ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Αρχιτεκτονική Πρακτόρων για την Υποστήριξη Κατανεμημένων Υπηρεσιών Επεξεργασίας Δεδομένων

Εργασία που υποβλήθηκε από τον

Μάριο Κ. Ζήκο

Ως μερική απαίτηση για την απόκτηση του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΞΕΙΔΙΚΕΥΣΗΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Αρχιτεκτονική Πρακτόρων για την Υποστήριξη Κατανεμημένων Υπηρεσιών Επεξεργασίας Δεδομένων

Εργασία που υποβλήθηκε από τον
Μάριο Κ. Ζήκο
Ως μερική απαίτηση για την απόκτηση του
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΞΕΙΔΙΚΕΥΣΗΣ

Οκτώβριος 1997

Συγγραφέας:

Εισηγητική Επιτροπή:

Στέλιος Ορφανουδάκης, Καθηγητής, Επόπτης

Απόστολος Τραγανίτης, Αναπληρωτής Καθηγητής, Μέλος

Πάνος Τραχανιάς, Επίκουρος Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος, Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Αρχιτεκτονική Πρακτόρων για την Υποστήριξη Κατανεμημένων Υπηρεσιών Επεξεργασίας Δεδομένων

Μάριος Κ. Ζήκος

Πανεπιστήμιο Κρήτης

Σχολή Θετικών Επιστημών

Τμήμα Επιστήμης Υπολογιστών

Ηράκλειο, Οκτώβρης, 1997

Αρχιτεκτονική Πρακτόρων για την Υποστήριξη Κατανεμημένων Υπηρεσιών Επεξεργασίας Δεδομένων

Μάριος Ζήκος

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών

Πανεπιστήμιο Κρήτης

ΠΕΡΙΛΗΨΗ

Η παρούσα διατριβή μελετά την παροχή υπηρεσιών επεξεργασίας δεδομένων μέσω ενός δικτύου υπολογιστών και προτείνει μία αρθρωτή, ευέλικτη και επεκτάσιμη αρχιτεκτονική η οποία μπορεί να αποτελέσει τον κορμό για την ανάπτυξη συστημάτων για την παροχή υπηρεσιών επεξεργασίας δεδομένων. Η βάση της αρχιτεκτονικής είναι μια κοινωνία κατανεμημένων αυτόνομων πρακτόρων λογισμικού, που συνεργάζονται για την εκτέλεση διεργασιών επεξεργασίας σε ένα κατανεμημένο περιβάλλον. Η εργασία προτείνει ακόμα έναν κατανεμημένο μηχανισμό διαχείρισης πόρων με βάση μηχανισμούς αγοράς, ο οποίος μελετάται με τη βοήθεια προσομοιωτή.

Η προτεινόμενη αρχιτεκτονική χρησιμοποιείται για την ανάπτυξη μιας ειδικευμένης εφαρμογής, ενός περιβάλλοντος παροχής υπηρεσιών επεξεργασίας εικόνων, του οποίου η κωδική ονομασία είναι DIPE (Distributed Image Processing Environment). Το περιβάλλον αυτό στοχεύει στο να προσφέρει στον χρήστη, είτε αυτός είναι ένας εξειδικευμένος ερευνητής είτε είναι ένας απλός χρήστης, τη δυνατότητα επεξεργασίας εικόνων χρησιμοποιώντας με διαφάνεια (ως προς την διεύθυνση, πρόσβαση και εκτέλεση) την υπολογιστική ισχύ και τους αλγόριθμους επεξεργασίας ενός ευρύτερου δικτύου. Επιπλέον, αποσκοπεί στο να παρέχει εύκολη και ενιαία πρόσβαση σε διάφορους μεμονωμένους αλγόριθμους επεξεργασίας εικόνων, δίνοντας μηχανισμούς για την εύκολη ενσωμάτωση

νέων αλγόριθμων καθώς και αλγόριθμων που έχουν αναπτυχθεί στο παρελθόν από τρίτους (δηλ. δεν υπάρχει πρόσβαση στον πηγαίο κώδικα). Το περιβάλλον παρέχει μηχανισμούς για την βέλτιστη χρήση των διαθέσιμων πόρων, ενώ η ποιότητα των προσφερόμενων υπηρεσιών μπορεί να διακυμανθεί, ανάλογα με το (εικονικό) ποσό που ο χρήστης είναι διατεθειμένος να πληρώσει για κάθε χρήση.

Απώτερος σκοπός της εργασίας είναι να γενικευθούν οι προτεινόμενες στο DIPE λύσεις και επιλογές και, χρησιμοποιώντας αφαιρετικές τεχνικές και μηχανισμούς για την ανεξαρτητοποίηση της αρχιτεκτονικής από την χρήση της σε έναν χώρο εφαρμογών, να προταθούν τρόποι και βήματα για μελλοντική δουλειά με την οποία μπορούμε εύκολα να φτάσουμε στην κατασκευή ενός πλαισίου εργασίας (framework) για την εύκολη και άμεση ανάπτυξη εφαρμογών παροχής υπηρεσιών επεξεργασίας δεδομένων, ανεξαρτήτως χώρου εφαρμογής.

Επόπτης: Στέλιος Ορφανουδάκης

Καθηγητής Επιστήμης Τπολογιστών

Πανεπιστήμιο Κρήτης

Agent Based Architecture for the Support of Distributed Information Processing Services

Marios Zikos

Master of Science Thesis

Department of Computer Science

University of Crete

ABSTRACT

This thesis is intended to explore the field of information processing services over computer networks. A modular, flexible and extensible architecture is proposed, that can be used for the development of environments that support information processing services. This architecture is based on a society of distributed autonomous agents that cooperate in order to execute processing algorithms in a distributed environment. This thesis also proposes a distributed resource management mechanism, which is based on the market metaphor and is studied through simulation experiments.

The proposed architecture is used for the development of a specific application, an environment for supporting image processing services, DIPE (Distributed Image Processing Environment). DIPE aims to support specialized researchers in the field of image processing as well as common end users, and offers the ability of transparent algorithm execution through the exploitation of computational power and collections of algorithms available in a network. In addition, the proposed environment aims to become the integration platform for various image processing algorithms, either new or developed by third parties (in which case there is no access to their source code). DIPE also offers resource management mechanisms, while quality of service depends on the (virtual) cost the user is prepared to pay for each execution.

The ultimate goal of this work is to generalize the design and implementation strategies proposed in DIPE, and lead towards the future development of a framework that can readily support implementation of environments for the provision of information processing services irrespective of the application domain.

Supervisor: Stelios Orphanoudakis

Professor of Computer Science

University of Crete

Ευχαριστίες

Καταρχήν θέλω να αναγνωρίσω την σημαντική συμβολή του Τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης, καθώς και του Ινστιτούτου Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας, το οποίο μου παρείχε τους απαραίτητους πόρους και τον απαιτούμενο εξοπλισμό για τη διεκπεραίωση της παρούσας εργασίας.

Θα ήθελα επίσης να ευχαριστήσω:

Τον επόπτη και καθηγητή μου, κ. Στέλιο Ορφανουδάκη, που με την καθοδήγησή του και τις σημαντικές παρεμβάσεις του βοήθησε στο να πάρει μορφή η παρούσα εργασία.

Την Ελένη Καλδούδη για την πολύ σημαντική βοήθεια στη σύλληψη της ιδέας, στο σχεδιασμό του περιβάλλοντος και στη συνεχή υποστήριξη της κατά τη διάρκεια της ανάπτυξης του. Ένα μεγάλο ευχαριστώ για τις κατά καιρούς διορθώσεις και εποικοδομητικές παρατηρήσεις σε όλα τα σχετικά με την εργασία κείμενα.

Τον Πάνο Τραχανιά που ήταν θερμός υποστηρικτής της εργασίας από την αρχή της (…χωρίς να πτοείται καθόλου από τις αναπόφευκτες ενδιάμεσες αποτυχίες). Τον ευχαριστώ ακόμα που μου έδωσε τη δυνατότητα να παρευρεθώ σε δύο πολύ σημαντικά διεθνή συνέδρια, που με βοήθησαν ουσιαστικά τόσο με τις εμπειρίες και τις γνώσεις που μου έδωσαν.

Τον δάσκαλό μου, Απόστολο Τραγανίτη που τόσα πολλά του οφείλω στα χρόνια της παρουσίας μου στο Πανεπιστήμιο Κρήτης.

Την παλιό παρέα: Γιώργο Αγγελόπουλο, Γιώργο Βερνάρδο, Πάνο Γκυκόπουλο, Κοσμά Πετρίδη, Νίκο Χαρδαβέλλα, Γιάννη Χειμαριώτη, που για αρκετά χρόνια με ατέλειωτες συζητήσεις και ξενύχτια πιστεύω ότι βοήθησαν πολύ σημαντικά στην αύξηση της κατανάλωσης οινοπνευματωδών ποτών στο νησί της Κρήτης.

Τέλος, θέλω να ευχαριστήσω την οικογένεια μου για την κάθε υποστήριξη που μου παρείχε όλα αυτά τα χρόνια που είμαι μακριά τους.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή 1

1.1 Υπηρεσίες Επεξεργασίας Δεδομένων 3

1.2 Σκοπός Εργασίας 5

1.3 Δομή Κειμένου 7

ΚΕΦΑΛΑΙΟ 2 Ανασκόπηση Τεχνολογιών Αιχμής 8

2.1 Συστήματα Επεξεργασίας Εικόνων 8

2.1.1 Εργαλειοθήκες – Βιβλιοθήκες 9

1.1.2 Ολοκληρωμένα Συστήματα Επεξεργασίας Εικόνων 10

1.1.3 Οπτικά Περιβάλλοντα Επεξεργασίας Εικόνων 10

1.1.4 Νέες Τάσεις 10

1.2 Πράκτορες (Agents) 12

1.1.1 Χαρακτηριστικά Πρακτόρων 13

1.1.2 Αρχιτεκτονικές 15

1.1.3 Χρήσεις 17

1.3 Διαχείριση Πόρων και Οικονομικά Μοντέλα 18

1.3.1 Μερικοί Ορισμοί 18

1.3.2 Στρατηγικές Κατανομής Φόρτου Εργασίας 19

1.1.3 Οικονομικά Μοντέλα 21

1.1.4 Σχετικές Εργασίες Μηχανισμών Διαχείρισης Πόρων 22

ΚΕΦΑΛΑΙΟ 3 Αρχιτεκτονική Περιβάλλοντος 25

3.1 Ζητούμενα Χαρακτηριστικά και Ανάγκες 26

1.2 Επισκόπηση Αρχιτεκτονικής 26

1.3 Πράκτορας Εκτέλεσης 28

1.4 Διαχειριστής 33

1.5 Διαδικασία Εκτέλεσης Υπηρεσίας 36

1.6 Ακολουθίες Αλγόριθμων 39

ΚΕΦΑΛΑΙΟ 4 Διαχείριση Πόρων 41

1.1 Διαδικασία Ανάθεσης Πόρων 43

1.2 Μοντέλο Συστήματος – Γενικές Παραδοχές 45

1.3 Υπολογισμός Χρόνου 46

1.3.1 Υπολογισμός Χρόνου Μετάδοσης στο Δίκτυο 46

1.3.2 Υπολογισμός Χρόνου Εκτέλεσης 47

1.4 Υπολογισμός Κόστους Εκτέλεσης 49

1.4.1 Υπολογισμός Κόστους Χρήσης Μνήμης 49

1.1.2 Υπολογισμός κόστους μεταφοράς 49

1.1.3 Υπολογισμός τελικού κόστους εκτέλεσης 50

1.5 Ενημέρωση Στατιστικού Προφίλ 50

ΚΕΦΑΛΑΙΟ 5 Υλοποίηση *52*

5.1 Περιορισμοί - Δυσκολίες **52**

5.2 Επιλογές Υλοποίησης **53**

5.2.1 Γλώσσα Προγραμματισμού 53
5.2.2 Ανεξαρτησία από Λειτουργικό Σύστημα και Τρόπο Προγραμματισμού του Δικτύου 54
1.1.3 Επαναχρησιμοποίηση Πηγαίου Κώδικα 54
1.1.4 Επεκτασιμότητα και Διασύνδεση Ετερογενών Εφαρμογών 56
1.1.5 Εσωτερική Επικοινωνία και Ανταλλασσόμενα Μηνύματα 56
1.1.6 Μηχανισμός Καταγραφής Γεγονότων 58
1.1.7 Διεπιφάνεια Χρήσης 58
1.1.8 Βάση Αλγόριθμων 59

1.3 Αποτελέσματα **61**

ΚΕΦΑΛΑΙΟ 6 Συμπεράσματα και Μελλοντική Εργασία *64*

ΠΑΡΑΡΤΗΜΑ Α Διαχείριση Πόρων: Προσομοιωτής - Πειράματα *71*

ΠΑΡΑΡΤΗΜΑ Β Συστήματα Επεξεργασίας Εικόνων *88*

ΠΑΡΑΡΤΗΜΑ Γ Συναρτήσεις Κατακερματισμού Ενός Δρόμου *92*

ΠΑΡΑΡΤΗΜΑ Δ Σχετικές Δημοσιεύσεις *94*

ΒΙΒΛΙΟΓΡΑΦΙΑ *95*

Κατάλογος Σχημάτων

Σχήμα 1: Υπηρεσίες επεξεργασίας δεδομένων. 4

Σχήμα 2: Ένα γενικό μοντέλο πράκτορα όπου φαίνεται και η αλληλεπίδραση με το περιβάλλον του. 13

Σχήμα 3: Ανασκόπηση προτεινόμενης αρχιτεκτονικής. 27

Σχήμα 4: Σχηματική αναπαράσταση της δομής του πράκτορα εκτέλεσης. Να σημειωθεί το συγκεκριμένο σχήμα αναφέρεται στη λειτουργική δομή και όχι στο μοντέλο ανάλυσης πρακτόρων όπως αυτό φαίνεται στο Σχήμα 2. 28

Σχήμα 5: Βήματα εκτέλεσης ενός αλγόριθμου μέσω του εκτελεστή αλγόριθμων. 30

Σχήμα 6: Αρχείο περιγραφής παραμέτρων εκτέλεσης αλγόριθμου. 31

Σχήμα 7: Επέκταση περιβάλλοντος μέσω δικτύου διαχειριστών. 34

Σχήμα 8: Ακολουθία ανάθεσης της εκτέλεσης αλγόριθμου σε πράκτορα εκτέλεσης. 35

Σχήμα 9 : Ανταλλασσόμενα μηνύματα κατά την ανάθεση εκτέλεσης ανάμεσα στον πράκτορα εκτέλεσης (ExecAgent), τον διαχειριστή (ManagementAgent) και την αιτούσα εφαρμογή (Application). 37

Σχήμα 10: Χρονική αλληλουχία ανταλλασσόμενων μηνυμάτων κατά την ανάθεση εκτέλεσης 38

Σχήμα 11: Ένα παράδειγμα ακολουθίας αλγόριθμων όπου φαίνεται μια ποικιλία πολυπλοκότητας σε σχέση με την αλληλουχία των επιμέρους αλγόριθμων και τις αλληλεξαρτήσεις των δεδομένων εισόδου και εξόδου. Οι επιμέρους αλγόριθμοι της ακολουθίας συμβολίζονται με «algo N», όπου N=1, 2, 3,…, ενώ τα δεδομένα εισόδου και εξόδου ενός αλγόριθμου algoN συμβολίζονται με «i.N.k» και «o.N.l», αντίστοιχα (όπου k, l = 1, 2, 3, …). Τέλος, τα δεδομένα εισόδου και εξόδου της ακολουθίας συμβολίζονται με «I.m» και «O.n», αντίστοιχα (όπου m, n = 1, 2, 3, …). 39

Σχήμα 12: Πράκτορας εκτέλεσης ακολουθιών (ΠΕΑ). 40

Σχήμα 13: Παράδειγμα χρήσης του συστήματος. Στο δεξί μέρος της οθόνης φαίνεται το παράθυρο επιλογής αλγόριθμων, όπου στο πάνω μέρος διακρίνεται το δένδρο της δομής των αρχείων ενώ στο κάτω μέρος φαίνονται οι πληροφορίες που συνοδεύουν τον επιλεγμένο αλγόριθμο. 61

Σχήμα 14: Παράδειγμα χρήσης του συστήματος. Στο δεξί μέρος της οθόνης φαίνεται το παράθυρο εκκίνησης εκτέλεσης ενός αλγόριθμου, και συγκεκριμένα η σελίδα για την εισαγωγή τιμών στα ορίσματα εισόδου του αλγόριθμου. 62

Σχήμα 15: Παράδειγμα χρήσης του συστήματος. Στο κάτω-δεξί μέρος της οθόνης φαίνεται το παράθυρο εκκίνησης εκτέλεσης ενός αλγόριθμου, και συγκεκριμένα η σελίδα για την εισαγωγή κριτηρίων για την ανάθεση εκτέλεσης του αλγόριθμου. 62

Σχήμα 16: Παράδειγμα χρήσης του συστήματος. Ο αριθμός των εικόνων που εμφανίζονται ταυτόχρονα στην οθόνη είναι μεταβλητός και επιλέγεται από τον χρήστη. Στο κάτω μέρος της οθόνης φαίνεται το παράθυρο πληροφοριών, στην σελίδα που αντιστοιχεί σε πληροφορίες για την κατάσταση των εκτελέσεων. Συγκεκριμένα, διακρίνεται πληροφορία για μια εκτέλεση που έχει ολοκληρωθεί, έναν αλγόριθμο που εκτελείται, και μια ακόμα εκτέλεση που βρίσκεται στο στάδιο της αρχικής επικοινωνίας. 63

Σχήμα 17: Μερικά από τα βοηθητικά παράθυρα της εφαρμογής χρήστη. Επιλογή εικόνας (Α), παράθυρο εκκίνησης εκτέλεσης στη σελίδα εισαγωγής τιμών στα ορίσματα εισόδου του αλγόριθμου (Β) και στη σελίδα εισαγωγής κριτηρίων ανάθεσης εκτέλεσης (Γ), και επιλογή αλγόριθμου (Δ). 63

Σχήμα 18: Το προτεινόμενο περιβάλλον ολοκληρώνει ανομοιογενείς υπολογιστικούς πόρους και εκτελέσιμα με διαφορετικές απαιτήσεις σε υλικό και λογισμικό, και παρέχει διάφανη εκτέλεση σε κατανεμημένο σύστημα ενός οργανισμού αξιοποιώντας όλους τους διαθέσιμους πόρους. 65

Σχήμα 19: Δημιουργία εικονικού οργανισμού μέσω του προτεινόμενου περιβάλλοντος και παροχή υπηρεσιών σε εξωτερικούς χρήστες. 66

