Χρησιμοποιώντας μια απλή τεχνική semaphore, η αποκλειστική πρόσβαση σε ένα αρχείο είναι εγγυημένη

existfile.m, existfile.c

Έλεγχος εάν ένα αρχείο υπάρχει. Για τη χρήση του ταχύτερου αρχείου MEX,

ο χρήστης μπορεί να πληκτρολογήσει "MEX-setup", να επιλέξει τον builtin Lcc μεταγλωτιστή και να πληκτρολογήσει "mex existfile.c" για τη μεταγλώττιση του αρχείου. Ωστόσο, το πακέτο λειτουργεί και χωρίς τη χρήση του αρχείου MEX.

findfiles.m

Επιστρέφει μία λίστα από ονόματα αρχείων που ταιριάζουν με
μια συγκεκριμένη προδιαγραφή σε έναν πίνακα

getusername.m

Επιστρέφει το όνομα του χρήστη.

tempdir2.m

Επιστρέφει το όνομα του προσωρινού καταλόγου.

3.4 Εκτέλεση συναρτήσεων παράλληλα με το πακέτο multicore

Στο πακέτο multicore του Octave οι συναρτήσεις τις οποίες ο χρήστης μπορεί να εκτελέσει είναι δυο:

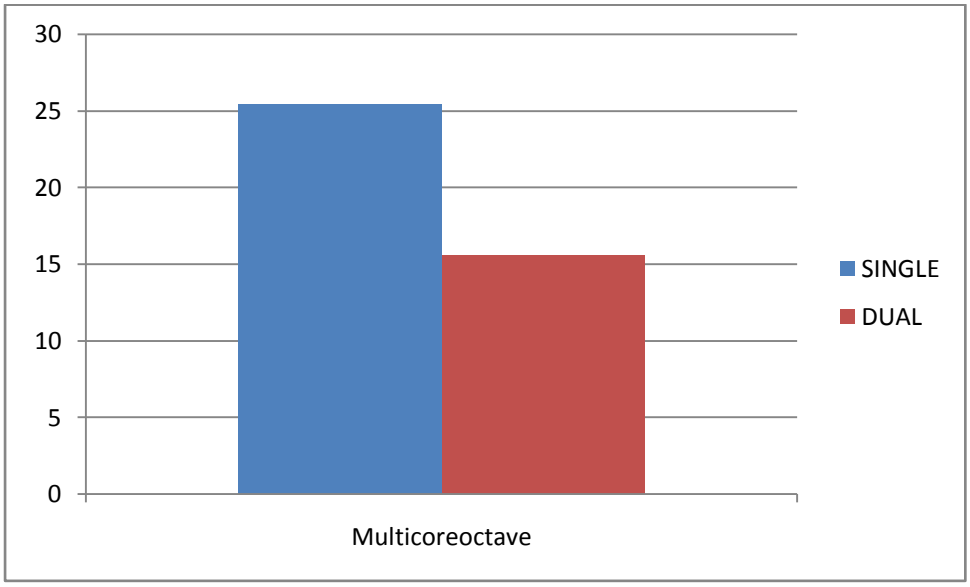
1. Multicoredemo
2. Multicoreoctave

3.4.1 Εκτέλεση συνάρτησης Multicoreoctave

Εικόνα 8 Εκτέλεση συνάρτησης multicoreoctave

Όπως βλέπουμε και στην παραπάνω εικόνα υπάρχει μεγάλη διάφορα στον χρόνο εκτέλεσης του αλγορίθμου όταν εκτελείται σε ένα πυρήνα και όταν εκτελείται σε δυο πυρήνες. Ο παρακάτω πινάκας και το αντίστοιχο διάγραμμα θα μας βοηθήσουν να καταλάβουμε καλυτέρα την διάφορα του χρόνου από τις δυο εκτελέσεις του αλγορίθμου

	ΕΚΤΕΛΕΣΗ ΣΕ ΈΝΑ ΠΥΡΗΝΑ	ΕΚΤΕΛΕΣΗ ΣΕ ΔΥΟ ΠΥΡΗΝΕΣ
ΧΡΟΝΟΙ	25,4097	15,55



Εικόνα 9 Διάγραμμα χρόνου εκτέλεσης

Από τι βλέπουμε και στο σχεδιάγραμμα όταν ο αλγόριθμος εκτελεστικέ παράλληλα σε δύο πυρήνες ο χρόνος εκτέλεσης του μειώθηκε περίπου κατά 40%.

3.5 Κατανεμημένη (Distribute) επεξεργασία στο octave

Στην επιστήμη υπολογιστών παράλληλα, κατανεμημένα ή ταυτόχρονα συστήματα ονομάζονται υπολογιστές οι οποίοι επιτρέπουν την ταυτόχρονη εκτέλεση πολλαπλών συνεργαζόμενων προγραμμάτων σε μία ή περισσότερες επεξεργαστικές μονάδες. Για να επετεύχθη αυτό χρειάζονται δυο πράγματα:

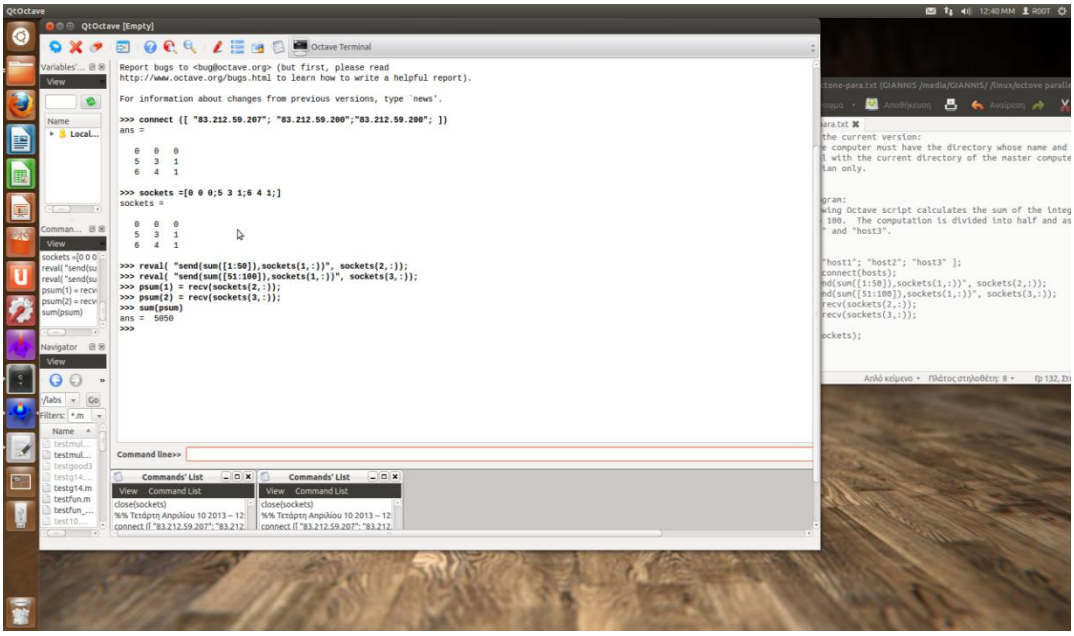
1. Να έχουμε ένα δίκτυο δυο ή περισσότερων υπολογιστών
2. Να έχουν όλοι οι υπολογιστές εγκατεστημένο το Octave