Κατάλογος Δειγμάτων Κώδικα

Δείγμα Κώδικα 1: Παράδειγμα χρήσης των διαφόρων σχεδιαστικών προτύπων. 55

Δείγμα Κώδικα 2: Η κλάση DMsg παρέχει τους βασικούς μηχανισμούς για την ανταλλαγή μηνυμάτων. 57

Δείγμα Κώδικα 3: Παράδειγμα ορισμού μηνύματος για την αίτηση πληροφορίας σχετικής με έναν αλγόριθμο. 58

Δείγμα Κώδικα 4: Δείγμα αρχείου περιγραφής κόμβου αλγόριθμων. 60

Δείγμα Κώδικα 5: Δείγμα αρχείου προφίλ αλγόριθμου (pgmgaussian.ini). 60

Κατάλογος Διαγραμμάτων

Διάγραμμα 1: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για κάθε κατηγορία αλγόριθμων ως προς την ταχύτητα εκτέλεσης τους και για διάφορους μηχανισμούς ανάθεσης εκτέλεσης αλγόριθμων, όπως περιγράφεται στο 1ο πείραμα. Ο κάθετος άξονας (X) αντιστοιχεί στις τιμές μέσης % αύξησης στο χρόνο εκτέλεσης. Κάθε μία από τις τρισδιάστατες ράβδους αντιστοιχεί σε ένα είδος αλγόριθμου και ένα μηχανισμό ανάθεσης εκτέλεσης. Συγκεκριμένα, οι ράβδοι L1-L4 αντιστοιχούν σε τοπική εκτέλεση, οι R1-R4 σε τυχαία κατανομή, και οι A1-A4 σε ανάθεση με δημοπρασία. Τέλος, για κάθε ράβδο ο αριθμός των μηχανημάτων αυξάνεται σταδιακά από 16 (στο βάθος) μέχρι 32 (μπροστά). Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M). 77

Διάγραμμα 2: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για τοπική εκτέλεση (L1), για τυχαία ανάθεση (R1) και για ανάθεση με δημοπρασία (A1), για πολύ γρήγορους αλγόριθμους (αναμενόμενος χρόνος εκτέλεσης 0.1-1 min). Ο κάθετος άξονας (Y) αντιστοιχεί στις τιμές μέσης % αύξησης του χρόνου εκτέλεσης, ενώ στον οριζόντιο άξονα παρουσιάζεται ο αριθμός των μηχανημάτων στο δίκτυο. Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M). 78

Διάγραμμα 3: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για τοπική εκτέλεση (L2), για τυχαία ανάθεση (R2) και για ανάθεση με δημοπρασία (A2), για γρήγορους αλγόριθμους (αναμενόμενος χρόνος εκτέλεσης 1-5 min). Ο κάθετος άξονας (Y) αντιστοιχεί στις τιμές μέσης % αύξησης του χρόνου εκτέλεσης, ενώ στον οριζόντιο άξονα παρουσιάζεται ο αριθμός των μηχανημάτων στο δίκτυο. Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M). 78

Διάγραμμα 4: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για τοπική εκτέλεση (L3), για τυχαία ανάθεση (R3) και για ανάθεση με δημοπρασία (A3), για μέρτιους αλγόριθμους (αναμενόμενος χρόνος εκτέλεσης 5-10 min). Ο κάθετος άξονας (Y) αντιστοιχεί στις τιμές μέσης % αύξησης του χρόνου εκτέλεσης, ενώ στον οριζόντιο άξονα παρουσιάζεται ο αριθμός των μηχανημάτων στο δίκτυο. Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M). 79

Διάγραμμα 5 : Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για τοπική εκτέλεση (L4), για τυχαία ανάθεση (R4) και για ανάθεση με δημοπρασία (A4), για αργούς αλγόριθμους (αναμενόμενος χρόνος εκτέλεσης 10-30 min). Ο κάθετος άξονας (Y) αντιστοιχεί στις τιμές μέσης % αύξησης του χρόνου εκτέλεσης, ενώ στον οριζόντιο άξονα παρουσιάζεται ο αριθμός των μηχανημάτων στο δίκτυο. Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M). 79

Διάγραμμα 6: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για κάθε κατηγορία αλγόριθμων ως προς την ταχύτητα εκτέλεσης τους, για τους 3 διαφορετικούς μηχανισμούς ανάθεσης εκτέλεσης, με διαφορετικό όγκο δεδομένων εισόδου και εξόδου [μικρός (S), μέτριος (M), μεγάλος (L)] και διαφορετικές χρονικές απαιτήσεις [πολύ γρήγορος (1), γρήγορος (2), μέτριος (3), αργός (4)]. 82

Διάγραμμα 7: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για κάθε κατηγορία αλγόριθμων ως προς την ταχύτητα εκτέλεσης τους, για τους 3 διαφορετικούς μηχανισμούς ανάθεσης εκτέλεσης, με διαφορετικό όγκο δεδομένων εισόδου και εξόδου [μικρός (S), μέτριος (M), μεγάλος (L)] και διαφορετικές χρονικές απαιτήσεις [πολύ γρήγορος (1), γρήγορος (2), μέτριος (3), αργός (4)]. 82

Διάγραμμα 8: Μέση % αύξηση χρόνου εκτέλεσης για τους πολύ γρήγορους και γρήγορους αλγόριθμους για τους 3 μηχανισμούς ανάθεσης εκτέλεσης. Ενδιαφέρον παρουσιάζει η αρνητική τιμή για την τοπική εκτέλεση, που οφείλεται στο γεγονός ότι ο αναμενόμενος χρόνος για την εκτέλεση των αλγόριθμων υπολογίζεται με βάση τα μηχανήματα με την μικρότερη υπολογιστική ισχύ, ενώ στο συγκεκριμένο πείραμα το δίκτυο διαθέτει 2 μηχανήματα με τριπλάσια υπολογιστική ισχύ. 84

Διάγραμμα 9: Μέση % αύξηση χρόνου εκτέλεσης για τους μέτριους και αργούς αλγόριθμους για τους 3 μηχανισμούς ανάθεσης εκτέλεσης. 84

Διάγραμμα 10: % χρήση του κάθε μηχανήματος που συμμετέχει στο δίκτυο για τους τρεις μηχανισμούς ανάθεσης εκτέλεσης. Να σημειωθεί ότι το διάγραμμα έχει νόημα μόνο για σημεία – οι γραμμές σχεδιάστηκαν για ευκολία παρουσίασης. 85

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Τα τελευταία χρόνια οι εξελίξεις στην τεχνολογία της πληροφορικής και των επικοινωνιών συντέλεσαν δραστικά στην αλλαγή του τρόπου ζωής. Στις μέρες μας η πληροφορική παίρνει τον χαρακτήρα της καθημερινότητας, αρχίζει δηλαδή να γίνεται μια από τις τεχνολογίες των οποίων η απουσία γίνεται συνήθως περισσότερο αισθητή από την παρουσία τους. Η ιστορία δείχνει ότι, στην πορεία προς την καθιέρωση, σχεδόν όλες οι τεχνολογίες περνούν από τα παρακάτω στάδια [1]:

- ξεκινούν ως επιστημονικοί πειραματισμοί,
- χρησιμοποιούνται από ένα μικρό αριθμό ειδικών για την επίλυση συγκεκριμένων προβλημάτων,
- περνούν στην παραγωγή και την ευρύτερη χρήση, αλλά ακόμα απαιτούν ειδική εκπαίδευση και χρησιμοποιούνται από μικρό ποσοστό του πληθυσμού, και τέλος
- μπαίνουν σε καθημερινή χρήση και θεωρούνται από τους περισσότερους ανθρώπους αναπόσπαστο μέρος των καθημερινών δραστηριοτήτων.

Μπορούμε να πούμε ότι η πληροφορική βρίσκεται σήμερα στο τέλος της τρίτης φάσης και περνά με γρήγορους ρυθμούς προς την ενσωμάτωσή της στην καθημερινή ζωή. Ο κόσμος στον οποίο ζούμε και εργαζόμαστε μεταμορφώνεται από ένα αναλογικό κόσμο, στον οποίο ο καθένας πρέπει να καταβάλει σημαντικές προσπάθειες για να πετύχει τους στόχους και τους σκοπούς του, σε ένα ψηφιακό κόσμο μέσα στον οποίο ο καθένας μπορεί πολύ πιο εύκολα να εργαστεί και να πετύχει τις επιδιώξεις του βοηθούμενος από τεχνολογίες της πληροφορικής. Οι στόχοι αυτοί μπορεί να ποικίλουν από την ανάγνωση ενός σπάνιου βιβλίου και την αποστολή ενός γράμματος σε παλιούς συμμαθητές όπου και να βρίσκονται στον κόσμο, μέχρι την παρακολούθηση του δελτίου ειδήσεων των εννέα οποιαδήποτε χρονική στιγμή, ή μιας αγαπημένης σειράς εθνικού σταθμού από κάποιο σημείο εκτός χώρας. Επίσης, οι υπολογιστές, και συγκεκριμένα η χρήση τους σε συνδυασμό με τεχνολογίες τηλεπικοινωνιών, τείνουν να εξαλείψουν τις αποστάσεις. Ο τρόπος με τον οποίο ο καθένας εργάζεται μεταβάλλεται σημαντικά. Πλέον, σε αρκετές περιπτώσεις, δεν χρειάζεται η φυσική παρουσία ενός ατόμου στο χώρο της εργασίας του, μια και μπορεί να δουλέψει στο σπίτι του ή όπου αλλού χρησιμοποιώντας με τον ίδιο (εύκολο ή δύσκολο) τρόπο τους πόρους που του παρέχει το περιβάλλον εργασίας του.

Η καθιέρωση της πληροφορικής στην καθημερινή ζωή (τουλάχιστον των βιομηχανοποιημένων χωρών) οφείλεται αφενός στην αλματώδη τεχνολογική πρόοδο (αυξανόμενη φιλικότητα του λογισμικού, χαμηλότερο κόστος, δίκτυα υψηλού ρυθμού μετάδοσης δεδομένων, κλπ.), αλλά και σε μια αλλαγή στον τρόπο εργασίας. Κατά τις τελευταίες δεκαετίες, παρατηρείται μία εμφανής μεταστροφή από τις παραγωγικές εργασίες (βιομηχανική παραγωγή, αγροτική παραγωγή, κλπ.) σε εργασίες παροχής υπηρεσιών (τράπεζες, εκπαιδευτικούς οργανισμούς, οργανισμούς υγείας, συμβουλευτικές εταιρείες, κτλ).

Ο όρος 'παροχή υπηρεσιών' αναφέρεται σε όλες τις οικονομικές δραστηριότητες των οποίων το παράγωγο δεν είναι ένα φυσικό προϊόν ή μία κατασκευή, αλλά γενικά καταναλώνεται τη στιγμή της παραγωγής και επιπλέον παρέχει επιπρόσθετη αξία με κάποιο άυλο τρόπο (π.χ. διασκέδαση, ευκολία, άνεση, υγεία) [2]. Ο χώρος της παροχής υπηρεσιών αποτελεί τον μεγαλύτερο τομέα των οικονομιών των βιομηχανοποιημένων κρατών [3] και συνεχώς μεγαλώνει σε απασχόληση, τεχνολογική πληρότητα και επένδυση κεφαλαίων, σε αντίθεση με τον κατασκευαστικό τομέα που έχει παραμείνει σταθερός και τον αγροτικό τομέα που συνεχώς συρρικνώνεται. Χαρακτηριστικά, στις Ηνωμένες Πολιτείες το 74% του συνολικού παραγόμενου προϊόντος είναι υπηρεσίες και απασχολεί το 77% του εργατικού δυναμικού [3]. Παράδειγμα της γενικότερης τάσης μεταστροφής σε εργασίες παροχής υπηρεσιών στην Ευρώπη αποτελεί το Λονδίνο, όπου το 85% του παραγωγικού πληθυσμού του εργάζεται στο γενικότερο χώρο της παροχής υπηρεσιών [4]. Η γενικότερη αυτή τάση έχει αρχίσει να διεισδύει και στο χώρο της πληροφορικής. Μάλιστα, στα επόμενα χρόνια αναμένεται να γίνει μια ακόμη μεγαλύτερη έκρηξη στο χώρο της παροχής υπηρεσιών πληροφορικής, καθώς η τεχνολογία των υπολογιστών και των δικτύων γίνεται όλο και πιο ισχυρή και συγχρόνως και πιο προσιτή στο ευρύ κοινό, όπως χαρακτηριστικά αναφέρει και ο Negroponte στο προφητικό βιβλίο του «Being Digital» [5].

Καθημερινά εμφανίζονται νέες πρωτοποριακές υπηρεσίες από οργανισμούς και εταιρείες που δίνουν τη δυνατότητα στον πελάτη να εκμεταλλευτεί ορισμένα μοναδικά προνόμια. Ταυτόχρονα, διάφορες τεχνολογίες που τις τελευταίες δεκαετίες αναπτύσονται και χρησιμοποιούνται κυρίως από ερευνητικές ομάδες των διαφόρων πανεπιστημίων και του στρατού έχουν αρχίσει να βγαίνουν από τα εργαστήρια και να χρησιμοποιούνται από τον απλό πολίτη. Οι τεχνολογικές εξελίξεις στα συστήματα υπολογιστών καθώς και στα δίκτυα, έκαναν εφικτή την διασύνδεση εκατοντάδων ετερογενών υπολογιστικών και δικτυακών πόρων και συστημάτων. Όλα αυτά τα συστήματα χρησιμοποιούνται από ένα ευρύ φάσμα χρηστών με πολύ διαφορετικές απαιτήσεις από τις παρεχόμενες υπηρεσίες. Είναι χαρακτηριστικό ότι ο αριθμός τόσο των διασυνδεδεμένων συστημάτων όσο και των χρηστών μεγαλώνει συνεχώς με πολύ μεγάλους ρυθμούς [6]. Υπάρχει επίσης μία γενικότερη τάση να τροποποιηθούν τα μέσα διάθεσης και οι μηχανισμοί χρέωσης του λογισμικού. Μέχρι σήμερα οι χρήστες, για να μπορέσουν να χρησιμοποιήσουν κάποιο λογισμικό, έπρεπε να έχουν στην κατοχή τους ένα νόμιμο αντίγραφο και την άδεια χρήσης του, και επιπλέον έπρεπε να το έχουν εγκαταστήσει στον υπολογιστή τους. Συχνό αποτέλεσμα αυτής της πρακτικής είναι οι μεγάλοι χρόνοι ανανέωσης του λογισμικού με νέες ή διορθωτικές εκδόσεις καθώς και η αγορά λογισμικού που δεν χρειάζεται άμεσα στον χρήστη. Σήμερα η διάθεση του λογισμικού (νέες εκδόσεις, διορθώσεις και επιπλέον λειτουργίες) αρχίζει να γίνεται μέσα από το δίκτυο. Επιπλέον, εκτός από τον τρόπο παράδοσης του λογισμικού στο χρήστη, υπάρχει μία τάση για δημιουργία νέων τρόπων χρέωσης. Οι καταναλωτές (χρήστες) δεν έχουν στην κατοχή τους το λογισμικό, αλλά πληρώνουν για τη συνολική χρήση του και ανάλογα με τις συγκεκριμένες λειτουργίες που θέλουν να χρησιμοποιήσουν. Μπορούν με μία απλή επιλογή από το δίκτυο σε πραγματικό χρόνο να 'νοικιάσουν' κάποιο διαφορετικό λογισμικό που ικανοποιεί καλύτερα τις ανάγκες τους εκείνη τη χρονική στιγμή, χωρίς να πρέπει να μπουν σε διαδικασίες απόκτησης και εγκατάστασης του νέου λογισμικού και των πιθανών πόρων που χρειάζεται [7].

Οι παραπάνω εξελίξεις έχουν ως αποτέλεσμα ο χρήστης να ενδιαφέρεται ολοένα και λιγότερο για το λειτουργικό σύστημα που χρησιμοποιεί, για τους υπολογιστικούς πόρους που έχει στην κατοχή του, για τον τόπο όπου βρίσκεται το λογισμικό που χρειάζεται, κλπ. Το μόνο που ενδιαφέρει πλέον τον χρήστη είναι να φέρει σε πέρας τη δουλειά του. Λογική συνέπεια αποτελεί η νέα γενιά των 'υπολογιστών δικτύου' (Network Computer, NC), όπως παρουσιάζονται σε καθημερινές ανακοινώσεις μεγάλων κατασκευαστικών εταιρειών υπολογιστών [8,9]. Αυτή η νέα γενιά υπολογιστών δεν θα έχει ενσωματωμένο λογισμικό με την κλασική έννοια, παρά μόνο τα κατάλληλα δικτυακά πρωτόκολλα και τους μηχανισμούς για τη σύνδεση στο δίκτυο έτσι ώστε ο χρήστης να χρησιμοποιεί ό,τι χρειάζεται από πηγές και πόρους κατανεμημένους στο παγκόσμιο δίκτυο.

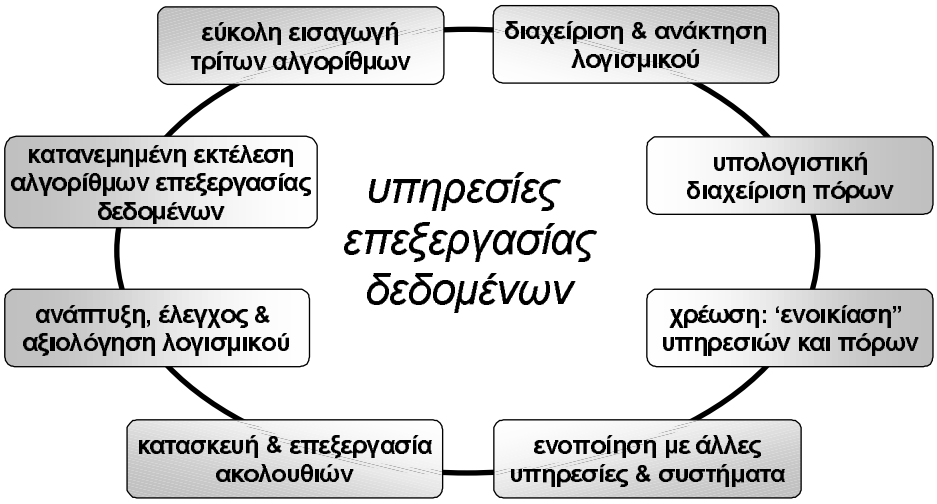
Σαν αποτέλεσμα όλων των παραπάνω, φαίνεται πια έντονη η ανάγκη να αλλάξει ο τρόπος με τον οποίο κατασκευάζεται το λογισμικό, ώστε να λάβει υπόψη του το νέο περιβάλλον μέσα στο οποίο καλείται να λειτουργήσει. Άμεσο αντίκτυπο είναι η κατάργηση της χρήσης πολλών προγραμμάτων, τα οποία ενώ παρέχουν χρήσιμες λειτουργίες, απλά δεν μπορούν να λειτουργήσουν κάτω από τις νέες συνθήκες. Είναι εμφανής η ανάγκη για τη διασύνδεση και ενοποίηση συμβατικών συστημάτων με καινούργια συστήματα, καθώς και η ανάπτυξη κατάλληλων μηχανισμών για την παροχή υπηρεσιών στο ενιαίο παγκόσμιο κατανεμημένο σύστημα που γίνεται πια πραγματικότητα.