Σε αυτή την εργασία δοκίμασα να συνδέσω μεταξύ τους τρεις υπολογιστές σε ένα τοπικό δίκτυο και αφού έκανα εγκατάσταση και στους τρεις το Octave δοκίμασα να εκτελέσω έναν απλό αλγόριθμο αθροίσματος. Η συλλογιστική ήταν ότι ο ένας από τους τρεις υπολογιστές να θεωρούνταν ο κεντρικός(server), και οι άλλοι δυο(slaves) θα εκτελούσαν τον αλγόριθμο και θα επέστρεφαν τα αποτελέσματα στον κεντρικό(server) υπολογιστή

Ο αλγόριθμος ήταν ο εξής:

```
hosts = [ "host1"; "host2"; "host3" ];
sockets = connect(hosts);
reval( "send(sum([1:50]),sockets(1,:))", sockets(2,:));
reval( "send(sum([51:100]),sockets(1,:))", sockets(3,:));
psum(1) = recv(sockets(2,:));
psum(2) = recv(sockets(3,:));
sum(psum)
```

Αυτός ο αλγόριθμος αφού συνδέσει τους υπολογιστές μεταξύ τους στέλνει στον έναν υπολογιστή να υπολογίσει το άθροισμα από 1 έως 50 έπειτα στέλνει στον δεύτερο υπολογιστή να υπολογίσει το άθροισμα από 51 έως 100 και τέλος ο κεντρικός υπολογιστής(server) λαμβάνει τα αποτελέσματα και εμφανίζει το τελικό άθροισμα



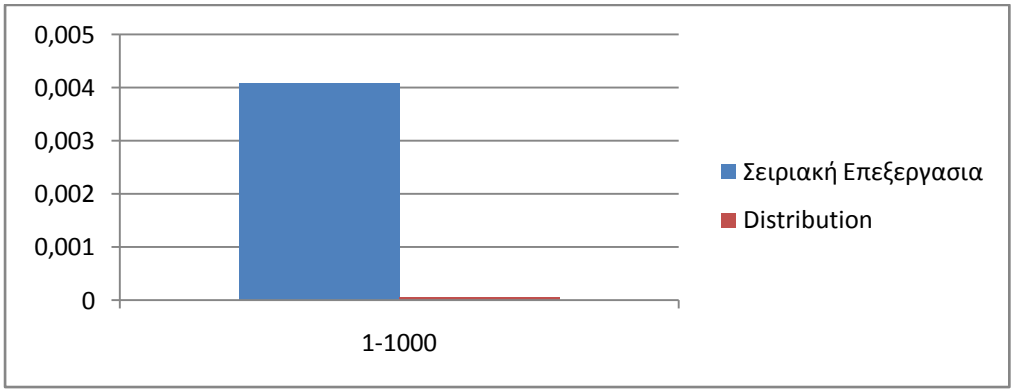
Εικόνα 10 Εκτέλεση αλγορίθμου αθροίσματος στο Octave

Έπειτα για να εξετάσουμε αν όντως αποδοτική αυτή η μορφή επεξεργασίας βάζαμε σταδιακά να υπολογίζει ο αλγόριθμος όλο και μεγαλύτερα αθροίσματα και καταγράφαμε τους χρόνους. Τα αποτελέσματα φαίνονται στον παρακάτω πινάκα και γραφήματα.

Εύρος αθροισμάτων	**1-1000**	**1-10000**	**1-100000**	**1-1000000**	**1-5000000**	**1-10000000**
Χρόνος κατανεμημένης επεξεργασίας	0,000072002	0,000063896	0,00006294	0,00006509	0,0000701	0,0007105
Χρόνος σειριακής επεξεργασίας	0,004087397	0,000055074	0,0002172	0,002481	0,01265	0,02104

Γραφήματα

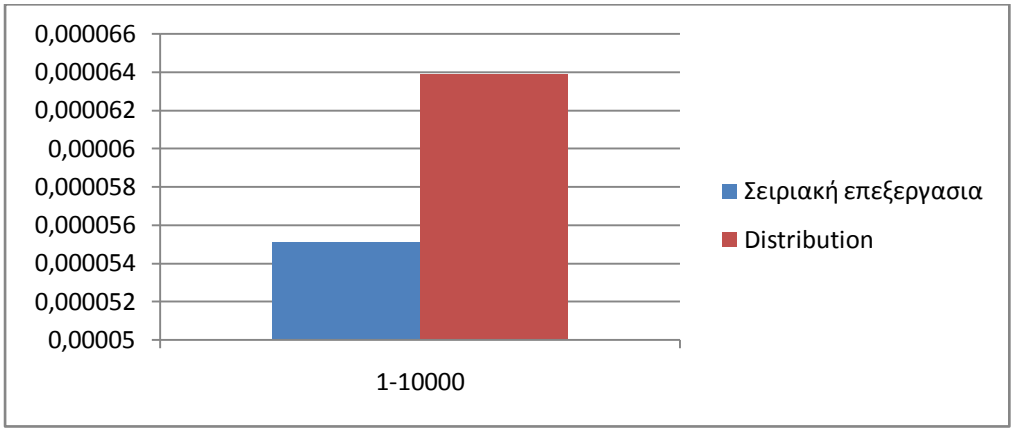
Στο πρώτο γράφημα βλέπουμε τον αλγόριθμο να εκτελείται από το 1 έως το 1000 και παρατηρούμε ότι η διάφορα ανάμεσα στην κατανεμημένη επεξεργασία και στην σειριακή είναι τεραστία