1.1 Υπηρεσίες Επεξεργασίας Δεδομένων

Ως ένας από τους σημαντικούς τομείς της πληροφορικής, η παροχή υπηρεσιών επεξεργασίας δεδομένων δεν πρόκειται να μείνει έξω από τις αλλαγές που συμβαίνουν και αναμένεται να εμφανίσει άνθηση τα προσεχή χρόνια. Ένας χρήστης που θέλει να επεξεργαστεί ένα σύνολο από δεδομένα και δεν έχει στην κατοχή του τον τρόπο (κατάλληλο λογισμικό και υλικό), θα αναζητήσει κάποια συγκεκριμένη υπηρεσία επεξεργασίας δεδομένων στο δίκτυο. Η επεξεργασία μπορεί να αφορά διάφορους τύπους δεδομένων. Για παράδειγμα ένας φοιτητής μετεωρολογίας, που ενδιαφέρεται να εκτιμήσει τις αναμενόμενες θερμοκρασίες των επόμενων ημερών, μπορεί να χρειάζεται κάποια ειδική στατιστική επεξεργασία που προσφέρει λογισμικό ειδικού εργαστηρίου απομακρυσμένου πανεπιστημίου. Κάποιος νέος 'παίκτης' στο χρηματιστήριο, σίγουρα θα εκτιμούσε μια ειδική επεξεργασία των δεικτών του χρηματιστηρίου για τις μετοχές που τον ενδιαφέρουν, υπηρεσία που στο μέλλον θα μπορούσε να προσφέρουν μέσα από το δίκτυο σε πραγματικό χρόνο και με σημαντικό αντίτιμο οι ανάλογες χρηματιστηριακές εταιρείες. Τέλος, ένας ειδικευόμενος γιατρός σε ένα μικρό ορεινό ιατρικό κέντρο, θα μπορούσε ίσως να μάθει πολλά από την επεξεργασία μιας ακολουθίας εικόνων που απέκτησε στο ιατρείο του με έναν καινούργιο αλγόριθμο που αναπτύχθηκε πρόσφατα σε ένα ερευνητικό κέντρο (*... σε ένα απομακρυσμένο νησί που συσσωρεύονται πολλοί καλοί επιστήμονες για να αποφύγουν τα φώτα της δημοσιότητας*).

Η πληθώρα των συμβατικών προγραμμάτων επεξεργασίας δεδομένων, αν όχι το σύνολο, δεν είναι σχεδιασμένα με την προοπτική να αποτελέσουν κάποια προσφερόμενη υπηρεσία μέσω του δικτύου. Τα περισσότερα είναι μονολιθικά προγράμματα που έχουν αναπτυχθεί για ένα ειδικό σκοπό, με πολύ περιορισμένες δυνατότητες επέκτασης και ενοποίησης με άλλα αντίστοιχα προγράμματα. Συνήθως, τα περισσότερα μπορούν να χρησιμοποιηθούν μόνο από τον κατασκευαστή τους ή από κάποιο πεπειραμένο ή ειδικά εκπαιδευμένο χρήστη.

Στα πλαίσια της σημερινής ανακατάταξης στον χώρο της πληροφορικής, η επεξεργασία δεδομένων σε ένα μεγάλο κατανεμημένο σύστημα πρέπει να αντιμετωπιστεί ως ένα σύνολο από επιμέρους λειτουργικότητες και υπηρεσίες. Αυτές συνοψίζονται στο Σχήμα 1.



Σχήμα 1: Υπηρεσίες επεξεργασίας δεδομένων.

Κύρια προϋπόθεση είναι η παροχή μηχανισμών για την βέλτιστη και διάφανη χρήση, διάθεση και χρέωση των κατανεμημένων πόρων, είτε πρόκειται για αλγόριθμους επεξεργασίας είτε για υπολογιστική ισχύ. Μία άλλη βασική απαίτηση είναι η υποστήριξη της εύκολης ενοποίησης υπάρχοντος λογισμικού μέσα σε ένα ενοποιημένο περιβάλλον και η παροχή μηχανισμών για τον έλεγχο και την αξιολόγηση της απόδοσης και χρήσης αλγόριθμων επεξεργασίας δεδομένων. Επιπρόσθετα, υπάρχει η ανάγκη για την ανάπτυξη έξυπνων και ευέλικτων μηχανισμών για την περιγραφή, διαχείριση και ανάκτηση λογισμικού επεξεργασίας δεδομένων, έτσι ώστε να γίνει εφικτή η εύκολη χρήση του από μη ειδικούς. Λαμβάνοντας υπόψη μας ότι τα συστήματα επεξεργασίας δεδομένων πολλές

φορές χρειάζονται την εκτέλεση πολύπλοκων ακολουθιών αλγόριθμων, είναι σημαντική η παροχή εργαλείων για την εύκολη κατασκευή, διαχείριση και εκτέλεση ακολουθιών αλγόριθμων. Επίσης, για να ικανοποιηθεί η γενικότερη ανάγκη για διαλειτουργικότητα, οι υπηρεσίες επεξεργασίας δεδομένων πρέπει να μπορούν να ενοποιηθούν και να ολοκληρωθούν με άλλα συστήματα παροχής υπηρεσιών, τόσο σε επίπεδο συστήματος όσο και σε επίπεδο υπηρεσιών. Τέλος, είναι αυτονόητη η ανάγκη για εύκολη πρόσβαση και διαφάνεια στην χρήση όσον αφορά στα παρακάτω:

- λειτουργικό σύστημα,
- υλικό που χρησιμοποιείται,
- τεχνολογίες δικτύου,
- χώρος εφαρμογών.

1.2 Σκοπός Εργασίας

Η παρούσα εργασία μελετά την παροχή υπηρεσιών επεξεργασίας δεδομένων μέσω ενός δικτύου υπολογιστών και προτείνει μία αρθρωτή, ευέλικτη και επεκτάσιμη αρχιτεκτονική η οποία μπορεί να αποτελέσει τον κορμό για την ανάπτυξη συστημάτων για την παροχή υπηρεσιών επεξεργασίας δεδομένων.

Στα πλαίσια της εργασίας, η προτεινόμενη αρχιτεκτονική χρησιμοποιείται για την ανάπτυξη μιας ειδικευμένης εφαρμογής, ενός περιβάλλοντος παροχής υπηρεσιών επεξεργασίας εικόνων, του οποίου η κωδική ονομασία είναι DIPE (Distributed Image Processing Environment). Το περιβάλλον αυτό στοχεύει στο να προσφέρει στον χρήστη, είτε αυτός είναι ένας εξειδικευμένος ερευνητής είτε είναι ένας απλός χρήστης, τη δυνατότητα επεξεργασίας εικόνων χρησιμοποιώντας με διαφάνεια (ως προς την διεύθυνση, πρόσβαση και εκτέλεση) την υπολογιστική ισχύ και τους αλγόριθμους επεξεργασίας ενός ευρύτερου δικτύου. Επιπλέον, αποσκοπεί στο να παρέχει εύκολη και ενιαία πρόσβαση σε διάφορους μεμονωμένους αλγόριθμους επεξεργασίας εικόνων, δίνοντας μηχανισμούς για την εύκολη ενσωμάτωση νέων αλγόριθμων καθώς και αλγόριθμων που έχουν αναπτυχθεί στο παρελθόν από τρίτους (δηλ. δεν υπάρχει πρόσβαση στον πηγαίο κώδικα). Το περιβάλλον παρέχει μηχανισμούς για την βέλτιστη χρήση των διαθέσιμων πόρων, ενώ η ποιότητα των προσφερόμενων υπηρεσιών μπορεί να διακυμανθεί, ανάλογα με το (εικονικό) ποσό που ο χρήστης είναι διατεθειμένος να πληρώσει για κάθε χρήση.

Απώτερος σκοπός της εργασίας είναι να γενικευθούν οι λύσεις και επιλογές που προτείνονται στα πλαίσια του DIPE και, χρησιμοποιώντας αφαιρετικές τεχνικές και μηχανισμούς για την ανεξαρτητοποίηση της αρχιτεκτονικής από την χρήση της σε έναν χώρο εφαρμογών, να προταθούν τρόποι και βήματα για μελλοντική δουλειά με την οποία μπορούμε εύκολα να φτάσουμε στην κατασκευή ενός πλαισίου εργασίας (framework) για την εύκολη και άμεση ανάπτυξη εφαρμογών παροχής υπηρεσιών επεξεργασίας δεδομένων, ανεξαρτήτως χώρου εφαρμογής.

Το άμεσο κίνητρο της εργασίας αποτέλεσε η ανάγκη για την υποστήριξη της κατανεμημένης επεξεργασίας ιατρικών εικόνων [10,11] καθώς και άλλων υπηρεσιών προστιθέμενης αξίας στην τηλε-ακτινολογία, μέσα στο ενοποιημένο περιφερειακό δίκτυο τηλεματικών εφαρμογών στην υγεία, που αναπτύσσεται από το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας στο νησί της Κρήτης [12,13]. Πιο συγκεκριμένα, το περιβάλλον DIPE αναπτύχθηκε αρχικά για να υποστηρίξει τις προχωρημένες απαιτήσεις επεξεργασίας εικόνων συστημάτων όπως τα I^2C [14] και I^2Cnet [15] (ένα σύστημα για την ταξινόμηση και ανάκτηση με βάση το περιεχόμενο ιατρικών εικόνων, καθώς και των επεκτάσεων του ώστε να συμπεριλάβει κατανεμημένες βάσεις ιατρικών εικόνων), καθώς και του CoMed [16] (ένα σύστημα για την σύγχρονη συνεργατική τηλεδιάσκεψη στην ιατρική). Ωστόσο, το DIPE μπορεί να βρει πολλές διαφορετικές εφαρμογές. Κατ' αρχήν, μπορεί να χρησιμοποιηθεί για την βελτιστοποίηση της χρήσης κατανεμημένων πόρων μέσα σε έναν οργανισμό που ασχολείται με την επεξεργασία δεδομένων (π.χ. σχεδιαστήριο βιομηχανίας, ερευνητικό κέντρο, εργαστήριο πανεπιστημίου, κλπ.). Επιπρόσθετα δίνει την δυνατότητα να σχηματιστούν υπερ-δίκτυα (ExtraNets), επιτρέποντας σε διαφορετικούς οργανισμούς που αναπτύσσουν συμπληρωματικά συστήματα επεξεργασίας δεδομένων και συνήθως χρησιμοποιούν διαφορετικό υλικό (hardware), να μοιραστούν τους πόρους τους κάτω από μία ευέλικτη πολιτική χρήσης και χρέωσης [17]. Επιπλέον, δίνει τη δυνατότητα σε οργανισμούς που έχουν στην κατοχή τους λογισμικό επεξεργασίας δεδομένων τελευταίας τεχνολογίας καθώς και τους κατάλληλους υλικούς πόρους, να νοικιάσουν τη χρήση τους μέσα από το δίκτυο σε εξωτερικούς χρήστες που δεν έχουν την κατάλληλη υποδομή, όπως για παράδειγμα στην περίπτωση νεοσύστατης εταιρείας παροχής υπηρεσιών ανάκλησης και επεξεργασίας εικόνων μέσω Internet για εκδότες, διαφημιστές, κλπ. [18]. Η προτεινόμενη αρχιτεκτονική μπορεί, τέλος, να αποτελέσει τον πυρήνα για μία γενικότερη αντιμετώπιση της διεξαγωγής διαφόρων διεργασιών σε ένα κατανεμημένο περιβάλλον, όπως φαίνεται από την πρόσφατη εργασία για την διαχείριση ροών εργασίας στο χώρο παροχής υπηρεσιών υγείας [19].

Σημαντικό είναι το γεγονός ότι το προτεινόμενο περιβάλλον χρησιμοποιεί τεχνικές από διάφορους χώρους για να επιτύχει τους παραπάνω στόχους. Κατ' αρχήν, χρησιμοποιούνται τεχνολογίες πρακτόρων λογισμικού (software agents) σε συνδυασμό με μηχανισμούς που παρουσιάζονται σε πολύπλοκα οικονομικά συστήματα, για την απλούστευση, μοντελοποίηση και λύση του πολύπλοκου προβλήματος της κατανομής εργασίας. Επιπλέον, χρησιμοποιούνται νέες τεχνικές και μεθοδολογίες για την ανάπτυξη μεγάλων συστημάτων και την επαναχρησιμοποίηση λογισμικού, όπως οντοκεντρικός σχεδιασμός και προγραμματισμός, χρήση σχεδιαστικών προτύπων (design patterns), καθώς και χρήση πλαισίων εργασίας.

Συνοψίζοντας, η παρούσα εργασία:

- προτείνει μία νέα αρχιτεκτονική για ανάπτυξη υπηρεσιών επεξεργασίας δεδομένων,
- συνδυάζει τεχνικές πρακτόρων και οικονομικά μοντέλα για διαχείριση πόρων,
- παρουσιάζει ένα περιβάλλον για υπηρεσίες επεξεργασίας εικόνων,
- θέτει τις βάσεις για την κατασκευή ενός πλαισίου εργασίας, και
- ακολουθεί νέες τεχνοτροπίες υλοποίησης μεγάλων εφαρμογών.

1.3 Δομή Κειμένου

Στο επόμενο κεφάλαιο παρουσιάζεται μία εισαγωγή στις διάφορες τεχνολογίες και τεχνικές που αναφέρονται και χρησιμοποιούνται στην εργασία. Αρχικά παρουσιάζεται μια σύντομη ανασκόπηση συστημάτων επεξεργασίας εικόνας, ενώ περισσότερες λεπτομέρειες για ένα αντιπροσωπευτικό δείγμα παρόμοιων συστημάτων δίνονται στο Παράρτημα Β. Στη συνέχεια, παρατίθεται μια εισαγωγή στις τεχνολογίες των πρακτόρων, και τέλος γίνεται μία εισαγωγή στη διαχείριση πόρων και στα οικονομικά μοντέλα που χρησιμοποιούνται ενώ παρουσιάζονται και ορισμένες σημαντικές εργασίες.

Στο τρίτο κεφάλαιο γίνεται μία παρουσίαση της προτεινόμενης αρχιτεκτονικής του συστήματος. Αναλύονται τα ζητούμενα χαρακτηριστικά και παρουσιάζονται τα διάφορα κομμάτια που την αποτελούν καθώς και ο τρόπος που συνεργάζονται. Επίσης, στο Παράρτημα Γ γίνεται μια αναφορά σε μηχανισμούς κατακερματισμού δεδομένων για την εξαγωγή ψηφιακής υπογραφής και τον μοναδικό χαρακτηρισμό ψηφιακών εικόνων.

Στο τέταρτο κεφάλαιο παρουσιάζεται λεπτομερώς ο προτεινόμενος τρόπος διαχείρισης πόρων του συστήματος, ο αλγόριθμος που χρησιμοποιείται καθώς και οι διάφοροι υπολογισμοί που γίνονται κατά την πορεία της απόφασης για την ανάθεση πόρων. Για τον έλεγχο του μηχανισμού διαχείρισης πόρων αναπτύχθηκε προσομοιωτής του συστήματος, και περιγράφεται στο Παράρτημα Α, μαζί με τα αποτελέσματα των σχετικών πειραμάτων.

Στο πέμπτο κεφάλαιο παρουσιάζονται οι περιορισμοί και οι δυσκολίες της υλοποίησης του προτεινόμενου συστήματος καθώς και οι διάφορες λύσεις και επιλογές που τελικά χρησιμοποιήθηκαν. Τέλος, στο έκτο κεφάλαιο, συζητούνται αποτελέσματα και συμπεράσματα της εργασίας και αναφέρονται πιθανές επεκτάσεις για μελλοντική δουλειά.

ΚΕΦΑΛΑΙΟ 2 Ανασκόπηση Τεχνολογιών Αιχμής

Η εργασία ασχολείται με μία πληθώρα θεμάτων. Καταρχήν ο άμεσος στόχος της εργασίας είναι η σχεδίαση και υλοποίηση ενός συστήματος που επιτρέπει την καλύτερη και πιο αποτελεσματική προσφορά υπολογιστικών υπηρεσιών επεξεργασίας εικόνων. Για την επίτευξη αυτού του στόχου χρησιμοποιούνται έννοιες και τεχνικές από τον τομέα του προγραμματισμού με χρήση πρακτόρων. Επίσης, για την μοντελοποίηση και λήψη αποφάσεων σχετικών με την εκτέλεση των υπηρεσιών, χρησιμοποιούνται σε μεγάλο βαθμό διάφορες έννοιες και τεχνικές από το χώρο των οικονομικών.

Αυτό το κεφάλαιο παρουσιάζει μια συνοπτική ανασκόπηση για κάθε ένα από τους προαναφερθέντες τομείς, ξεκινώντας με μια παρουσίαση του χώρου των εφαρμογών επεξεργασίας εικόνων, συνεχίζοντας με μια παρουσίαση του χώρου των πρακτόρων και τελειώνοντας με μια αναφορά σε μηχανισμούς διαχείρισης πόρων και οικονομικών μοντέλων, καθώς και σχετικών συστημάτων που έχουν αναπτυχθεί.

2.1 Συστήματα Επεξεργασίας Εικόνων

Η επεξεργασία οπτικής πληροφορίας είναι σήμερα ένας αρκετά ώριμος και πλούσιος τομέας. Κατά καιρούς έχουν αναπτυχθεί πάρα πολλοί αλγόριθμοι επεξεργασίας εικόνας και βίντεο για μια πληθώρα διαφορετικών εφαρμογών. Σαν αποτέλεσμα, η ανάγκη για συστήματα επεξεργασίας οπτικής πληροφορίας δεν είναι καινούργια και έχει ήδη ικανοποιηθεί με πολλούς τρόπους. Σήμερα υπάρχει ένας μεγάλος αριθμός συστημάτων, είτε εμπορικών είτε ακαδημαϊκών, που καλύπτουν ένα σημαντικό εύρος προβλημάτων. Χαρακτηριστική είναι η έκταση πρόσφατου καταλόγου με εταιρείες (περισσότερες από 2000 εγγραφές) που δραστηριοποιούνται στο χώρο συστημάτων επεξεργασίας οπτικής πληροφορίας [20].