Εικόνα 11 Διάγραμμα χρόνου εκτέλεσης(1-1000)

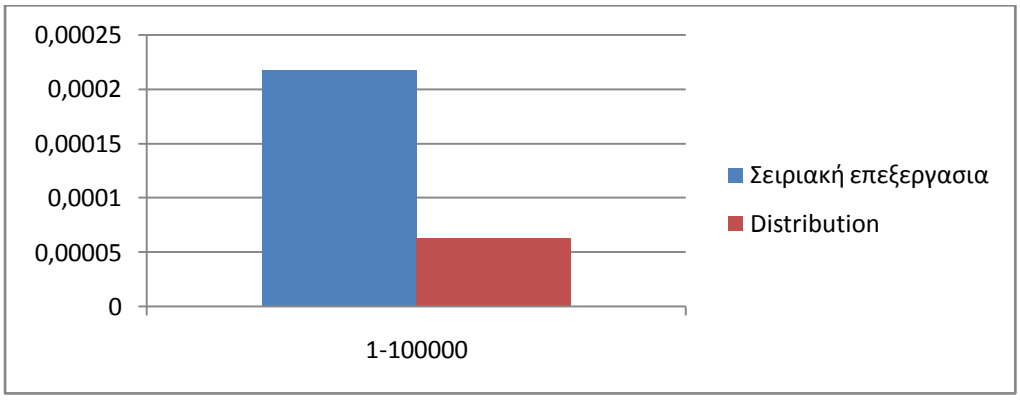
Στο δεύτερο γράφημα εκτελούμε τον αλγόριθμο για το εύρος τιμών απο 1 έως 10000. Παραδόξως παρατηρούμε ότι με την μέθοδο της κατανεμημένης επεξεργασίας ο αλγόριθμος εκτελέστηκε πιο αργά από τι στην σειριακή επεξεργασία. Αυτό οφείλεται σε παράγοντες του τοπικού δικτύου και είναι ένα πολύ σημαντικό μειονέκτημα για την κατανεμημένη

επεξεργασία, αλλά όπως θα δούμε και στα παρακάτω γραφήματα το γεγονός αυτό δεν συμβάλει στο τελικό συμπέρασμα το όποιο είναι ότι με την κατανεμημένη επεξεργασία οι χρόνοι για να εκτελεστεί ένας αλγόριθμος είναι σαφώς μικρότεροι και με τεραστία διάφορα μάλιστα.

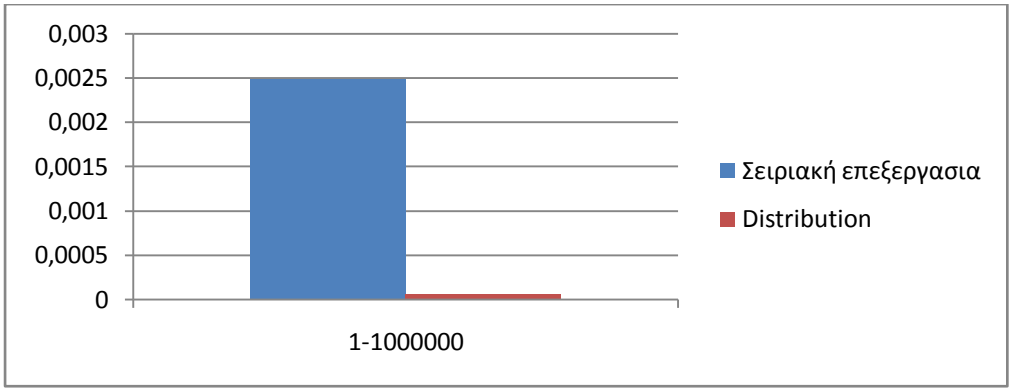


Εικόνα 12 Διάγραμμα χρόνου εκτέλεσης(1-10000)

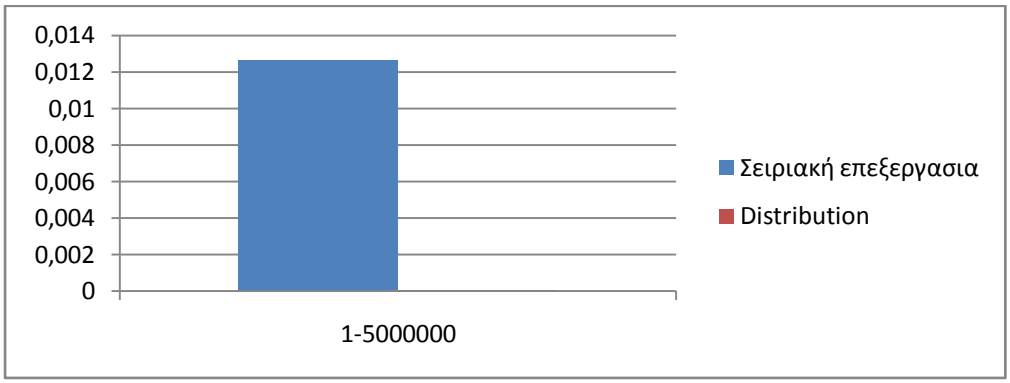
Στο τρίτο διάγραμμα όπως και στα υπόλοιπα τα αποτελέσματα είναι όπως προείπα αυτά που περιμέναμε, δηλαδή με την μέθοδο της κατανεμημένης επεξεργασίας ο χρόνος εκτέλεσης του αλγορίθμου είναι πολύ μικρότερος



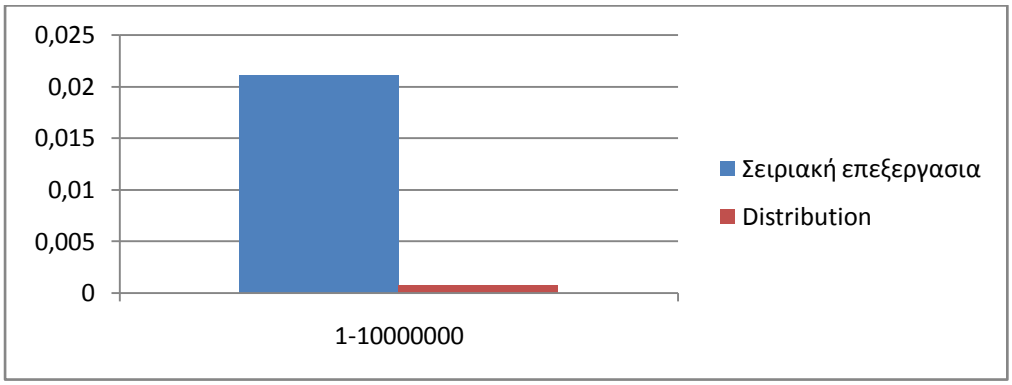
Εικόνα 13 Διάγραμμα χρόνου εκτέλεσης(1-100000)



Εικόνα 14 Διάγραμμα χρόνου εκτέλεσης(1-1000000)



Εικόνα 15 Διάγραμμα χρόνου εκτέλεσης(1-5000000)



Εικόνα 16 Διάγραμμα χρόνου εκτέλεσης(1-10000000)

Συμπερασματικά δεν μπορούμε να ορίσουμε ποσό ακριβώς είναι η μείωση του χρόνου της κατανεμημένης επεξεργασίας σε σύγκριση με την σειριακή επεξεργασία διότι οι παράγοντες του δικτύου δεν μας το επιτρέπουν. Το γεγονός αυτού του πειράματος όμως είναι ότι με την μέθοδο της κατανεμημένης επεξεργασίας ο χρόνος εκτέλεσης ενός αλγορίθμου είναι πολύ μικρότερος.

ΚΕΦΑΛΑΙΟ 4

Οδηγός εγκατάστασης octave και εκτέλεσης αλγορίθμων

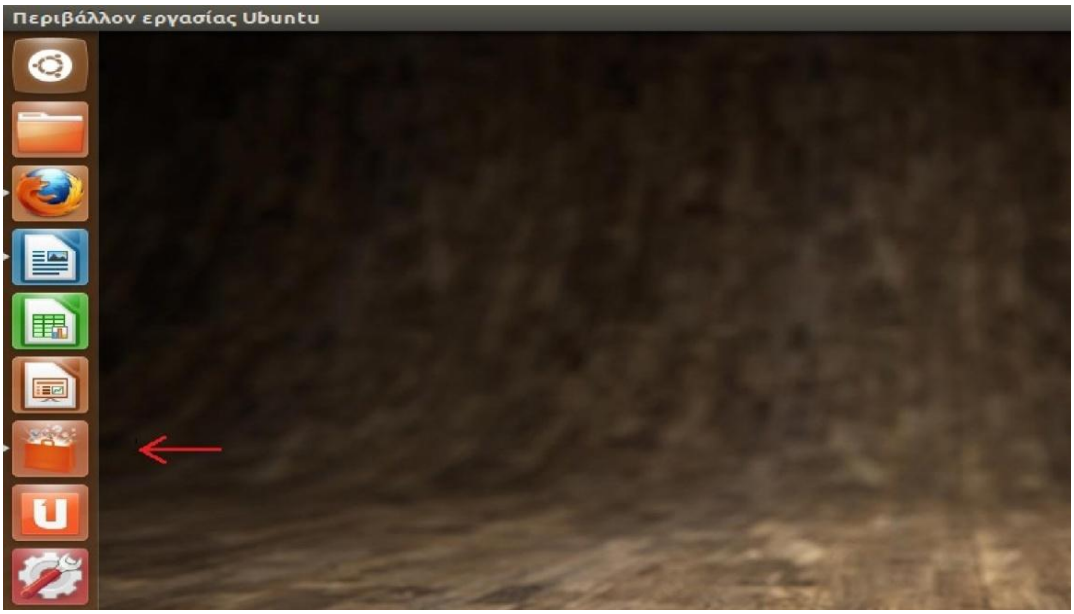
4.1 Οδηγός εγκατάστασης του Octave - εκτέλεση αλγορίθμων

Η εγκατάσταση του λογισμικού Octave μπορεί να γίνει με δυο τρόπους.

ΠΡΩΤΟΣ ΤΡΟΠΟΣ

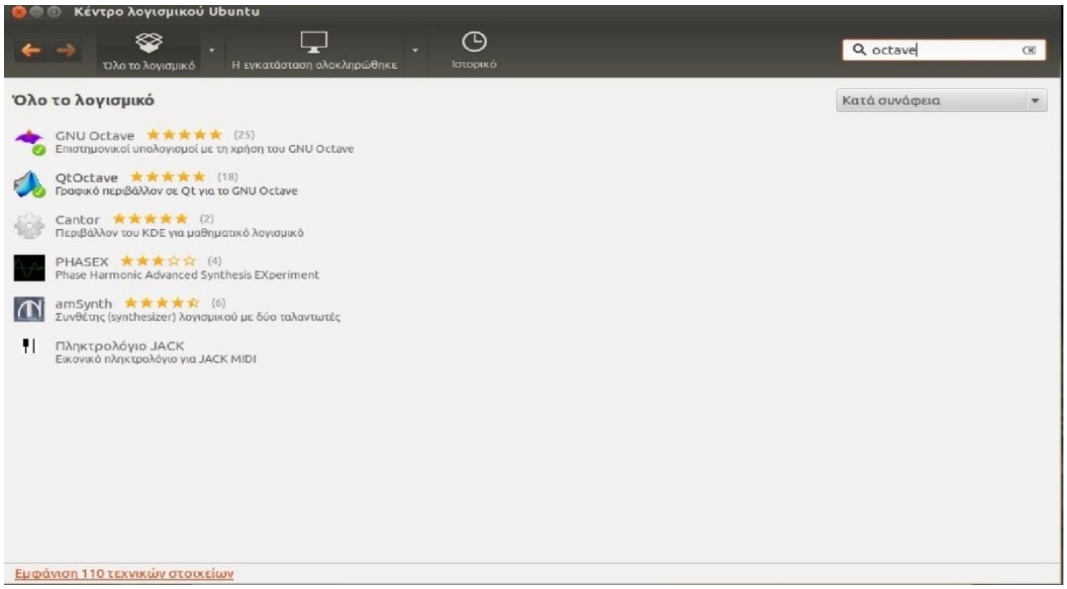
Η έκδοση Ubuntu 12.04 μας προσφέρει ένα πολύ χρήσιμο εργαλείο το οποίο ονομάζεται Κέντρο Λογισμικού Ubuntu.

Πατώντας το εικονίδιο που βλέπουμε στην φωτογραφία μας εμφανίζεται η κεντρική σελίδα του Κέντρο Λογισμικού.