Τα διάφορα συστήματα που έχουν αναπτυχθεί εκτείνονται από αποκλειστικά συστήματα, που στοχεύουν στην επίλυση πολύ εξειδικευμένων προβλημάτων, μέχρι ανοιχτά περιβάλλοντα που είναι σχεδιασμένα για ερευνητικούς και αναπτυξιακούς σκοπούς. Μπορούμε να τα κατατάξουμε σε τρεις μεγάλες κατηγορίες: τις εργαλειοθήκες ή βιβλιοθήκες (toolkits), τα ολοκληρωμένα προγράμματα παρουσίασης και επεξεργασίας εικόνων και τα οπτικά περιβάλλοντα επεξεργασίας εικόνων. Μια πλήρης, αναλυτική μελέτη και αναφορά σε όλα τα σχετικά συστήματα ξεφεύγει από τον σκοπό αυτής της ανασκόπησης. Οι παρακάτω παράγραφοι παρουσιάζουν μόνο τις γενικότερες τάσεις στον χώρο, ενώ στο Παράρτημα Β δίνεται μια σύντομη περιγραφή από αντιπροσωπευτικά δείγματα συστημάτων επεξεργασίας οπτικής πληροφορίας.

2.1.1 Εργαλειοθήκες – Βιβλιοθήκες

Εργαλειοθήκη είναι μια βιβλιοθήκη με συναρτήσεις έτοιμες για χρήση από τους κατασκευαστές των αλγόριθμων. Ο χρήστης της βιβλιοθήκης συνήθως δεν έχει πρόσβαση στις εσωτερικές λεπτομέρειες της υλοποίησης της βιβλιοθήκης καθώς και στον πηγαίο κώδικα της. Ο προγραμματιστής ενδιαφέρεται κυρίως για τον τρόπο με τον οποίο μπορεί να χειριστεί τις λειτουργίες που προσφέρει η βιβλιοθήκη, ενώ συνήθως αδιαφορεί για τον τρόπο που αυτές υλοποιούνται (εκτός από εξειδικευμένες περιπτώσεις, όπου η υλοποίηση μπορεί να έχει σημαντικό αντίκτυπο στον χρόνο εκτέλεσης). Επιπλέον, μια βιβλιοθήκη συχνά παρέχει στον προγραμματιστή ένα πολύ καλά καθορισμένο τρόπο με τον οποίο πρέπει να τη χρησιμοποιήσει για να πάρει τα επιθυμητά αποτελέσματα.

Οι λειτουργίες που παρέχονται από τη βιβλιοθήκη, εκτελούν πολύ βασικές πράξεις πάνω στα δεδομένα εισόδου. Εκτελούν δηλαδή κάτι αντίστοιχο με τις βασικές πράξεις πρόσθεσης – πολλαπλασιασμού πινάκων, στα αντίστοιχα πάντα δεδομένα (εικόνες). Ο προγραμματιστής πρέπει να συνδυάσει τις διάφορες λειτουργίες που του παρέχονται από τη βιβλιοθήκη, ώστε να επιτύχει το σκοπό του. Σε πολύ σπάνιες περιπτώσεις η βιβλιοθήκη παρέχει και ορισμένες υψηλότερου επιπέδου λειτουργίες.

Πολλές βιβλιοθήκες έχουν αναπτυχθεί κατά καιρούς για διάφορους σκοπούς. Συνήθως κάθε σοβαρή προσπάθεια για ανάπτυξη ενός συστήματος για την επεξεργασία εικόνων, περιλαμβάνει πρώτα την ανάπτυξη μιας βιβλιοθήκης, την οποία θα χρησιμοποιήσει αργότερα για την κατασκευή του τελικού συστήματος. Παραδείγματα βιβλιοθηκών και εργαλειοθηκών αποτελούν τα ελεύθερα προς διάθεση PBMPlus [21], NetPBM [22], Utah Raster Toolkit [23] και Visualization Toolkit [24], καθώς και τα προϊόντα ImageGear [25], ImageVision [26] και LeadTools [27]. Αξιοσημείωτη είναι η σημαντική προσπάθεια του προγράμματος Image Understanding Environment (IUE) [28] να δημιουργήσει μια ενιαία υποδομή από κλάσεις ιεραρχίας, εργαλεία διεπιφάνειας χρήστη-μηχανής, και αλγόριθμων επεξεργασίας που απαιτούνται για την έρευνα στο χώρο της ανάλυσης και κατανόησης εικόνας.

Η μεγάλη πληθώρα διαφορετικών βιβλιοθηκών, οι οποίες δεν είναι συμβατές μεταξύ τους ανέδειξε την ανάγκη να δημιουργηθεί ένα κοινό σύνολο εντολών διασύνδεσης εφαρμογής – προγραμματιστή (Application Programmers Interface - API) για χρήση στις βιβλιοθήκες επεξεργασίας εικόνας. Η χρήση API θα επέτρεπε σε αλγόριθμους επεξεργασίας εικόνας να μπορούν να χρησιμοποιήσουν με διάφανο τρόπο διάφορες βιβλιοθήκες, δίνοντας έτσι στον προγραμματιστή ανεξαρτησία από κατασκευαστές βιβλιοθηκών και κατ' επέκταση ανεξαρτησία από υπολογιστικές μηχανές και τα λειτουργικά συστήματα. Σαν αποτέλεσμα, διάφορες προσπάθειες για την κατασκευή ενός τέτοιου πρότυπου ενώθηκαν κάτω από την ομπρέλα του Διεθνούς Οργανισμού Τυποποίησης (ISO) και αναπτύχθηκε το πρότυπο PIKS (Programmer's Imaging Kernel System) [29].

2.1.2 Ολοκληρωμένα Συστήματα Επεξεργασίας Εικόνων

Τα συστήματα αυτής της κατηγορίας είναι ουσιαστικά επεκτάσεις σε διάφορες βιβλιοθήκες αλγόριθμων επεξεργασίας προσφέροντας ταυτόχρονα και το κατάλληλο περιβάλλον για την παρουσίαση και την επεξεργασία δεδομένων με ενιαίο τρόπο. Συνήθως στοχεύουν σε συγκεκριμένες εφαρμογές επεξεργασίας οπτικής πληροφορίας και απευθύνονται κυρίως στον τελικό χρήστη, όχι στον ερευνητή ή τον δημιουργό αλγόριθμων επεξεργασίας. Τα συστήματα αυτής της κατηγορίας συμπεριφέρονται ουσιαστικά σαν 'μαύρα κουτιά', δίνοντας την δυνατότητα παρουσίασης εικόνων και εκτέλεσης των συγκεκριμένων ενσωματωμένων αλγόριθμων επεξεργασίας. Τα περισσότερα επιτρέπουν απλές μορφές επεξεργασίας εικόνας, και κάπως πιο προχωρημένες λειτουργίες ή και εξειδικευμένους αλγόριθμους επεξεργασίας εικόνας. Ελάχιστα προγράμματα παρέχουν τη δυνατότητα στο χρήστη να τα προγραμματίσει και να μπορέσει να προσθέσει λειτουργίες στις ήδη υπάρχουσες. Παραδείγματα ολοκληρωμένων προγραμμάτων παρουσίασης και επεξεργασίας εικόνων περιλαμβάνουν επαγγελματικά συστήματα επεξεργασίας εικόνας όπως τα Adobe Photoshop [30], Corel Photopaint [31] και Paint Shop Pro [32]. Υπάρχει επίσης μια πληθώρα συστημάτων επεξεργασίας εικόνας για γενική χρήση στον ακαδημαϊκό χώρο, για παράδειγμα τα XV [33], GIMP [34], και ImageMagick [35]. Τέλος υπάρχει ένας μεγάλος αριθμός από εξειδικευμένα συστήματα, και ενδεικτικά παραδείγματα περιλαμβάνουν το VisiLog [36] για επιστημονικές και βιομηχανικές εφαρμογές, τα OSIRIS [37] και BIPS-HELIOS [38] για επεξεργασία ιατρικών εικόνων, το AIPS [39] για επεξεργασία εικόνων στον χώρο της αστρονομίας, κλπ.

2.1.3 Οπτικά Περιβάλλοντα Επεξεργασίας Εικόνων

Τα οπτικά περιβάλλοντα επεξεργασίας εικόνων έρχονται να καλύψουν τις ανάγκες κυρίως των χρηστών που ασχολούνται με την ανάπτυξη νέων αλγόριθμων επεξεργασίας, αλλά δεν έχουν βαθιές γνώσεις προγραμματισμού και υπολογιστικών συστημάτων. Τα συστήματα αυτής της κατηγορίας βασίζονται σε οπτικές γλώσσες προγραμματισμού, οι οποίες επιτρέπουν στον χρήστη να αναπτύξει αλγόριθμους μέσω ενός συνόλου γραφικών συμβόλων, κατάλληλα συνδεδεμένων σε ένα δυσδιάστατο (ή μεγαλύτερο) χώρο [40]. Παραδείγματα συστημάτων που χρησιμοποιούν οπτικές γλώσσες προγραμματισμού αποτελούν τα KBVision [41], Aphelion [42], SCIL-Image [43] και Khoros [44,45,46].

2.1.4 Νέες Τάσεις

Τα περισσότερα από τα παραπάνω συστήματα και περιβάλλοντα είναι μονολιθικά προγράμματα τα οποία δεν επιτρέπουν την ενσωμάτωση αλγόριθμων και προγραμμάτων τα οποία έχουν αναπτυχθεί για διαφορετικό σκοπό, κάτω από διαφορετικό περιβάλλον με διαφορετικές προδιαγραφές. Επίσης, ένα πολύ βασικό μειονέκτημά τους είναι ότι δεν λαβαίνουν υπόψη τους ότι ο υπολογιστής στον οποίο εκτελούνται συνήθως αποτελεί μέρος ενός δικτύου υπολογιστών. Πρόσφατα, έχουν γίνει κάποιες περιορισμένες προσπάθειες για την επέκταση των δυνατοτήτων συμβατικών συστημάτων επεξεργασίας οπτικής

πληροφορίας τα οποία χαρακτηρίζονται από κάποιες λειτουργικότητες προστιθέμενης αξίας.

Συγκεκριμένα, νεότερη έκδοση του Khoros [46] υποστηρίζει την απομακρυσμένη εκτέλεση αλγόριθμων. Ωστόσο, αυτό γίνεται με τρόπο όχι διάφανο για τον χρήστη, ο οποίος πρέπει να ξέρει σε ποιο ακριβώς μηχάνημα υπάρχει το εκτελέσιμο του αλγόριθμου που τον ενδιαφέρει. Ένα δεύτερο μειονέκτημα της απομακρυσμένης εκτέλεσης που υποστηρίζει το Khoros είναι ότι δεν γίνεται καμία προσπάθεια για την επίτευξη της βέλτιστης δυνατής κατανομής των εκτελέσεων στους διαθέσιμους υπολογιστικούς πόρους του δικτύου. Αυτές οι αδυναμίες είναι ωστόσο αναμενόμενες, μια και το Khoros δεν σχεδιάστηκε με την προοπτική της δυνατότητας απομακρυσμένης εκτέλεσης αλγόριθμων, οπότε η λύση που προσφέρει στο συγκεκριμένο θέμα είναι λύση ανάγκης. Μηχανισμός για αυτόματη κατανομή εκτελέσεων αλγόριθμων σε δίκτυο υπολογιστών χρησιμοποιείται και από το TRACEE [47], ένα εξειδικευμένο σύστημα επεξεργασίας εικόνων για αναγνώριση στόχων. Ωστόσο, η ανάθεση υπολογιστικών πόρων βασίζεται σε ένα κεντρικό μοντέλο περιγραφής του τοπικού δικτύου, χωρίς τη χρήση μιας εξελιγμένης πολιτικής κατανομής εκτελέσεων, ενώ η διανομή των αρχείων δεδομένων εισόδου βασίζεται εκτενέστατα στο λειτουργικό σύστημα (χρησιμοποιώντας το NFS - Network File System πρωτόκολλο).

Με δεδομένα τον μεγάλο αριθμό και την ποικιλία συμβατικών συστημάτων επεξεργασίας οπτικής πληροφορίας, είναι αξιοσημείωτη η έλλειψη ενός περιβάλλοντος που να υποστηρίζει όλες εκείνες τις λειτουργικότητες προστιθέμενης αξίας που συνιστούν τις υπηρεσίες επεξεργασίας οπτικής πληροφορίας, όπως έχουν αναφερθεί στην εισαγωγή αυτής της εργασίας και αναλύονται στο τρίτο κεφάλαιο. Η παροχή υπηρεσιών επεξεργασίας δεδομένων απαιτεί ικανότητα ευελιξίας ως προς την πολιτική και στρατηγική για την ανάθεση πόρων και την χρέωση των παρεχόμενων υπηρεσιών, ικανότητα άμεσης επέκτασης γεωγραφικά και λειτουργικά, υποστήριξη ετερογενών υπολογιστικών συστημάτων (υλικό και λογισμικό), καθώς και εύκολη αναπροσαρμογή στις εξελισσόμενες ανάγκες των χρηστών. Είναι φανερό ότι οι απαιτήσεις ενός τέτοιου περιβάλλοντος δεν μπορούν να ικανοποιηθούν εύκολα από μια απλή επέκταση ή προσαρμογή συστημάτων που αρχικά έχουν σχεδιαστεί για συμβατική επεξεργασία οπτικής πληροφορίας. Για να αντιμετωπίσει τις παραπάνω απαιτήσεις, η αρχιτεκτονική που προτείνει αυτή η εργασία βασίζεται σε τεχνικές από τον χώρο των πρακτόρων λογισμικού. Οι επόμενες παράγραφοι δίνουν μια σύντομη ανασκόπηση αυτού του χώρου.

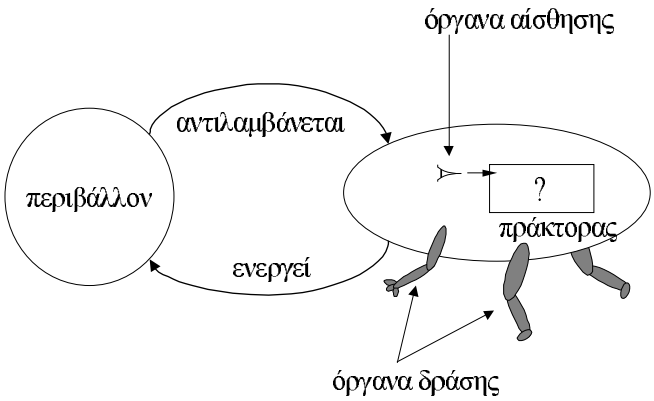
2.2 Πράκτορες (Agents)

Η λεξικογραφική ερμηνεία της αγγλικής λέξης «agent» είναι ένα άτομο το οποίο ενεργεί εκ μέρους ενός άλλου ατόμου, δηλαδή ο αντιπρόσωπος, ο πράκτορας ή ο μεσίτης. Η ελληνική μετάφραση της λέξης «agent» όταν χρησιμοποιείται σε ορολογία σχετική με υπολογιστές είναι «πράκτορας», σύμφωνα με τον ΕΛΟΤ (Ελληνικός Οργανισμός Τυποποίησης) [48]. Εφόσον υπάρχει επίσημη μετάφραση της λέξης «agent» κρίνουμε απολύτως σωστή τη χρήση της, όσο και αν ξενίζει στη ροή του κειμένου και παρόλα τα πολλά υπονοούμενα της (... *πράκτορας 007*!).

Σε όλους μας είναι η γνωστή η έννοια του πράκτορα, όπως αυτή χρησιμοποιείται στην καθημερινή ζωή. Για παράδειγμα θεωρούμε τον ρόλο που παίζει ένας ταξιδιωτικός πράκτορας ή ένας κτηματομεσίτης. Και οι δύο έχουν αντιπροσωπευτικό ρόλο, δηλαδή ενεργούν εκ μέρους κάποιου άλλου. Στην περίπτωση του ταξιδιωτικού πράκτορα εκ μέρους των ξενοδοχείων και των αεροπορικών εταιρειών, ενώ στην περίπτωση του κτηματομεσίτη εκ μέρους του ιδιοκτήτη. Επιπλέον, εκτός από το χαρακτήρα του μεσολαβητή που εμφανίζουν οι δύο προαναφερθέντες πράκτορες, έχουν ένα δεύτερο εξίσου σημαντικό χαρακτηριστικό, εμφανίζουν ένα διαφορετικό βαθμό αυτονομίας. Για παράδειγμα οι κτηματομεσίτες αναλαμβάνουν να κλείσουν ραντεβού για επίδειξη ενός σπιτιού σε ένα πιθανό αγοραστή, χωρίς να ανακατεύουν τους ιδιοκτήτες, ενώ κάποιοι απ' αυτούς έχουν και την επιπλέον εξουσιοδότηση να διαπραγματευτούν και την τιμή πώλησης. Ένα τρίτο σημαντικό χαρακτηριστικό του κάθε πράκτορα είναι ο βαθμός της αντίδρασης και προνοητικότητας που υπάρχει στην συμπεριφορά του. Για παράδειγμα ένας κτηματομεσίτης που απλά τοποθετεί μία πινακίδα με το σήμα «Πωλείται» έξω από μία ιδιοκτησία και περιμένει τους πελάτες να εμφανιστούν στο γραφείο του βασίζεται περισσότερο στην αντίδραση. Αντίθετα, ένας κτηματομεσίτης ο οποίος διαφημίζει γενικά την προσφορά ιδιοκτησίας στον τοπικό τύπο, ενεργεί προνοητικά προκαλώντας το ενδιαφέρον πιθανών πελατών ακόμη και πριν να έχει έτοιμη για διάθεση κάποια ιδιοκτησία. Οι έννοιες της αντίδρασης (reactiveness) και της προνοητικότητας (proactiveness) δεν είναι αντίθετες. Ο ίδιος πράκτορας μπορεί να εμφανίσει υψηλούς βαθμούς αντίδρασης και προνοητικότητας σε διαφορετικές χρονικές στιγμές. Τέλος, οι προαναφερθέντες πράκτορες εμφανίζουν και άλλα χαρακτηριστικά, όπως η δυνατότητα μάθησης, η συνεργατικότητα και η κινητικότητα.

Αντίστοιχα δεδομένα ισχύουν και για τους υπολογιστικούς πράκτορες (computational agents). Ο όρος «πράκτορας» χρησιμοποιείται ευρύτατα στο χώρο της πληροφορικής για να περιγράψει ένα μεγάλο εύρος από υπολογιστικές οντότητες όπως λογισμικά συστήματα και αυτόνομα ρομπότ. Ο όρος «πράκτορας» στον χώρο των υπολογιστών έχει τη βάση του στο χώρο της τεχνητής νοημοσύνης όπου οι ερευνητές προσπαθούν να κατασκευάσουν τεχνητά όντα τα οποία μιμούνται ανθρώπινες ικανότητες [49].

Ως πράκτορα μπορούμε να θεωρήσουμε ένα σύστημα το οποίο προσπαθεί να ικανοποιήσει ένα σύνολο στόχων μέσα σε ένα πολύπλοκο, δυναμικό περιβάλλον [50]. Ο πράκτορας βρίσκεται μέσα στο περιβάλλον, μπορεί να αντιληφθεί το περιβάλλον του μέσω διαφόρων αισθητήρων (sensors) και να ενεργήσει στο περιβάλλον μέσω διαφόρων οργάνων δράσης (actuators) [51]. Ένας ανθρώπινος πράκτορας έχει μάτια, αυτιά και άλλα όργανα για αισθητήρες, έχει χέρια, πόδια, στόμα και άλλα μέλη του σώματος για την εξάσκηση δράσης. Ένας ρομποτικός πράκτορας υποκαθιστά με κάμερες και υπέρυθρους ανιχνευτές πεδίου τους αισθητήρες ενώ επιτυγχάνει δράση με διάφορα μοτέρ. Ένας λογισμικός πράκτορας έχει κωδικοποιημένες σειρές χαρακτήρων ως όργανα αίσθησης και δράσης. Ένα γενικό διάγραμμα ενός πράκτορα παρουσιάζεται στο Σχήμα 2. Οι στόχοι που έχει ο κάθε πράκτορας μπορούν να πάρουν πολλές διαφορετικές μορφές. Μπορεί να είναι τελικοί στόχοι ή καταστάσεις τις οποίες ο πράκτορας προσπαθεί να επιτύχει, μία αμοιβή την οποία ο πράκτορας προσπαθεί να μεγιστοποιήσει, εσωτερικές ανάγκες ή κίνητρα τα οποία ο πράκτορας πρέπει να κρατήσει μέσα σε ορισμένα πλαίσια και όρια [51, 52].



Σχήμα 2: Ένα γενικό μοντέλο πράκτορα όπου φαίνεται και η αλληλεπίδραση με το περιβάλλον του.

2.2.1 Χαρακτηριστικά Πρακτόρων

Πολλοί ερευνητές έχουν δώσει κατά καιρούς διάφορους ορισμούς για την έννοια του πράκτορα, ελπίζοντας να εξωτερικεύσουν το δικό τους τρόπο χρήσης των «πρακτόρων». Μία συνοπτική παρουσίαση διαφορετικών ορισμών παρουσιάζεται στο [53]. Παρόλη την εκτεταμένη χρήση του όρου «πράκτορας» στην πληροφορική δεν υπάρχει σήμερα ένας σαφής και ξεκάθαρος ορισμός του. Αντίθετα, η έκταση της χρήσης του όρου τείνει να θολώσει τις διαφορές μεταξύ εντελώς διαφορετικών κατευθύνσεων στην χρήση πρακτόρων.

Για παράδειγμα, κάποιοι πράκτορες είναι σχεδιασμένοι να δουλεύουν μόνοι τους, άλλοι συνεργάζονται μέσα σε κοινωνίες πρακτόρων, μερικοί είναι κινητοί, άλλοι είναι στατικοί, άλλοι επικοινωνούν μεταξύ τους μέσω μηνυμάτων, άλλοι μαθαίνουν και αλλάζουν τρόπο συμπεριφοράς και άλλοι όχι.

Το βασικό κοινό χαρακτηριστικό, αυτό που κάνει ένα κομμάτι υλικού ή λογισμικού να ονομάζεται πράκτορας, είναι το γεγονός ότι ο δημιουργός του επέλεξε να το αναλύσει, να το σχεδιάσει και να το ελέγξει χρησιμοποιώντας νοητικούς και ανθρωπομορφικούς όρους. Έτσι, πράκτορας είναι μία οντότητα της οποίας την κατάσταση μπορούμε να την περιγράψουμε με ανθρωπομορφικές έννοιες όπως γνώσεις, ικανότητες, επιλογές και υποχρεώσεις. Αντί λοιπόν αυστηρού ορισμού, η ανασκόπηση αυτή παραθέτει ένα σύνολο από χαρακτηριστικά που διακρίνουν τους πράκτορες λογισμικού [50, 51,52, 54]:

- **Αυτονομία**: Οι πράκτορες δρουν εξ ορισμού αυτόνομα, μεσολαβώντας για ένα χρήστη, χωρίς την απευθείας παρεμβολή του χρήστη ή άλλων και έχουν σε κάποιο βαθμό τον έλεγχο πάνω στις πράξεις τους και την εσωτερική τους κατάσταση.

- **Νοημοσύνη**: Σαν αποτέλεσμα της αυτονομίας οι πράκτορες παρουσιάζουν κάποιο επίπεδο νοημοσύνης, το οποίο συνήθως αναδεικνύεται μέσα από το σύνολο της συμπεριφοράς τους και ενισχύεται από μηχανισμούς για την προσαρμογή σε αλλαγές του περιβάλλοντος και την αυτο-εκμάθηση.

- **Αντίδραση - Προνοητικότητα**: Η νοημοσύνη αυτή επιτρέπει στους πράκτορες να αντιλαμβάνονται το περιβάλλον τους και σαν αποτέλεσμα μπορούν να δρουν όχι μόνο αντιδραστικά (reactively) αλλά και προνοητικά (proactively).

- **Κοινωνική ένταξη**: Οι πράκτορες είναι κοινωνικά ενταγμένοι και ενεργοποιημένοι, δηλαδή μπορούν να επικοινωνούν με τον χρήστη και χρειάζονται ειδικούς πόρους για κάτι τέτοιο. Πιο προχωρημένοι πράκτορες πιθανά να συνεργάζονται και με άλλους πράκτορες για να πετύχουν στόχους πέρα των δυνατοτήτων τους, χρησιμοποιώντας κάποια ειδική γλώσσα για την επικοινωνία μεταξύ τους. Το περιβάλλον και η κοινωνία των πρακτόρων, δηλαδή, μπορεί να αποτελείται είτε μόνο από τον χρήστη, είτε από μια ποικιλία χρηστών και άλλων πρακτόρων.

Εκτός από τα παραπάνω βασικά χαρακτηριστικά των πρακτόρων, υπάρχει κι ένα σύνολο από δευτερεύουσες ιδιότητες που συχνά τους χαρακτηρίζουν:

- **Ειλικρίνεια:** Οι πράκτορες δεν θα δώσουν ποτέ εν γνώση τους λάθος πληροφορία όταν τους ζητηθεί.

- **Καλή προαίρεση:** Οι πράκτορες δεν έχουν συγκρουόμενους στόχους, δηλαδή κάθε πράκτορας προσπαθεί να κάνει αυτό που του ζητήθηκε.

- **Λογική:** Ένας πράκτορας ενεργεί πάντα με τρόπο που να προσπαθεί να πετύχει τους στόχους του, και ποτέ δεν θα ενεργήσει με τρόπο που να απαγορεύει την επίτευξη των στόχων του.

- **Κινητικότητα:** Τελικά, είτε σαν κινητά ή ενεργά αντικείμενα, μπορούν να μεταφέρονται μεταξύ συστημάτων για να χρησιμοποιήσουν απομακρυσμένους πόρους ή να συναντηθούν και να συνεργαστούν με άλλους πράκτορες.

Η χρήση εννοιών από το χώρο των πρακτόρων, μπορεί πολλές φορές να βοηθήσει στην καλύτερη μοντελοποίηση και κατανόηση ενός προβλήματος. Η ερώτηση «τι είναι πράκτορας», μετατρέπεται στην ερώτηση «ποια κομμάτια λογισμικού μπορούν να περιγραφούν με ανθρωπομορφικές έννοιες». Έτσι έχει δημιουργηθεί ένας νέος τρόπος προγραμματισμού με κέντρο την έννοια του «πράκτορα», ο οποίος στη διεθνή βιβλιογραφία είναι γνωστός με την κωδική ονομασία AOP (Agent Oriented Programming) [55]. Πρέπει να σημειωθεί ότι ο τρόπος αυτός προγραμματισμού αποτελεί ειδίκευση του οντοκεντρικού προγραμματισμού (Object Oriented Programming, OOP). Ενώ ο OOP, προτείνει τη θεώρηση ενός υπολογιστικού συστήματος σαν μία συλλογή από αντικείμενα τα οποία έχουν κάποια εσωτερική κατάσταση, είναι ικανά να επικοινωνούν μεταξύ τους και έχει το καθένα ξεχωριστούς τρόπους να χειριστεί τα εισερχόμενα μηνύματα, ο AOP εξειδικεύει τους παραπάνω ορισμούς και θεωρεί την κατάσταση (=νοητική κατάσταση) των αντικειμένων (=πράκτορες) να αποτελείται από διάφορα κομμάτια όπως τα πιστεύω του καθενός (σχετικά με τον κόσμο, τον εαυτό του, τους άλλους), τις ικανότητες του και τις αποφάσεις που μπορεί να πάρει. Διάφοροι περιορισμοί μπαίνουν στην νοητική κατάσταση του κάθε πράκτορα, οι οποίοι αντιστοιχούν στα φυσικά ανάλογα τους. Τέλος, κάθε υπολογισμός αντιστοιχεί σε πράκτορες που ανταλλάσσουν πληροφορίες, κάνουν αιτήσεις, προσφέρουν, απορρίπτουν, συναγωνίζονται και βοηθάνε ο ένας τον άλλο.

2.2.2 Αρχιτεκτονικές

Ο προγραμματισμός με βάση τους πράκτορες, όπως αναφέρθηκε, προτείνει τη μοντελοποίηση και επίλυση ενός προβλήματος χρησιμοποιώντας διάφορα αντικείμενα – πράκτορες, τα οποία αλληλεπιδρούν και συνεργάζονται μεταξύ τους, χωρίς όμως να προτείνει το τρόπο με το οποίο γίνεται κάτι τέτοιο. Η μεθοδολογία με την οποία αναπτύσσονται οι πράκτορες ορίζεται από την αρχιτεκτονική τους, η οποία καθορίζει τον τρόπο με τον οποίο ο πράκτορας μπορεί να αναλυθεί σε διάφορα κομμάτια και πως αυτά τα κομμάτια αλληλεπιδρούν μεταξύ τους. Το σύνολο των κομματιών και οι αλληλοεπιδράσεις τους καθορίζουν τις ενέργειες του πράκτορα και την μελλοντική κατάσταση του σε σχέση με τα δεδομένα των αισθητηρίων και σε συνδυασμό με την εσωτερική κατάσταση του πράκτορα. Οι αρχιτεκτονικές πρακτόρων μπορούν σε γενικές γραμμές να χωριστούν στις παρακάτω κατηγορίες [52]:

- Αρχιτεκτονικές 'προκαθορισμού' (deliberative): Ο πράκτορας διαθέτει ένα απόλυτα καθορισμένο συμβολικό μοντέλο του κόσμου, και οι αποφάσεις δράσης παίρνονται με λογική αιτιολόγηση, η οποία βασίζεται σε ταίριασμα προτύπων και δεδομένων αισθητήρων και μετασχηματισμό συμβόλων.

- Αρχιτεκτονικές αντίδρασης (reactive): Εδώ δεν υπάρχει κεντρικό συμβολικό μοντέλο του κόσμου και οι αποφάσεις δράσης δεν απαιτούν πολύπλοκο συλλογισμό με χρήση

συμβόλων. Αντίθετα, η δράση του πράκτορα είναι αποτέλεσμα απλής αντίδρασης στη λήψη συγκεκριμένων δεδομένων στους αισθητήρες. Ανάλογο από τον βιολογικό κόσμο μπορεί να θεωρηθεί η φυσιολογία των αντανακλαστικών.

- Υβριδικές αρχιτεκτονικές: Ένα μεγάλο μέρος των αρχιτεκτονικών πρακτόρων έχει στοιχεία και από τις δύο παραπάνω ακραίες αρχιτεκτονικές.

Επιπλέον, μια άλλη ταξινόμηση των συστημάτων πρακτόρων αφορά τον αριθμό των πρακτόρων. Έτσι διακρίνουμε συστήματα ενός πράκτορα (SAS, Single Agent Systems) και συστήματα πολλών πρακτόρων (MAS, Multi Agent Systems). Στα συστήματα SAS, ο πράκτορας ενεργεί σαν μεσολαβητής ενός χρήστη ή μιας διεργασίας. Ενώ εκτελεί την εργασία του ο πράκτορας μπορεί να επικοινωνήσει με τον χρήστη ή με κάποιο τοπικό ή απομακρυσμένο πόρο του συστήματος, αλλά σε καμία περίπτωση με κάποιον άλλον πράκτορα. Σε αντίθεση, στα συστήματα MAS, οι πράκτορες δεν επικοινωνούν μόνο με το χρήστη και με τους διάφορους πόρους του συστήματος, αλλά επικοινωνούν και συνεργάζονται ενεργά με άλλους πράκτορες, δίνοντας λύση σε προβλήματα που ξεπερνάνε τις δυνατότητες του καθενός χωριστά. Στα συστήματα MAS πολύ συχνά εμφανίζονται προβλήματα συντονισμού και συνεργασίας της κοινότητας των πρακτόρων. Για την επίλυσή τους εφαρμόζονται μέθοδοι κατανεμημένης λύσης προβλημάτων. Πολλές φορές χρειάζονται διαπραγματεύσεις για την επίλυση πιθανών συγκρούσεων. Τέτοιες συγκρούσεις προκαλούνται όταν οι πράκτορες ανταγωνίζονται μεταξύ τους ή όταν προσπαθούν να συνεργαστούν στην κατασκευή ενός κοινού σχεδίου για την επίλυση ενός προβλήματος. Έτσι έχουμε δύο ειδών πράκτορες που συμμετέχουν σε MAS συστήματα:

Ανταγωνιστικοί Πράκτορες: Ο στόχος του κάθε πράκτορα είναι η μεγιστοποίηση των κερδών του, ενώ προσπαθεί να φτάσει σε συμφωνία με τους υπόλοιπους. Παραδείγματα αποτελούν πράκτορες που πουλάνε ή αγοράζουν προϊόντα όπου οι διαβουλεύσεις για την επίλυση των συγκρούσεων γίνονται σε ανταγωνιστικό επίπεδο. Κάθε ένας προσπαθεί να πετύχει την υψηλότερη ή χαμηλότερη δυνατή τιμή για το δικό του όφελος και όχι για το καλό της εμπορικής κοινότητας σαν σύνολο. Έτσι αυτοί οι πράκτορες εργάζονται για κάποιο συγκεκριμένο χρήστη και όχι για κάποια ενοποιημένη κοινότητα χρηστών.

Συνεργατικοί Πράκτορες: Σε αντίθεση με τους προηγούμενους, οι συνεργατικοί πράκτορες μοιράζονται τη γνώση τους και τα πιστεύω τους και προσπαθούν να μεγιστοποιήσουν το όφελος της κοινότητας στο σύνολό της. Για παράδειγμα, θεωρούμε ένα σενάριο για τον έλεγχο της εναέριας κυκλοφορίας του αεροδρομίου των Σπάτων, όπου οι πράκτορες, οι οποίοι αντιπροσωπεύουν τους πιλότους και το κέντρο ελέγχου της εναέριας κυκλοφορίας, λαβαίνουν μέρος σε μία συνεργατική διεργασία προγραμματισμού του καλύτερου σχεδίου για την προσέγγιση και απομάκρυνση των αεροπλάνων από το αεροδρόμιο. Εδώ η διαπραγμάτευση γίνεται σε συνεργατικό επίπεδο, όπου οι πράκτορες προσπαθούν να λύσουν τις συγκρούσεις μέσω συνεργατικών συζητήσεων, και όχι μέσω ανταγωνιστικών δημοπρασιών.

Τέλος, στα συστήματα MAS είναι εμφανές το πρόβλημα της επικοινωνίας μεταξύ των πρακτόρων, για το οποίο έχουν προταθεί δύο τρόποι επικοινωνίας, η απευθείας επικοινωνία μεταξύ των πρακτόρων και η επικοινωνία μέσω μεσολαβητή [56].

2.2.3 Χρήσεις

Οι τεχνολογίες πρακτόρων έχουν αρχίσει να εφαρμόζονται σε πολλές διαφορετικές εφαρμογές. Αναμένεται μία τεράστια άνθηση στο τομέα σχεδιασμού και ανάπτυξης λογισμικού με χρήση τεχνολογιών πρακτόρων τα προσεχή χρόνια [57].

Μερικές περιοχές και προβλήματα στα οποία έχουν εφαρμοστεί τεχνολογίες πρακτόρων είναι [51,58,59,60,61,62]:

- άντληση πληροφοριών από το Internet
- διασκέδαση
- διαχείριση διεργασιών
- διαχείριση δικτύου
- διαχείριση προσωπικού E-mail
- διαχείριση προσωπικών πληροφοριών
- διαχείριση ροών εργασίας
- διαχωρισμός ηλεκτρονικών ειδήσεων (Usenet Netnews)
- διεπιφάνεια χρήσης (User Interface)
- ηλεκτρονικό εμπόριο
- κατανομή πόρων
- προγραμματισμός συναντήσεων
- πρόταση για αγορά βιβλίων, μουσικής, κλπ.
- ρομποτικά συστήματα.

Στην συγκεκριμένη εργασία, οι πράκτορες χρησιμοποιούνται για τον σχεδιασμό και την ανάπτυξη της συνολικής αρχιτεκτονικής και του αντίστοιχου περιβάλλοντος για την υποστήριξη υπηρεσιών επεξεργασίας εικόνας. Ωστόσο, ένα σημαντικό μέρος της εργασίας αποτελεί ο μηχανισμός διαχείρισης υπολογιστικών πόρων για την βέλτιστη κατανομή των εκτελέσεων των αλγόριθμων τόσο σε σχέση με την καλύτερη αξιοποίηση των διαθέσιμων πόρων ενός δικτύου όσο και σε σχέση με το συνολικό κόστος της προσφερόμενης υπηρεσίας. Ο προτεινόμενος μηχανισμός βασίζεται εξολοκλήρου σε συστήματα πολλαπλών πρακτόρων που παίρνουν μέρος σε δημοπρασίες για την ανάθεση της εκτέλεσης κάθε αλγόριθμου. Η ανταγωνιστικότητα των πρακτόρων μπορεί να ποικίλει ανάλογα με την συγκεκριμένη εφαρμογή του προτεινόμενου περιβάλλοντος. Ο μηχανισμός διαχείρισης πόρων παρουσιάζεται αναλυτικά στο τέταρτο κεφάλαιο, ωστόσο απαραίτητη είναι μια ανασκόπηση σχετικών εργασιών στον χώρο της διαχείρισης πόρων και των οικονομικών μοντέλων, όπως παρουσιάζονται στις επόμενες παραγράφους.

2.3 Διαχείριση Πόρων και Οικονομικά Μοντέλα

Τα τελευταία χρόνια ο τρόπος χρήσης των υπολογιστών έχει μεταφερθεί από τα μεγάλα κεντρικά συστήματα (mainframes) σε δίκτυα από προσωπικούς υπολογιστές. Ένα τέτοιο δίκτυο έχει τη δυνατότητα να προσφέρει συνολικά υψηλότερη απόδοση, μεγαλύτερη εμπιστοσύνη και ευκολότερη επεκτασιμότητα με μικρότερο κόστος. Για να υλοποιηθούν όμως αυτές οι δυνατότητες, πρέπει οι σχεδιαστές να λύσουν το πρόβλημα της διαχείρισης των πόρων του κατανεμημένου συστήματος με τρόπο που αυτοί να αξιοποιούνται κατά το βέλτιστο.