Εικόνα 17 Επιφάνεια εργασίας Ubuntu 12.4

Άμα πληκτρολογήσουμε στην μπάρα αναζήτησης πάνω δεξιά την λέξη octave το πρόγραμμα μας εμφανίζει το πρόγραμμα καθώς και ότι άλλες επεκτάσεις το αφορούν



Εικόνα 18 Κέντρο λογισμικού Ubuntu

Έπειτα πατάμε στο κουμπί "Εγκατάσταση" και έτσι το octave εγκαταστάθηκε εύκολα και γρήγορα στον υπολογιστή μας

Όπως βλέπουμε και στην παρακάτω εικόνα θα χρειαστούμε και το QtOctave το οποίο είναι το γραφικό περιβάλλον του προγράμματος. Μέτα την εγκατάσταση παρατηρούμε ότι έχουν δημιουργηθεί και τα 2 εικονίδια του Octave και είναι έτοιμα για χρήση

Εικόνα 19 Επιφάνεια εργασίας Ubuntu 12.4 μετα την εγκατάσταση Octave

ΔΕΥΤΕΡΟΣ ΤΡΟΠΟΣ

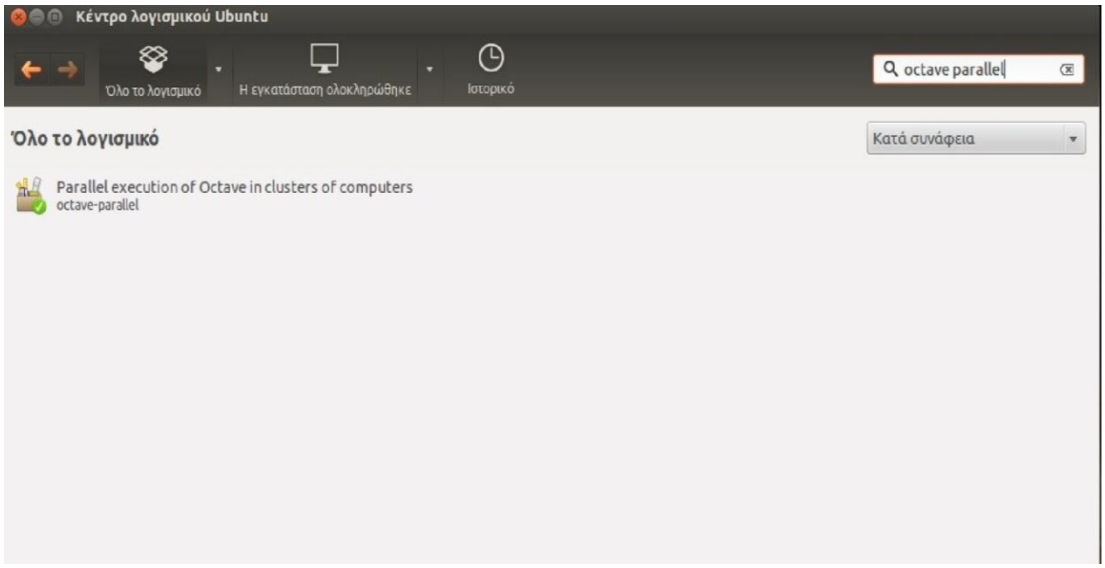
Ένας δεύτερος τρόπος που θα μπορούσαμε να κάνουμε εγκατάσταση το πρόγραμμα μας θα ήταν να μεταβούμε στην σελίδα του Octave (http://www.gnu.org/software/octave/ ή http://octave.sourceforge.net/) και να κατεβάσουμε την έκδοση του Octave που θέλουμε
Έπειτα πρέπει να ανοίξουμε το τερματικό και να δώσουμε την εξής εντολή

pkg install file name.tar.gz

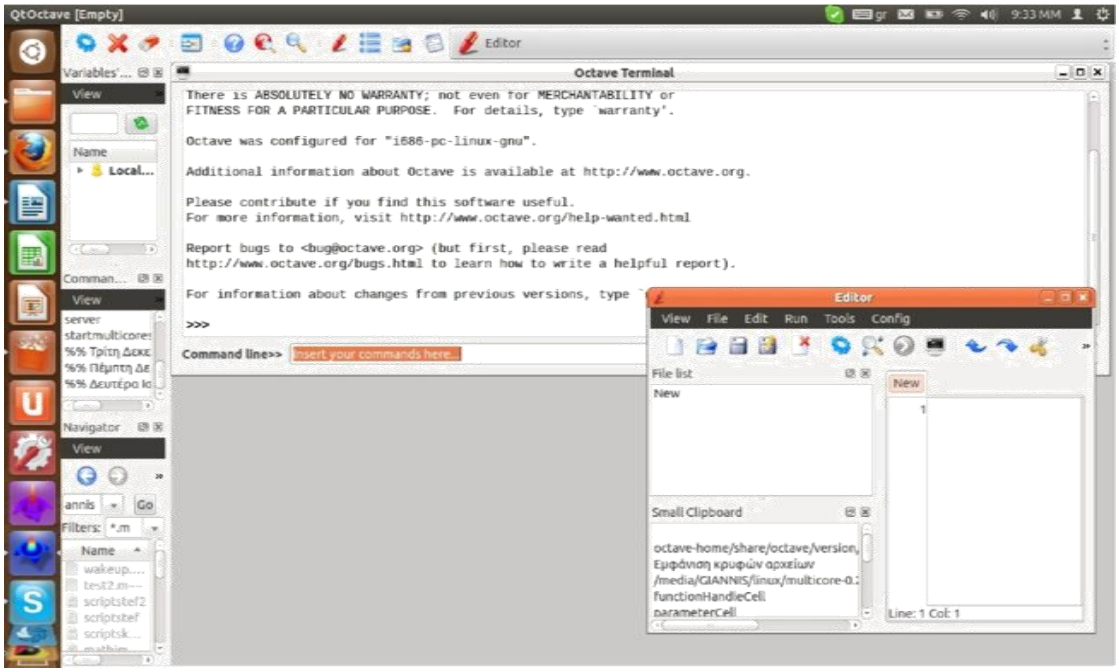
4.2 Οδηγός εκτέλεσης παράλληλων αλγορίθμων

Για να εκτελέσουμε παράλληλους αλγορίθμους μεσώ του Gnu Octave θα πρέπει πρώτα να κατεβάσουμε κάποια επιπλέον πακέτα που χρειάζονται. Ο τρόπος και τα πακέτα που χρειάζονται φαίνονται και αναγράφονται παρακάτω