2.3.1 Μερικοί Ορισμοί

Ως **πόρος** ορίζεται κάθε επαναχρησιμοποιήσιμο, σχετικά σταθερό κομμάτι υλικού ή λογισμικού ενός υπολογιστικού συστήματος το οποίο είναι χρήσιμο στους χρήστες ή στις διεργασίες τις οποίες αυτοί εκτελούν [63]. Οι πόροι ενός υπολογιστικού συστήματος διακρίνονται στις παρακάτω δύο κατηγορίες:

1. **Φυσικοί πόροι**: οι οποίοι είναι μόνιμα φυσικά κομμάτια ενός υπολογιστικού συστήματος, όπως είναι ο επεξεργαστής, η κυρίως μνήμη, τα περιφερειακά εισόδου και εξόδου, δίσκοι, και το δίκτυο.

2. **Λογικοί πόροι**: (αναφέρονται και ως πόροι λογισμικού) είναι συλλογές πληροφορίας, όπως είναι οι διεργασίες, αρχεία, κοινά προγράμματα, κλπ., οι οποίες αποθηκεύονται σε φυσικούς πόρους (κυρίως μνήμη και λοιπά αποθηκευτικά μέσα).

Η διαχείριση των πόρων αναφέρεται σε ένα σύνολο από έννοιες και υποχρεώσεις όπως συνοψίζονται παρακάτω [63]:

- προγραμματισμός και οργάνωση της διάθεσης των πόρων, καθώς και καταγραφή της τοποθεσίας των πόρων, της δυνατότητας χρήσης και του κόστους χρήσης τους,

- έλεγχος της χρήσης και πρόσβασης στον πόρο σύμφωνα με τους κανόνες πρόσβασης και την συνάρτηση βελτιστοποίησης χρήσης,

- διαδικασία της βεβαίωσης ότι οι πόροι παραμένουν προσιτοί και ότι λειτουργούν κανονικά, και όταν κάτι τέτοιο δεν μπορεί να επιτευχθεί, η βεβαίωση ότι θα υπάρχουν κατάλληλα σήματα που να υποδεικνύουν το σφάλμα.

Η διαχείριση πόρων πρέπει να υποστηρίζει τη χρήση των πόρων με ένα εύκολο, αποτελεσματικό και δίκαιο τρόπο. Για τον σκοπό αυτό έχουν αναπτυχθεί στρατηγικές για το ελεγχόμενο μοίρασμα των πόρων στις διάφορες διεργασίες προς εκτέλεση.

2.3.2 Στρατηγικές Κατανομής Φόρτου Εργασίας

Πολλοί ερευνητές έχουν μελετήσει έξυπνες στρατηγικές για την κατανομή του φόρτου εργασίας σε ένα δίκτυο από υπολογιστές, δηλαδή στρατηγικές για την λήψη απόφασης για την εκτέλεση μιας εισερχόμενης διεργασίας τοπικά ή απομακρυσμένα, και αν απομακρυσμένα σε ποιον ακριβώς κόμβο [64,65,66,67]. Οι στρατηγικές κατανομής φόρτου εργασίας ταξινομούνται με διάφορους τρόπους: ως προς την προσέγγιση για την βέλτιστη κατάσταση του συστήματος, ως προς τον τρόπο λήψης των αποφάσεων (κεντρικά ή κατανεμημένα), και ως προς τον τρόπο συλλογής πληροφοριών για την κατάσταση του συστήματος. Μια σύντομη αναφορά στις διάφορες ταξινομήσεις δίνεται παρακάτω.

Οι υπάρχουσες προσεγγίσεις στο πρόβλημα της κατανομής πόρων ανήκουν σε ένα φάσμα λύσεων που βρίσκεται ανάμεσα σε δύο άκρα [68]. Στο ένα άκρο βρίσκεται η αλγοριθμική προσέγγιση, όπου προϋπόθεση είναι ο αρχικός ορισμός της απαιτούμενης συμπεριφοράς του συστήματος συναρτήσει των διαθέσιμων υπολογιστικών πόρων και των διάφορων περιορισμών στη χρήση τους, καθώς και της αρχιτεκτονικής του συστήματος. Ο αλγόριθμος κατανομής, είτε κεντρικοποιημένος με πλήρες σύνολο πληροφοριών για την κατάσταση του συστήματος, είτε κατανεμημένος με ελλιπείς συνήθως πληροφορίες, προσπαθεί να βελτιστοποιήσει κάποιο συνολικό μέτρο της απόδοσης του συστήματος (π.χ. μέσος χρόνος απόκρισης, διαμεταγωγή, κλπ.). Αυτή η προσέγγιση εγγυάται την βέλτιστη διαχείριση των πόρων βασισμένη στο γεγονός ότι στο σύστημα έχουν ενσωματωθεί ένα σύνολο περιορισμοί που αντιστοιχούν σε συγκεκριμένες προϋποθέσεις για την κατάσταση του συστήματος.

Στο άλλο άκρο του φάσματος βρίσκεται η προσέγγιση των πρακτόρων. Εδώ αντί να οριστεί η επιθυμητή συμπεριφορά του συστήματος κάτω από ορισμένες προϋποθέσεις και περιορισμούς, ορίζονται πρωτόκολλα και μηχανισμοί με βάση τα οποία αυτόνομοι πράκτορες μπορούν να επικοινωνήσουν και να συνδιαλαγούν έτσι ώστε να πετύχουν μια επιθυμητή συμπεριφορά. Αυτή η προσέγγιση είναι πιο ευέλικτη και επιτρέπει τη δυναμική αλλαγή των χαρακτηριστικών κάθε πράκτορα καθώς και του γενικότερου συστήματος.

Εδώ πρέπει να σημειωθεί ότι τα σημερινά κατανεμημένα συστήματα που σχετίζονται με υπηρεσίες επεξεργασίας δεδομένων διακρίνονται για την πολυπλοκότητά τους, η οποία οφείλεται σε παράγοντες όπως το μέγεθος συστήματος (αριθμός χρηστών και μηχανημάτων), η ανομοιογένεια και πολυπλοκότητα εφαρμογών και συστημάτων, καθώς και η δυναμικότητα του περιβάλλοντος ως προς το μέγεθος, τη σύνθεση, το βαθμό απασχόλησης, κλπ. Λαμβάνοντας υπόψη την αυξανόμενη πολυπλοκότητα και την δυναμικότητα των σημερινών κατανεμημένων συστημάτων και εφαρμογών, είναι πολύ δύσκολο να οριστεί ένα αποδεκτό μέτρο για την συνολική απόδοση. Αυτό καθιστά μη πρακτικούς τους τρόπους για ανάθεση πόρων της πρώτης προσέγγισης.

Ανεξάρτητα από την παραπάνω ταξινόμηση, οι στρατηγικές ανάθεσης πόρων διακρίνονται σε κεντρικοποιημένες και κατανεμημένες [69]. Οι κεντρικοποιημένες στρατηγικές περιορίζουν το κέντρο λήψης αποφάσεων σε ένα κεντρικό μέρος, και ταιριάζουν καλύτερα στον προγραμματισμό αλληλεξαρτημένων διεργασιών. Αντίθετα οι κατανεμημένες στρατηγικές μοιράζουν τις ευθύνες σε πολλά μέρη, και προτιμούνται στον προγραμματισμό ανεξάρτητων μεταξύ τους διεργασιών καθώς και διεργασιών των οποίων η αίτηση για

εκτέλεση μπορεί να προέρχεται από οποιοδήποτε κόμβο του συστήματος. Πρέπει να σημειωθεί ότι οι διεργασίες σε συστήματα επεξεργασίας δεδομένων είναι ανεξάρτητες μεταξύ τους, οπότε στη συγκεκριμένη περίπτωση ενδείκνυται η χρήση κυρίως κατανεμημένων μεθόδων.

Τέλος, μια άλλη ταξινόμηση λαμβάνει υπόψη το είδος και τον τρόπο συλλογής της πληροφορίας για την κατάσταση του κατανεμημένου συστήματος, χωρίζοντας τις στρατηγικές κατανομής φόρτου εργασίας σε στατικές και δυναμικές. Οι στατικές στρατηγικές αποφασίζουν σε ποιον κόμβο θα εκτελέσουν κάθε εισερχόμενη διεργασία χωρίς γνώση της τρέχουσας κατάστασης του συστήματος. Είναι ταιριαστές στον προγραμματισμό διεργασιών με προβλέψιμη χρήση των διαφόρων πόρων, σε ένα σύστημα με κόμβους που έχουν αναμενόμενο διακυματισμό στο φόρτο εργασίας τους.

Αντίθετα, οι δυναμικές στρατηγικές λαμβάνουν αποφάσεις σε πραγματικό χρόνο, χρησιμοποιώντας διάφορους δείκτες φόρτου εργασίας για να διαφοροποιήσουν τους υψηλά από τους χαμηλά φορτωμένους κόμβους του συστήματος. Οι δυναμικές στρατηγικές διαφοροποιούνται στη συνέχεια σε διακοπτόμενες και μη-διακοπτόμενες. Οι πρώτες επιτρέπουν την αποδήμηση μιας διεργασίας η οποία έχει ήδη ξεκινήσει την εκτέλεση της. Έτσι, αν οι συνθήκες φόρτου αλλάξουν δραματικά κατά τη διάρκεια της εκτέλεσης, οι διακοπτόμενες στρατηγικές επιτρέπουν τη διόρθωση μιας άσχημης κατανομής διεργασίας. Αυτό δεν είναι δυνατό στις μη-διακοπτόμενες στρατηγικές. Να σημειωθεί ότι στην γενική περίπτωση, ένα κατανεμημένο σύστημα επεξεργασίας δεδομένων χαρακτηρίζεται από ένα ιδιαίτερα δυναμικό και ετερογενές περιβάλλον, με αποτέλεσμα να απαιτεί και ανάλογη δυναμικότητα και ευελιξία από την στρατηγική ανάθεσης πόρων.

Για τον σχεδιασμό μηχανισμών για την κατανομή φόρτου εργασίας προτείνονται οι παρακάτω κατευθυντήριες οδηγίες [68]:

- κομμάτιασμα ενός μεγάλου πολύπλοκου προβλήματος κατανομής σε μικρότερα, ανεξάρτητα προβλήματα,
- αποκεντρικοποίηση της πρόσβασης στους πόρους και στους μηχανισμούς ελέγχου,
- σχεδιασμός επεκτάσιμων αρχιτεκτονικών για την πρόσβαση στους πόρους σε ένα πολύπλοκο σύστημα,
- παροχή εγγυήσεων απόδοσης στους χρήστες και στις εφαρμογές, όπως μέσος χρόνος απόκρισης, διαμεταγωγή, πιθανότητα σφάλματος, μέγιστος χρόνος απόκρισης,
- καθορισμός κριτηρίων απόδοσης συστήματος τα οποία αντικατοπτρίζουν τα διαφορετικά κριτήρια των χρηστών και των εφαρμογών,
- σχεδιασμός ενός ομοιόμορφου περιβάλλοντος μέσω του οποίου οι χρήστες έχουν διάφανη πρόσβαση στις υπηρεσίες του κατανεμημένου συστήματος (η πολυπλοκότητα των πολλαπλών παροχών πόρων και των διάφορων πολιτικών δεν πρέπει να είναι ορατή για τον χρήστη).

2.3.3 Οικονομικά Μοντέλα

Μια σχετικά καινούρια προσέγγιση για την ανάπτυξη μηχανισμών κατανομής φόρτου εργασίας αποτελεί η χρήση τεχνικών από το χώρο των οικονομικών, αυτό που στη διεθνή βιβλιογραφία αναφέρεται με τον όρο 'αγοροστραφής' προγραμματισμός (market oriented programming) [70]. Η κύρια ιδέα του αγοροστραφή προγραμματισμού είναι η επίλυση ενός προβλήματος ανάθεσης φόρτου εργασίας σε κατανεμημένους πόρους σχηματίζοντας μία υπολογιστική οικονομία και βρίσκοντας το σημείο της ανταγωνιστικής ισορροπίας. Για την περιγραφή ενός προβλήματος με βάση το μοντέλο μιας υπολογιστικής οικονομίας, πρέπει να γίνει μια αντιστοίχηση των στοιχείων του δεδομένου προβλήματος σε όρους και έννοιες όπως η κατανάλωση και η παραγωγή αγαθών, καθώς και να οριστεί ένα σύνολο από πράκτορες που επιλέγουν στρατηγικές για την κατανάλωση και παραγωγή βασισμένοι στις ικανότητες τους και στις προτιμήσεις τους, καθώς και στις τιμές της αγοράς.

Το βασικό κίνητρο για τη χρήση του αγοροστραφή προγραμματισμού είναι η εκμετάλλευση των διαφόρων πλαισίων και θεωριών που έχουν αναπτυχθεί στην οικονομική θεωρία ως ένα σχεδιαστικό εργαλείο για την ανάπτυξη πολύ-πρακτορικών συστημάτων. Συγκεκριμένα, σημαντική βοήθεια για την μείωση της πολυπλοκότητας και της επίτευξης της αποκεντρικοποίησης του ελέγχου των πόρων προσφέρουν τα μαθηματικά μοντέλα οικονομικής θεωρίας, τα οποία δίνουν νέες απόψεις για την λύση προβλημάτων κατανομής πόρων [71].

Σε μία πραγματική οικονομία, η αποκεντρικοποίηση επιτυγχάνεται με βάση το γεγονός ότι τα οικονομικά μοντέλα που χρησιμοποιούνται αποτελούνται από πράκτορες που εγωιστικά προσπαθούν να επιτύχουν τους στόχους τους. Εκεί υπάρχουν δύο ειδών πρακτόρων, οι προμηθευτές και οι καταναλωτές. Ο καταναλωτής προσπαθεί να βελτιώσει τα προσωπικά του κριτήρια, αποκτώντας τα προϊόντα που χρειάζεται και δεν ενδιαφέρεται για την συνολική απόδοση του συστήματος. Αντίθετα, ο προμηθευτής πουλάει τα προϊόντα του στους καταναλωτές με μοναδικό στόχο τη μεγιστοποίηση των κερδών του από την κατάλληλη πώληση των προϊόντων στους πελάτες του. Οι πράκτορες καταναλωτές αρχικά έχουν ένα αρχικό ποσό από αγαθά και αναμιγνύονται σε αγοραπωλησίες με σκοπό να μεγιστοποιήσουν την ικανοποίηση τους. Οι πράκτορες παραγωγοί χρησιμοποιούν μία τεχνολογία με την οποία μπορούν να μετατρέψουν κάποιο αγαθό σε κάποιο άλλο (υπολογιστικό χρόνο σε χρήματα, κλπ.). Ο μοναδικός στόχος των παραγωγών είναι να επιλέξουν τις ενέργειες που θα εκτελέσουν με την υπάρχουσα τεχνολογία που έχουν στην κατοχή τους ώστε να μεγιστοποιήσουν τα κέρδη τους. Από την μεριά των πρακτόρων, η κατάσταση του κόσμου στον οποίο βρίσκονται περιγράφεται πλήρως με τις τρέχουσες τιμές, δηλαδή οι τιμές επηρεάζουν καθοριστικά τις συμπεριφορές τους. Σαν αποτέλεσμα, οι πράκτορες δεν χρειάζεται να γνωρίζουν τις προτιμήσεις και τις ικανότητες των υπολοίπων πρακτόρων. Η επικοινωνία μεταξύ τους αποτελείται μόνο από προσφορές για διάφορα αγαθά σε διάφορες τιμές.

Τα περισσότερα οικονομικά μοντέλα εισάγουν την έννοια των χρημάτων σε συνδυασμό με την χρέωση των υπηρεσιών ως τον τρόπο με τον οποίο ελέγχουν την εγωιστική συμπεριφορά των πρακτόρων. Η απόδοση του συστήματος είναι αποτέλεσμα ενός συνδυασμού από τα διάφορα κριτήρια απόδοσης που έχει ο κάθε πράκτορας. Συχνά,

μεγάλα κατανεμημένα συστήματα και δίκτυα υπολογιστών απλώνονται σε πολλούς χώρους, και ο έλεγχος των πόρων μοιράζεται σε πολλούς οργανισμούς που κατέχουν καθορισμένα κομμάτια του συνολικού δικτύου. Σε ένα τέτοιο περιβάλλον, κάθε οργανισμός έχει ένα σύνολο από υπηρεσίες που παρέχει. Οι οικονομικές αρχές της χρέωσης και η χρήση ανταγωνιστικών μοντέλων μπορούν να δώσουν ενδιαφέρουσες λύσεις στην δημιουργία αποκεντρωμένων μηχανισμών για τον έλεγχο και παροχή υπηρεσιών στους τελικούς χρήστες.

Τέλος, ένα πολύ βασικό θέμα στο σχεδιασμό αρχιτεκτονικών για την παροχή υπηρεσιών σε μεγάλα υπολογιστικά δίκτυα είναι η επεκτασιμότητα. Με την όλο και αυξανόμενη ζήτηση για νέες υπηρεσίες, υπάρχει άμεση ανάγκη για αρχιτεκτονικές που να υποστηρίζουν την ευέλικτη παροχή υπηρεσιών. Η χρήση οικονομικών μοντέλων παρέχει, έναν φυσικό τρόπο για την ανάπτυξη μηχανισμών βασισμένων στην προσφορά και την ζήτηση των πόρων.

Πολλοί ερευνητές έχουν χρησιμοποιήσει οικονομικά μοντέλα για την επίλυση προβλημάτων κατανομής πόρων, χρησιμοποιώντας δύο διαφορετικές προσεγγίσεις, την μη χρεωστική και την χρεωστική. Η πρώτη, η μη χρεωστική, βασίζεται στη θεωρία παιγνίων για την κατασκευή κατανεμημένων αλγόριθμων και η δεύτερη, η χρεωστική, βασίζεται στην αλληλεπίδραση προμηθευτών και καταναλωτών μέσω μιας αγοράς βασισμένης σε τιμές και χρήματα.

2.3.4 Σχετικές Εργασίες Μηχανισμών Διαχείρισης Πόρων

Με βάση την ανασκόπηση που παρουσιάστηκε στις προηγούμενες παραγράφους, και με δεδομένα τα χαρακτηριστικά ενός περιβάλλοντος παροχής υπηρεσιών επεξεργασίας όπως αυτά έχουν ήδη παρουσιαστεί, είναι φανερό ότι το περιβάλλον που απασχολεί την συγκεκριμένη εργασία απαιτεί ένα μηχανισμό διαχείρισης πόρων κατανεμημένο και δυναμικό, ενώ αποκλείονται προσεγγίσεις που βασίζονται σε προκαθορισμένα μοντέλα του συστήματος. Κατά συνέπεια, οι επόμενες παράγραφοι εστιάζονται σε μια ανασκόπηση αντιπροσωπευτικών δειγμάτων μηχανισμών διαχείρισης πόρων που έχουν παρόμοια χαρακτηριστικά.

Ένα αντιπροσωπευτικό παράδειγμα των κατανεμημένων ευρηστικών αλγόριθμων για τη διαχείριση υπολογιστικών πόρων παρουσιάζεται στην εργασία των Eager et al [64]. Συνοπτικά, ο αλγόριθμος τους δουλεύει ως εξής: Μία διεργασία ξεκινάει την εκτέλεσή της σε ένα υπολογιστικό κόμβο. Αν ο κόμβος δεν είναι φορτωμένος, συνεχίζεται η εκτέλεση τοπικά. Αν ο κόμβος είναι υπέρ-φορτωμένος, ο τοπικός διαχειριστής επικοινωνεί με τους υπόλοιπους κόμβους διερευνώντας την κατάσταση του φορτίου τους. Αν κάποιος κόμβος δεν είναι φορτωμένος, η διεργασία στέλνεται σε αυτόν. Ο κόμβος παραλήπτης πρέπει να επεξεργαστεί τη διεργασία, ακόμη και αν κατά τη στιγμή της άφιξής της είναι πια υπέρ-φορτωμένος. Ένα βασικό μειονέκτημα του αλγόριθμου είναι ότι δεν είναι ευέλικτος σε μεταβλητές συνθήκες, και επιπλέον παρουσιάζει σημαντικά προβλήματα όταν πολλοί κόμβοι αποφασίσουν να στείλουν μία διεργασία σε έναν συγκεκριμένο κόμβο.

Τα παραπάνω προβλήματα προσπαθεί να λύσει η μέθοδος για έξυπνο αποκεντρικοποιημένο έλεγχο σε ένα μεγάλο κατανεμημένο σύστημα του Pasquale [72]. Σε κάθε κόμβο υπάρχει ένας πράκτορας ο οποίος χειρίζεται τις τοπικές αιτήσεις και αποφασίζει σε ποιον κόμβο θα ανατεθεί η εκτέλεση κάθε διεργασίας. Η απόφαση του εξαρτάται από πολλούς παράγοντες, όπως ο τοπικός φόρτος εργασίας, η ιστορία των γειτονικών κόμβων, η ιστορία της διεργασίας προς εκτέλεση, καθώς και η σιγουριά που έχει για την κατάσταση των γειτονικών κόμβων. Ο κάθε πράκτορας κατασκευάζει ένα μοντέλο για την κατάσταση του κάθε άλλου πράκτορα, το χρησιμοποιεί για να υπολογίσει σε τι κατάσταση βρίσκεται, ώστε να ελαττώσει το επιπλέον κόστος και την καθυστέρηση της επικοινωνίας για την αποσαφήνιση της κατάστασης του καθενός. Ωστόσο, το μοντέλο που παρουσιάζεται είναι αρκετά δύσκολο να υλοποιηθεί στην πραγματικότητα ή να αντεπεξέλθει σε απρόβλεπτες αλλαγές της τοπολογίας και σύνθεσης του συστήματος, μια και η απόφαση του κάθε πράκτορα βασίζεται σε πολύπλοκη γνώση όχι μόνο για τη δική του κατάσταση αλλά και για την κατάσταση όλων των άλλων πρακτόρων.

Μία διαφορετική προσέγγιση παρουσιάζεται από τους Kurose et al [73]. Στην εργασία τους κάνουν χρήση μη χρεωστικών οικονομικών μοντέλων για την ανάπτυξη κατανεμημένων αλγόριθμων για την διαχείριση πόρων (που στην συγκεκριμένη περίπτωση είναι αρχεία ή συστήματα αρχείων) με πράκτορες που είναι συνεργατικοί. Κάθε πράκτορας υπολογίζει τοπικά μια συνάρτηση χρησιμότητας με κριτήρια το κόστος επικοινωνίας και τον χρόνο επεξεργασίας. Στη συνέχεια η τιμή της συνάρτησης ανακοινώνεται σε όλους τους άλλους πράκτορες. Με βάση το μέσο όρο των τιμών συνάρτησης χρησιμότητας, ο κάθε πράκτορας μεταβάλει την κατανομή των πόρων του. Η όλη διαδικασία επαναλαμβάνεται μέχρι να υπάρξει ικανοποιητική σύγκλιση. Το βασικό πρόβλημα αυτής της προσέγγισης είναι το μεγάλο κόστος επικοινωνίας μέχρι να επέλθει σύγκλιση, γεγονός που καθιστά μη πρακτική την εφαρμογή της σε περιβάλλοντα που παρουσιάζουν δυναμικές αλλαγές (εκτός από την περίπτωση ιδιαίτερα αργών ρυθμών μεταβολής).

Ένα άλλο γενικό μοντέλο συνεργατικών πρακτόρων είναι το Contract Net που αναπτύχθηκε από τους Davis και Smith [74]. Η κατανομή των εργασιών αποτελεί μία μορφή διαβουλεύσεων για την ανάθεση έργου. Όταν ένας πράκτορας αποφασίσει ότι οι πόροι του δεν είναι ικανοποιητικοί για την εκτέλεση μιας διεργασίας, ανακοινώνει μία αίτηση για απομακρυσμένη εκτέλεση της εργασίας και γίνεται ο διαχειριστής της. Οι υπόλοιποι πράκτορες του συστήματος μπορούν να εξετάσουν την ανακοίνωση και να αποφασίσουν αν ενδιαφέρονται γι' αυτή με βάση την κατάστασή τους τη δεδομένη στιγμή. Οι πράκτορες που ενδιαφέρονται για την εργασία που ανακοινώθηκε, στέλνουν τις προσφορές τους στον διαχειριστή. Ο διαχειριστής της εργασίας αξιολογεί τις προσφορές και αποφασίζει σε ποιους πράκτορες θα την αναθέσει (το κριτήριο για την αξιολόγηση εξαρτάται άμεσα από το συγκεκριμένο χώρο εφαρμογών). Στη συνέχεια αυτοί που κέρδισαν μία ανάθεση έργου μπορούν να προσχωρήσουν στο διαχωρισμό του έργου σε μικρότερα, και να επαναληφθεί η διεργασία ανάθεσης έργου από την αρχή.

Υπάρχουν διάφορες υλοποιήσεις και επεκτάσεις του παραπάνω μοντέλου. Συγκεκριμένα, ο Sandholm το χρησιμοποίησε για να κατασκευάσει ένα σύστημα για την ανάθεση εργασιών μεταφοράς φορτίων σε ένα δίκτυο από εταιρείες μεταφορών [75,76]. Επίσης, ο Malone [77] κατασκεύασε ένα σύστημα για τον προγραμματισμό διεργασιών σε δίκτυο

προσωπικών υπολογιστών. Αντί να χρησιμοποιεί την παραδοσιακή άποψη όπου ο κάθε υπολογιστής χρησιμοποιείται μόνο από το κάτοχο του, προσπαθεί να αντιστοιχίσει διεργασίες που βρίσκονται σε αναμονή σε κάποιον ελεύθερο πόρο του συστήματος. Ο στόχος είναι η ελαχιστοποίηση της συνολικής αναμονής των διεργασιών στο σύστημα. Το κατανεμημένο πρωτόκολλο προγραμματισμού διεργασιών του ακολουθεί τα εξής βήματα. Ένας πράκτορας (αντίστοιχος με το διαχειριστή στο Contract Net) ανακοινώνει μία διεργασία με ένα επίπεδο προτεραιότητας, την περιγραφή των απαιτούμενων πόρων και αρκετή πληροφορία ώστε οι διάφοροι υπολογιστές να εκτιμήσουν τον αναμενόμενο χρόνο εκτέλεσης. Οι υπολογιστές που δεν χρησιμοποιούνται τη δεδομένη στιγμή κάνουν μία προσφορά ανακοινώνοντας τον αναμενόμενο χρόνο για την εκτέλεση της κάθε διεργασίας. Ο διαχειριστής αξιολογεί τις προσφορές και αναθέτει τη διεργασία στον πιο κατάλληλο υπολογιστή. Η παραπάνω μέθοδος μπορεί να αντιμετωπίσει δυναμικά ως προς τις διεργασίες και την κατάσταση των κόμβων περιβάλλοντα με σχετικά μικρό κόστος σε επικοινωνίες. Ωστόσο, υπάρχει η απαίτηση ο κάθε πράκτορας να έχει πληροφορίες για την ύπαρξη όλων των άλλων πρακτόρων και να επικοινωνεί μ' αυτούς για κάθε δημοπρασία. Αυτό κάνει μη πρακτική την εφαρμογή του μοντέλου για περιβάλλοντα με δυναμικές μεταβολές στην τοπολογία και σύνθεση των κόμβων.

Ένα αντίστοιχο, αλλά χρεωστικό, μοντέλο παρουσιάζεται από τους Ferguson et al [68,78]. Προϋποθέτουν ένα δίκτυο από επεξεργαστές με ορισμένη απόδοση, οι οποίοι συνδέονται με προκαθορισμένες συνδέσεις με συγκεκριμένη καθυστέρηση μετάδοσης δεδομένων. Οι πόροι που συμμετέχουν στις αγοραπωλησίες είναι ο χρόνος του επεξεργαστή και το εύρος διαμεταγωγής για την επικοινωνία. Σε κάθε διεργασία προς εκτέλεση διατίθεται ένα εικονικό ποσό χρημάτων, το οποίο το χρησιμοποιεί για να αγοράσει διάφορες υπηρεσίες που χρειάζεται, όπως υπολογιστική ισχύ και εύρος διαμεταγωγής στο δίκτυο. Κάθε διεργασία προσπαθεί να ελαχιστοποιήσει το κόστος και το χρόνο εκτέλεσης της. Ο βασικός μηχανισμός της οικονομίας που παρουσιάζεται, βασίζεται σε δημοπρασίες που εκτελούνται από τους διάφορους επεξεργαστές που συμμετέχουν στο σύστημα και δέχονται προσφορές από τις διάφορες διεργασίες που θέλουν να εκτελεστούν στο σύστημα. Και πάλι το βασικό μειονέκτημα του μοντέλου είναι η απαίτηση για προκαθορισμένη τοπολογία του κατανεμημένου συστήματος.

Τέλος, πρέπει να αναφερθεί μια ιδιαίτερα πρόσφατη εργασία των Chavez et al [62], που προτείνει ένα μοντέλο βασισμένο στην εργασία των Davis και Smith καθώς και του Malone, το οποίο αποτελείται από διάφορους πράκτορες που βρίσκονται σε κάθε υπολογιστικό κόμβο που ανήκει στο κατανεμημένο σύστημα. Ο κάθε πράκτορας διαχειρίζεται μόνο τους τοπικούς πόρους (χρόνο επεξεργαστή). Για την ανάθεση των διεργασιών σε κάθε πράκτορα, το σύστημα χρησιμοποιεί ένα μοντέλο δημοπρασιών. Πριν την ανάθεση μιας διεργασίας γίνεται μία δημοπρασία, όπου ο κάθε πράκτορας κάνει την προσφορά του, και στο τέλος αξιολογούνται όλες οι προσφορές και ανατίθεται η διεργασία στον καλύτερο πράκτορα. Οι πράκτορες έχουν την δυνατότητα εκμάθησης ώστε να μπορούν να αποφασίσουν δυναμικά πότε να εκτελέσουν μία διεργασία τοπικά και πότε όχι, ανάλογα με τις καθυστερήσεις που παρουσιάζονται στο δίκτυο. Στην εργασία δεν αναφέρονται τρόποι με τους οποίους υπολογίζεται ο αναμενόμενος χρόνος μιας διεργασίας και δεν λαμβάνονται υπόψη τα πιθανά δεδομένα εισόδου και εξόδου που χρειάζεται η κάθε διεργασία.

ΚΕΦΑΛΑΙΟ 3 Αρχιτεκτονική Περιβάλλοντος

Οι έννοιες της υπηρεσίας καθώς και της παροχής υπηρεσιών, όπως έχουν παρουσιαστεί στο εισαγωγικό κεφάλαιο, μπορούν να χρησιμοποιηθούν για την περιγραφή ενός μεγάλου συνόλου ενεργειών που λαμβάνουν μέρος στην καθημερινή μας ζωή, στα πλαίσια της συναναστροφής με άλλους ανθρώπους, της εργασίας, της αλληλεπίδρασης με τους διάφορους κρατικούς και ιδιωτικούς οργανισμούς, της αλληλεπίδρασης με τα διάφορα υπολογιστικά συστήματα. Οι διάφορες μορφές που παίρνουν οι παρεχόμενες υπηρεσίες δεν τις διαφοροποιούν μεταξύ τους ως προς τις ανάγκες και τον τρόπο υποστήριξής τους από πληροφοριακά συστήματα. Συνεχίζουν να αποτελούν υπηρεσίες, οι οποίες δέχονται κάποια δεδομένα, τα επεξεργάζονται και παράγουν κάποιο αποτέλεσμα (το οποίο πιθανά να χρησιμοποιηθεί από κάποια άλλη υπηρεσία), και λειτουργούν σε ένα εκτεταμένο, ετερογενές κατανεμημένο περιβάλλον.

Η αρχιτεκτονική που παρουσιάζεται στις επόμενες παραγράφους μπορεί να χρησιμοποιηθεί ως βάση για την ανάπτυξη συστημάτων για την παροχή υπηρεσιών επεξεργασίας δεδομένων σε διάφορους χώρους εφαρμογών. Στην συγκεκριμένη εργασία η προτεινόμενη αρχιτεκτονική χρησιμοποιείται για την ανάπτυξη ενός περιβάλλοντος παροχής υπηρεσιών επεξεργασίας εικόνων. Για την απλούστευση της περιγραφής της αρχιτεκτονικής, στη συνέχεια της εργασίας θα περιοριστούμε κυρίως στην έννοια της υπηρεσίας όπως αυτή εμφανίζεται στο στενότερο χώρο της επεξεργασίας εικόνων.

Συνοψίζοντας τις έννοιες και λειτουργίες που συνιστούν τις υπηρεσίες επεξεργασίας δεδομένων, το ζητούμενο είναι ένα περιβάλλον που να προσφέρει στον χρήστη, είτε αυτός είναι ένας εξειδικευμένος ερευνητής είτε είναι ένας απλός χρήστης, τη δυνατότητα επεξεργασίας εικόνων χρησιμοποιώντας με διαφάνεια (ως προς την διεύθυνση, πρόσβαση και εκτέλεση) την υπολογιστική ισχύ και τους αλγόριθμους επεξεργασίας ενός ευρύτερου ετερογενούς δικτύου. Επιπλέον, πρέπει να παρέχει εύκολη και ενιαία πρόσβαση σε διάφορους μεμονωμένους αλγόριθμους επεξεργασίας εικόνων που έχουν αναπτυχθεί στο παρελθόν και δεν δίνουν τη δυνατότητα επαναχρησιμοποίησής τους με τρόπο εύκολο, λειτουργικό και ενιαίο. Επίσης, απαιτούνται μηχανισμοί για την βέλτιστη χρήση των διαθέσιμων πόρων, και την προσφορά υπηρεσιών με ποιότητα ανάλογη με το εικονικό ποσό που ο χρήστης είναι διατεθειμένος να πληρώσει για κάθε χρήση. Επιγραμματικά, στόχος της εργασίας είναι ο σχεδιασμός αρχιτεκτονικής και η ανάπτυξη περιβάλλοντος που να υποστηρίζει τις παρακάτω λειτουργίες:

- εκτέλεση αλγόριθμων επεξεργασίας εικόνων σε κατανεμημένο σύστημα,
- σταθερότητα και συνέχεια εκτέλεσης,
- διάφανη εκτέλεση αλγόριθμων σε ετερογενές κατανεμημένο σύστημα,
- βέλτιστη χρήση των διαθέσιμων υπολογιστικών πόρων,
- χρέωση υπηρεσιών,
- ενσωμάτωση λογισμικού και εκτελέσιμων που έχουν κατασκευαστεί από τρίτους,

- δυνατότητα μελέτης και αξιολόγησης αλγόριθμων,
- υποστήριξη εκτέλεσης ακολουθιών αλγόριθμων, με ευέλικτη κατανομή εργασίας στους διαθέσιμους υπολογιστικούς πόρους,
- δυνατότητα επέκτασης των λειτουργιών και της φυσικής έκτασης του περιβάλλοντος,
- δυνατότητα διασύνδεσης με άλλες εφαρμογές.

3.1 Ζητούμενα Χαρακτηριστικά και Ανάγκες

Ένα περιβάλλον που υποστηρίζει υπηρεσίες επεξεργασίας δεδομένων όπως ορίζονται αναλυτικά στην προηγούμενη παράγραφο, πρέπει να είναι καταρχήν επεκτάσιμο όσο αναφορά στο είδος και στη λειτουργικότητα των υπηρεσιών, αλλά και στη γεωγραφική έκταση που καλύπτει. Παράλληλα πρέπει να μπορεί να χρησιμοποιηθεί ταυτόχρονα από ένα μεγάλο αριθμό χρηστών, οι οποίοι μπορεί να ανήκουν σε διαφορετικούς οργανισμούς με πολύπλοκη ιεραρχία.

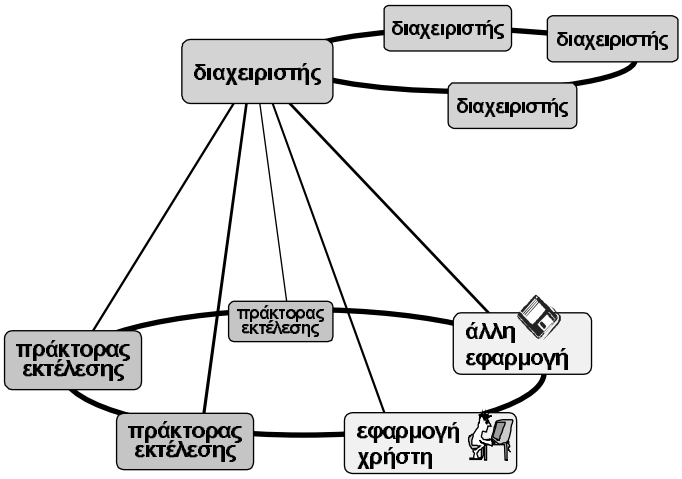
Λαμβάνοντας υπόψη μας τις ειδικές ανάγκες που μπορεί να έχουν διαφορετικές υπηρεσίες, όπως τον πολύ μεγάλο απαιτούμενο χρόνο για την διεκπεραίωση μιας αίτησης, το περιβάλλον πρέπει να εγγυάται τη συνέχεια παροχής υπηρεσιών (session persistency). Επίσης, για να επιτρέπει την ελεύθερη ενσωμάτωση αλγόριθμων που έχουν κατασκευαστεί από τρίτους ή βρίσκονται ακόμα στο στάδιο ανάπτυξης (και συνεπώς μπορεί να μη συμπεριφέρονται πάντα με τον αναμενόμενο σωστό τρόπο), το περιβάλλον πρέπει να σχεδιαστεί με τρόπο τέτοιο που να διαβεβαιώνει και να εγγυάται ένα ελάχιστο σίγουρο επίπεδο ποιότητας κατά την παροχή υπηρεσιών κάτω από οποιεσδήποτε συνθήκες (robustness).