Αρχικά μέσω του Κέντρου λογισμικού και γράφοντας στην μπάρα αναζήτησης “octave parallel” κατεβάζουμε και εγκαθιστούμε το parallel execution of Octave in clusters of computers όπως φαίνεται και στην παρακάτω εικόνα



Εικόνα 20 Κέντρο λογισμικού Ubuntu12.4



Εικόνα 21 Περιβάλλον Octave

Έπειτα κάνουμε κλικ στο εικονίδιο QtOctave και μπαίνουμε στο γραφικό περιβάλλον του προγράμματος που μπορούμε να δούμε το παράθυρο εντολών του προγράμματος καθώς και τον Editor του Octave

4.3 Πακέτα multicore και socket

Σε αυτό το σημείο θα χρειαστούμε δύο επιπλέον πακέτα τα οποία μπορούμε να τα κατεβάσουμε μεσώ του Octave ή από την επίσημη σελίδα του(http://octave.sourceforge.net) και έπειτα θα είμαστε έτοιμοι να εκτελέσουμε τους αλγορίθμους μας.

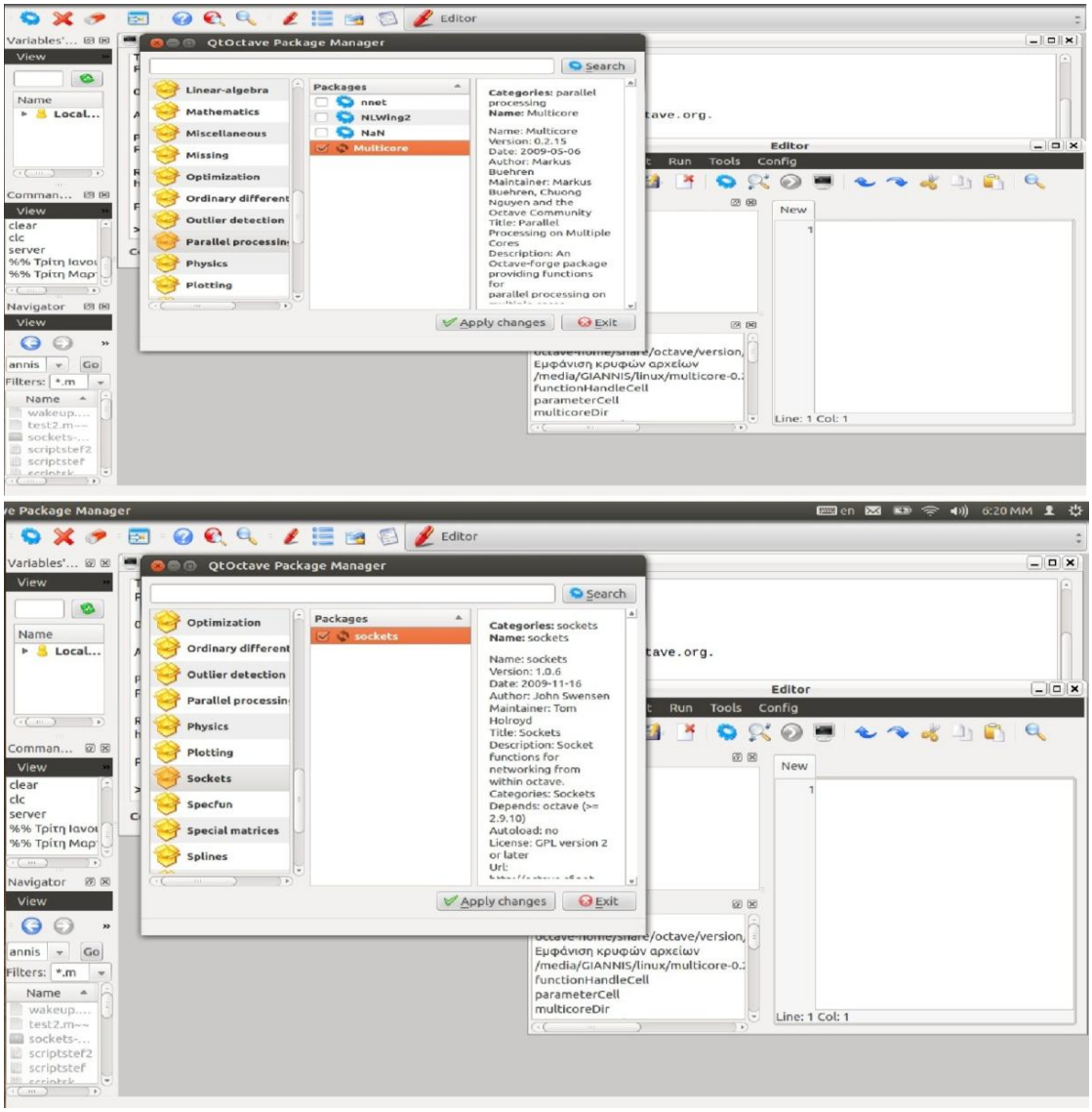
Τα πακέτα είναι τα εξής

1.Multicore(http://octave.sourceforge.net/multicore/index.html)

2.Sockets(http://octave.sourceforge.net/sockets/index.htm)

4.3.1 Εγκατάστασης μέσω Octave

Αρχικά κάνουμε κλικ στην επιλογή **Config** και έπειτα **Install Octave Packages** Βρίσκουμε τα πακέτα που θέλουμε , κάνουμε διπλό κλικ πάνω τους και έπειτα πατάμε την επιλογή Apply Changes όπως φαίνεται στην παρακάτω εικόνα



Εικόνα 22 Εγκατάσταση πακετων Multicore και Sochets

4.3.2 Εγκατάσταση μέσω Ιστοσελίδας

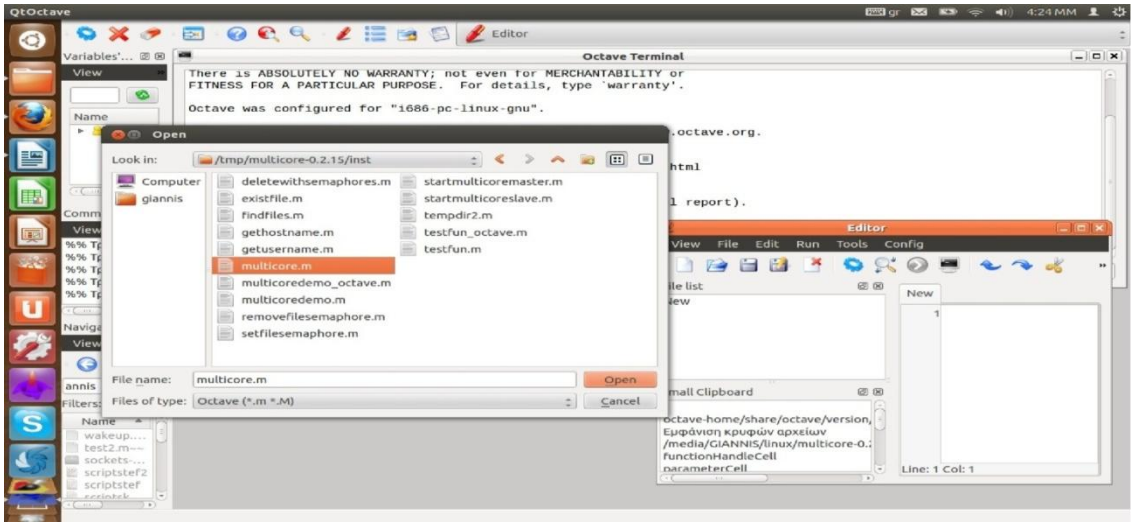
Η διαδικασία είναι πολύ απλή .Μεταβαίνουμε στους συνδέσμους που αναγράφονται παραπάνω και επιλέγουμε να τα κατεβάσουμε .Αφού κατεβούν τα πακέτα δεν χρειάζεται να κάνουμε τίποτα άλλο πάρα μόνο να τα αποσυμπιέσουμε.

4.4 Εκτέλεση αλγορίθμων με το πακετο Multicore

Τέλος το μόνο που μένει να κάνουμε είναι να φορτώσουμε τα script αρχεία από το πακέτο multicore που κατεβάσαμε μεσώ του Editor και να τα εκτελέσουμε.Ηδιαδικασία

φαίνεται παρακάτω

Επιλέγουμε File->Open στο μενού του Editor ,βρίσκουμε τον φάκελο multicore που κατεβάσαμε ,μπαίνουμε σε αυτόν και έπειτα μεταβαίνουμε στον υποφακελο inst. Εκεί θα βρούμε όλες τις απαραίτητες συναρτήσεις που θα χρειαστούμε για να εκτελέσουμε παράλληλους αλγορίθμους



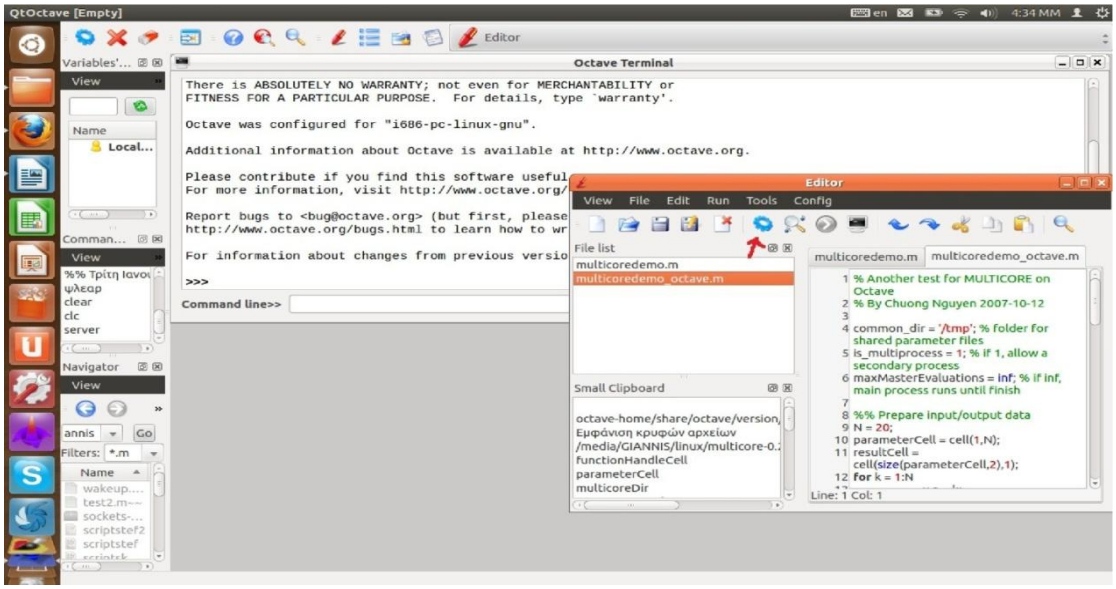
Εικόνα 23 Άνοιγμα και εκτέλεση συνάρτησης

Οι συναρτήσεις που μπορούμε να τρέξουμε στο πακέτο multicore ειναι δυο:

1.multicore_octave

2.multicoredemo

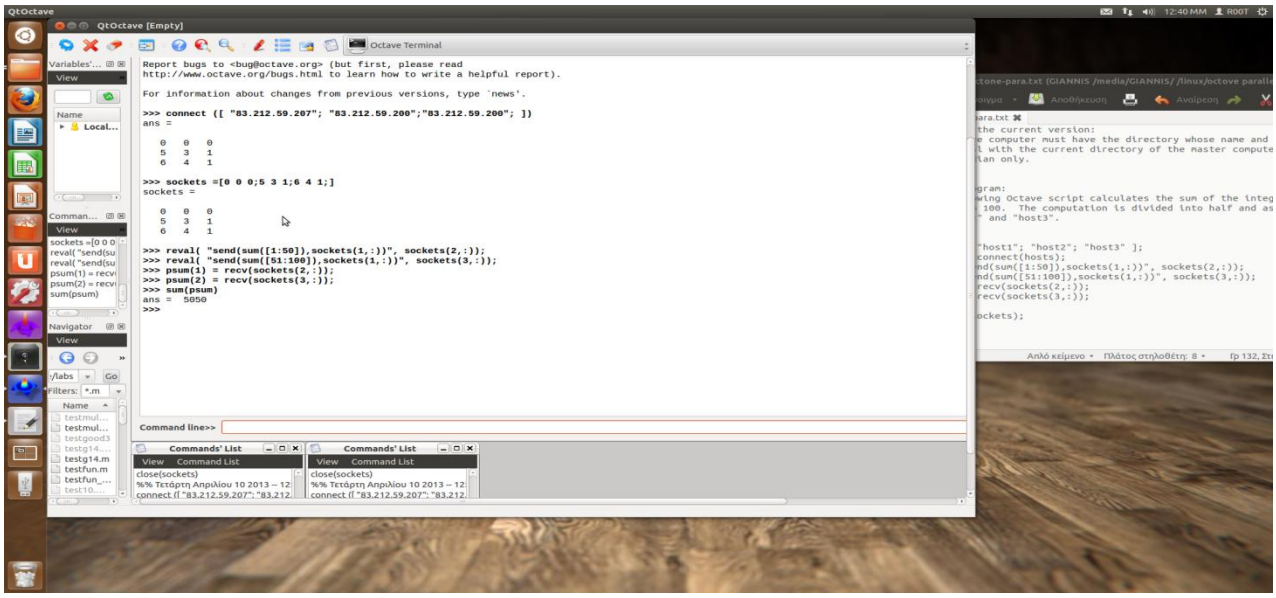
Αφού τις ανοίξουμε με τον Editor το μόνο που μένει να κάνουμε είναι να πατήσουμε το κουμπί εκτέλεσεις



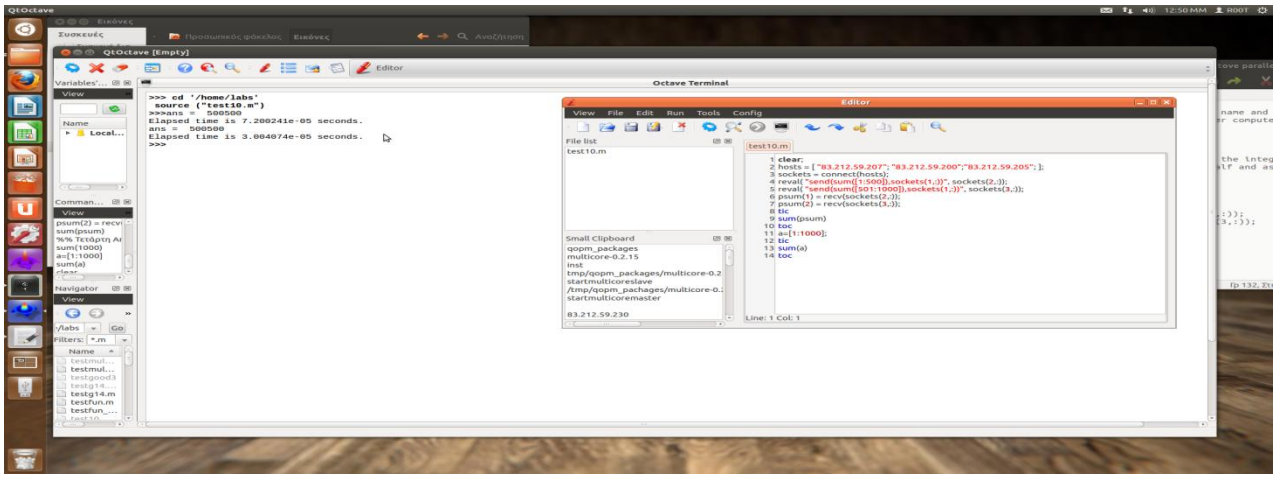
Εικόνα 24 Εκτέλεση συνάρτησης

4.5Εκτέλεση αλγορίθμων κατανεμημένης επεξεργασίας

Για να εκτελέσουμε τον αλγόριθμο του αθροίσματος που αναφέραμε παραπάνω το μόνο που αρκεί να κάνουμε είναι να γράψουμε τις εντολές στο τερματικό του προγράμματος και αυτές θα εκτελεστούν ή θα μπορούσαμε να γράψουμε ολόκληρο το πρόγραμμα στον Editor του Octave και να το εκτελέσουμε από κει. Και οι δύο τρόποι φαίνονται αντίστοιχα στις παρακάτω εικόνες



Εικόνα 25 Εκτέλεση αλγορίθμου αθροίσματος



Εικόνα 26 Εκτέλεση αλγορίθμου αθροίσματος μέσου Editor Octave

ΚΕΦΑΛΑΙΟ 5

ΣΥΜΠΕΡΑΣΜΑΤΑ

Συμπερασματικά, αυτό που προσπαθούμε κατά βάση να πετύχουμε με τη παράλληλη επεξεργασία είναι να κερδίσουμε χρόνο ή και να επεξεργαστούμε μεγαλύτερο μέγεθος δεδομένων. Το υλικό γίνεται συνεχώς φθηνότερο και ισχυρότερο, το λογισμικό είναι γενικά δημόσια διαθέσιμο και βελτιώνεται συνεχώς, ο όγκος των δεδομένων αυξάνεται με αλματώδεις ρυθμούς και λόγω διαδικτύου, ο διαθέσιμος χρόνος μας παραμένει ίδιος. Η παράλληλη επεξεργασία προσπαθεί να εκμεταλλευτεί όσο γίνεται πιο αποδοτικά το άφθονο υλικό, έτσι ώστε με τη βοήθεια ειδικού λογισμικού να επιτύχει μέγιστη επεξεργασία δεδομένων στον ελάχιστο χρόνο. Σε αυτή την πτυχιακή με την βοήθεια του λογισμικού Octave καταφέραμε να δείξουμε και επιβεβαιώσαμε ότι με την παράλληλη επεξεργασία και με τους παράλληλους αλγορίθμους κερδίζουμε πολύτιμο χρόνο στην εκτέλεση ενός αλγορίθμου όπως επίσης καταφέραμε να αξιοποιήσουμε αποδοτικά τους υπολογιστές που είχαμε στην διάθεση μας έτσι ώστε να επιτύχουμε την παράλληλη επεξεργασία δεδομένων με αποτέλεσμα την μειώσει του χρόνο εκτέλεσης.

Βιβλιογραφία

1. http://whatis.techtarget.com/definition/parallel-processing-software
2. http://www.it.uom.gr/project/parallel/
3. https://computing.llnl.gov/tutorials/parallel_comp/#WhyUse
4. http://www.aoki.ecei.tohoku.ac.jp/octave/patch/README.parallel
5. http://www.gnu.org/software/octave/doc/interpreter/
6. http://pdplab.it.uom.gr/teaching/llnl-gr/Introduction%20to%20Parallel%20Computing.htm
7. https://www.scilab.org/
8. http://en.wikipedia.org/wiki/Apache_Hadoop
9. http://www.mathworks.com/products/matlab/
10. http://en.wikipedia.org/wiki/Parallel_computing
11. http://en.wikipedia.org/wiki/GNU_Octave