Επιπρόσθετα, πρέπει να μπορεί να υλοποιηθεί χωρίς πολύ επιπλέον κόπο σε διαφορετικά μηχανήματα (διαφορετικό υλικό) με διαφορετικά λειτουργικά συστήματα (διαφορετικό λογισμικό), πιθανά χρησιμοποιώντας διαφορετικούς τρόπους διασύνδεσης των εφαρμογών καθώς και διαφορετική δικτυακή υποδομή. Μια τελική απαίτηση είναι η δυνατότητα να μπορεί να συνεργαστεί σε διαφορετικά επίπεδα με άλλα πληροφοριακά συστήματα.

3.2 Επισκόπηση Αρχιτεκτονικής

Λαμβάνοντας υπόψη μας τις διάφορες απαιτήσεις για επεκτασιμότητα, ευελιξία ως προς στρατηγικές, σταθερότητα και συνέχεια στην παροχή υπηρεσιών, η εργασία προτείνει μία αρθρωτή αρχιτεκτονική που βασίζεται στη συνεργασία αυτόνομων πρακτόρων. Ο πυρήνας της προτεινόμενης αρχιτεκτονικής αποτελείται από μία κοινότητα πρακτόρων εκτέλεσης και ενός κεντρικού πράκτορα, του διαχειριστή, οι οποίοι βρίσκονται διασκορπισμένοι σε

ένα δίκτυο υπολογιστών [**Σχήμα 3**]. Η κοινότητα αυτή δέχεται αιτήσεις για την έναρξη διαφόρων επεξεργασιών εικόνων από τους χρήστες του περιβάλλοντος, οι οποίοι είναι συνδεμένοι με το περιβάλλον μέσω ειδικών εφαρμογών. Επιπλέον, οι αιτήσεις για επεξεργασία μπορεί να προέρχονται και από οποιαδήποτε άλλο πρόγραμμα ή εφαρμογή (μέσω κατάλληλης διασύνδεσης βασισμένης είτε σε προκαθορισμένο από το περιβάλλον σύνολο μηνυμάτων είτε σε πρότυπα επικοινωνίας ετερογενών εφαρμογών, όπως εξηγείται στο κεφάλαιο της υλοποίησης).



Σχήμα 3: Ανασκόπηση προτεινόμενης αρχιτεκτονικής.

Σε κάθε υπολογιστή του κατανεμημένου συστήματος που μπορεί να εκτελέσει αλγόριθμους επεξεργασίας αντιστοιχεί κι ένας πράκτορας εκτέλεσης, ο οποίος μπορεί να θεωρηθεί ως μία πολύπλοκη οντότητα με μία πληθώρα διαφορετικών ρόλων. Ο βασικός στόχος του είναι η έναρξη της εκτέλεσης ενός αλγόριθμου επεξεργασίας εικόνας, η παρακολούθηση της διαδικασίας εκτέλεσης κάθε αλγόριθμου, καθώς και η διατήρηση των παραγόμενων δεδομένων μετά το τέλος της εκτέλεσης, για την μελλοντική παράδοσή τους στο νόμιμο παραλήπτη τους, δηλαδή τον χρήστη ή το πρόγραμμα που ζήτησε την έναρξη της εκτέλεσης του συγκεκριμένου αλγόριθμου.

Οι επικοινωνίες των διάφορων πρακτόρων εκτέλεσης οργανώνονται και ελέγχονται από ένα κεντρικό πράκτορα, τον διαχειριστή. Αν και τα περισσότερα μηνύματα ανταλλάσσονται απευθείας ανάμεσα στα διάφορα αυτόνομα κομμάτια (πράκτορες εκτέλεσης και εφαρμογές), ο διαχειριστής παίζει ένα πολύ σημαντικό ρόλο κατά τη διάρκεια της έναρξης μιας εκτέλεσης ενώ παράλληλα συμμετέχει στη φάση της διαχείρισης των διάφορων πόρων (ανάθεση διεργασίας και ανάκληση αλγόριθμου). Τέλος ο διαχειριστής επιτρέπει την

επεκτασιμότητα της τοπικής κοινωνίας των πρακτόρων με άλλες παρόμοιες κοινωνίες στον ίδιο ή άλλους οργανισμούς.

Οι παραπάνω οντότητες αποτελούν τον πυρήνα της αρχιτεκτονικής και υποστηρίζουν τις βασικές υπηρεσίες που προσφέρονται από το περιβάλλον. Συγκεκριμένα, υποστηρίζουν την διάφανη εκτέλεση αλγόριθμων σε ένα ετερογενές δίκτυο υπολογιστών, με μηχανισμούς βελτιστοποίησης κατανομής φόρτου εργασίας και χρέωσης εκτέλεσης, προσφέρουν σταθερότητα του περιβάλλοντος ως προς την εκτέλεση αλγόριθμων, συνέχεια στην παροχή υπηρεσιών εκτέλεσης, δυνατότητα ενσωμάτωσης εκτελέσιμων από τρίτους κατασκευαστές, καθώς και δυνατότητα ολοκλήρωσης με άλλα συστήματα. Λεπτομερέστερη περιγραφή ακολουθεί στις επόμενες παραγράφους.

3.3 Πράκτορας Εκτέλεσης

Ο πράκτορας εκτέλεσης (ΠΕ), αποτελείται από διάφορα λειτουργικά κομμάτια και υποσυστήματα για τη διαχείριση:

- τοπικής συλλογής εκτελέσιμων των αλγόριθμων,
- τοπικής κρυφής μνήμης δεδομένων εισόδου,
- προσωρινού χώρου αποθήκευσης αποτελεσμάτων,
- εκτέλεσης αλγόριθμων,
- τοπικών πόρων και της πολιτικής διάθεσης και χρέωσης χρήσης τους.

όπως φαίνονται και στο **Σχήμα 4**.

Σχήμα 4: Σχηματική αναπαράσταση της δομής του πράκτορα εκτέλεσης. Να σημειωθεί το συγκεκριμένο σχήμα αναφέρεται στη λειτουργική δομή και όχι στο μοντέλο ανάλυσης πρακτόρων όπως αυτό φαίνεται στο Σχήμα 2.

Οι παραπάνω επιμέρους ρόλοι του πράκτορα εκτέλεσης μπορεί να υποτονιστούν ή να ενισχυθούν επιλεκτικά, ανάλογα με τις ανάγκες της συγκεκριμένης υλοποίησης της προτεινόμενης αρχιτεκτονικής. Στην γενική περίπτωση, ο πράκτορας εκτέλεσης μπορεί να θεωρηθεί ως ένα σύνολο από επιμέρους εξειδικευμένους πράκτορες, ο καθένας από τους οποίους αναλαμβάνει έναν από τους ρόλους που περιγράφονται στις επόμενες παραγράφους. Πρέπει να σημειωθεί ότι κάθε πράκτορας εκτέλεσης λαμβάνει ενεργά μέρος στην διαδικασία κατανομής των εκτελέσεων στους πόρους του συστήματος και τον υπολογισμό του κόστους κάθε εκτέλεσης. Ωστόσο, αυτός ο ρόλος του περιγράφεται αναλυτικά στο επόμενο κεφάλαιο.

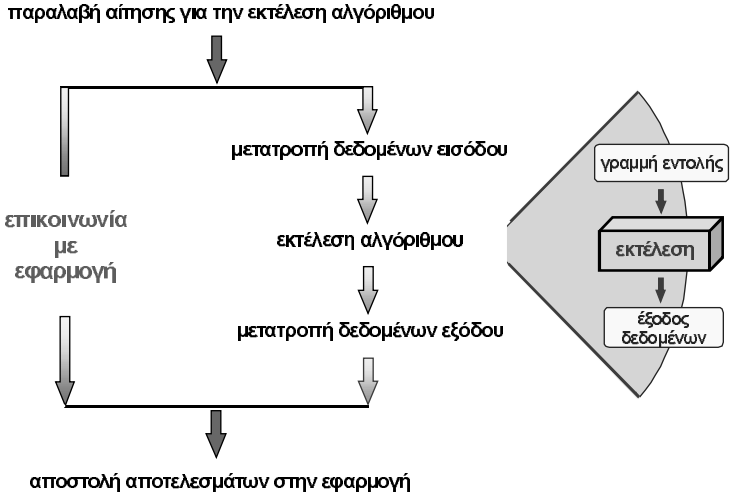
Πριν την αναλυτικότερη περιγραφή των ρόλων του πράκτορα εκτέλεσης, πρέπει να τονίσουμε ότι ακολουθώντας το γενικό μοντέλο για την ανάλυση και περιγραφή πρακτόρων, ο πράκτορας εκτέλεσης μπορεί να αναλυθεί στα τρία λειτουργικά μέρη: αισθητήρες, γνωστικό μέρος και όργανα δράσης (βλέπε δεύτερο κεφάλαιο, σχήμα 2). Καταρχήν, οι αισθητήρες με τους οποίους αντιλαμβάνεται το περιβάλλον του στη συγκεκριμένη περίπτωση είναι μηχανισμοί που του επιτρέπουν να γνωρίζει τον φόρτο του μηχανήματος στο οποίο βρίσκεται, την κατάσταση της τοπικής κρυφής μνήμης, το μέγεθος του διαθέσιμου ελεύθερου χώρου για την προσωρινή αποθήκευση δεδομένων, την κατάσταση των διεργασιών για τις οποίες είναι υπεύθυνος, καθώς και να 'ακούει' μηνύματα από άλλους πράκτορες ή εφαρμογές του χρήστη. Το γνωστικό μέρος αφορά στη δυνατότητα να παίρνει αποφάσεις βασισμένος στα διάφορα σήματα τα οποία λαμβάνει από το περιβάλλον του σε συνδυασμό με τις διάφορες αντιλήψεις που έχει σχετικές με τον τρόπο με τον οποίο συνεργάζεται με τους υπόλοιπους πράκτορες, γνώσεις σχετικές με το δίκτυο, καθώς και πολιτικές τις οποίες πρέπει να εφαρμόσει. Τέλος, το αποτέλεσμα του δεύτερου μέρους καθοδηγεί τα όργανα δράσης, τα οποία στη συγκεκριμένη περίπτωση υλοποιούνται είτε στέλνοντας κάποιο μήνυμα σε κάποιον άλλο πράκτορα ή εφαρμογή ενός χρήστη, είτε ξεκινώντας την εκτέλεση κάποιας διεργασίας, είτε αποθηκεύοντας κάποια δεδομένα στην τοπική κρυφή μνήμη ή στον προσωρινό αποθηκευτικό χώρο. Στις παρακάτω παραγράφους αναλύονται με περισσότερες λεπτομέρειες τα διάφορα λειτουργικά τμήματα του πράκτορα εκτέλεσης.

Εκτέλεση Αλγόριθμων

Ο κύριος ρόλος του πράκτορα εκτέλεσης είναι προφανώς η διαχείριση της εκτέλεσης αλγόριθμων, που υλοποιείται μέσω του εκτελεστή αλγόριθμων, μιας οντότητας που λειτουργεί ως 'περιτύλιγμα' (wrapper) για τα διάφορα εκτελέσιμα αλγόριθμων επεξεργασίας που ενσωματώνονται στο περιβάλλον. Ο εκτελεστής αλγόριθμων μετατρέπει τα δεδομένα εισόδου σε τύπο κατάλληλο για χρήση από τον αλγόριθμο, εκτελεί τον αλγόριθμο, και τέλος, μετά το πέρας της εκτέλεσης, μετατρέπει τα δεδομένα εξόδου του σε τύπο κατάλληλο για χρήση από το πρόγραμμα ή την εφαρμογή που ζήτησε την εκτέλεση, όπως φαίνεται στο Σχήμα 5.

Κατά τη διάρκεια της εκτέλεσης του αλγόριθμου, ο εκτελεστής αλγόριθμων είναι επίσης υπεύθυνος για την ικανοποίηση αιτήσεων από την εφαρμογή του χρήστη, όπως ο

τερματισμός ή το προσωρινό σταμάτημα της εκτέλεσης ενός αλγόριθμου, ή η συνέχιση της εκτέλεσης ενός προσωρινά σταματημένου αλγόριθμου.



Σχήμα 5: Βήματα εκτέλεσης ενός αλγόριθμου μέσω του εκτελεστή αλγόριθμων.

Διαχείριση Τοπικής Συλλογής Αλγόριθμων

Κάθε ΠΕ είναι υπεύθυνος για τη διαχείριση μιας τοπικής βάσης που περιέχει τα εκτελέσιμα των αλγόριθμων που μπορούν να εκτελεστούν στο συγκεκριμένο υπολογιστή, καθώς και τα απαραίτητα στοιχεία και δεδομένα που χρειάζεται κάθε παρεχόμενος αλγόριθμος για να προγραμματιστεί και να συντελεστεί η εκτέλεσή του. Τονίζεται ότι στη βάση που χειρίζεται ο κάθε πράκτορας εκτέλεσης υπάρχουν πληροφορίες και εκτελέσιμα μόνο για τους αλγόριθμους που μπορούν να εκτελεστούν από τον υπολογιστή στον οποίο αντιστοιχεί ο συγκεκριμένος πράκτορας. Γενικά, η βάση αλγόριθμων είναι διαφορετική σε κάθε ΠΕ. Όπως είναι φυσικό, πράκτορες εκτέλεσης που βρίσκονται σε υπολογιστές με διαφορετική αρχιτεκτονική, θα έχουν τελείως διαφορετική συλλογή εκτελέσιμων. Ακόμη και ΠΕ που βρίσκονται σε μηχανήματα με την ίδια αρχιτεκτονική μέσα στον ίδιο οργανισμό, είναι δυνατό να έχουν διαφορετικές συλλογές αλγόριθμων, εξαιτίας πιθανών περιορισμών που ορίζουν οι διαχειριστές του κατανεμημένου συστήματος.

Ένα πολύ σημαντικό χαρακτηριστικό ενός περιβάλλοντος που υποστηρίζει υπηρεσίες επεξεργασίας δεδομένων είναι η δυνατότητα εύκολης εισαγωγής αλγόριθμων επεξεργασίας δεδομένων που προέρχονται από τρίτους κατασκευαστές. Σ' αυτήν την περίπτωση, υπάρχει μόνο το εκτελέσιμο αρχείο για κάποιο συγκεκριμένο λειτουργικό σύστημα, η περιγραφή του τρόπου χρήσης τους, ενώ είναι γνωστός ο τύπος των δεδομένων εισόδου και εξόδου. Στο προτεινόμενο περιβάλλον είναι δυνατή η ενσωμάτωση των προαναφερθέντων τύπων αλγόριθμων μέσω ειδικών λειτουργιών που εκτελεί ο εκτελεστής αλγόριθμων του πράκτορα

εκτέλεσης. Η κατασκευή ενός εκτελεστή αλγόριθμου αρκετά ισχυρού ώστε να μπορεί να διαχειριστεί την πληθώρα των αλγόριθμων επεξεργασίας δεδομένων, χρειάζεται ένα γενικό τρόπο περιγραφής της σύνταξης της γραμμής εκτέλεσης του αλγόριθμου, καθώς και μία περιγραφή του τύπου των δεδομένων εισόδου και εξόδου. Οι πιθανοί τύποι των παραμέτρων ενός αλγόριθμου επεξεργασίας εικόνας είναι: αρχείο εικόνας, διάνυσμα, αριθμός (ακέραιος ή δεκαδικός), σημείο, χαρακτήρας ή οποιοσδήποτε συνδυασμός των παραπάνω. Κατά την εισαγωγή του αλγόριθμου στο σύστημα, κατασκευάζεται ένα αρχείο περιγραφής του τρόπου που εκτελείται ο αλγόριθμος και των τύπων δεδομένων που χρησιμοποιεί στην γραμμή εκτέλεσης του, όπως φαίνεται στο **Σχήμα 6**.

```
[segment-tree]
exepath = /usr/local/bin/segment-tree   $SEED  $PAR  $IMAGE_IN  $IMAGE_OUT
numparams = 4

    [param 1]                          [param 2]
    name = SEED                        name = PAR
    type = POINT                       type = INT
    format = POINT_CDM_LN              format = INT_CDM_LN
    required = YES                     required = NO
    input = YES                        input = YES
    prefix = -s                        prefix = -l

        [param 3]                          [param 4]
        name = IMAGE_IN                    name = IMAGE_OUT
        type = IMAGE                       type = IMAGE
        format = RASTER                    format = RASTER
        required = YES                     required = YES
        input = YES                        input = NO
```

Σχήμα 6: Αρχείο περιγραφής παραμέτρων εκτέλεσης αλγόριθμου.

Εκτός από τα εκτελέσιμα αρχεία των αλγόριθμων, η βάση μπορεί να κρατάει πληροφορία σχετική με τα χαρακτηριστικά του αλγόριθμου σαν λογισμικό, όπως ο τύπος του αλγόριθμου, ο κατασκευαστής του, η προτεινόμενη χρήση του καθώς και πληροφορία για τα χαρακτηριστικά της εκτέλεσης του. Τα χαρακτηριστικά της εκτέλεσης ενός αλγόριθμου είναι δυναμική πληροφορία η οποία συλλέγεται από τον πράκτορα εκτέλεσης στο τέλος κάθε εκτέλεσης και περιλαμβάνει στοιχεία όπως τα μεγέθη των δεδομένων που χρησιμοποιήθηκαν στην είσοδο και παράχθηκαν στην έξοδο, το μέγεθος της απαιτούμενης μνήμης που χρειάζεται για να εκτελεστεί ο αλγόριθμος (συγκριτικά με το μέγεθος της εισόδου του), και τέλος τον χρόνο που χρειάζεται για την εκτέλεση (συγκριτικά με το μέγεθος της εισόδου του). Όλα τα προαναφερθέντα μεγέθη μπορούν μετά να χρησιμοποιηθούν για να δώσουν το στατιστικό προφίλ του κάθε αλγόριθμου.

Τέτοιου είδους πληροφορίες σε συνδυασμό με τους στόχους και τις προτιμήσεις του κάθε χρήστη αποτελούν τη βάση για την ανάπτυξη μηχανισμών για την έξυπνη ανάκτηση αλγόριθμων. Επιπλέον, η δυνατότητα του χρήστη να αλληλεπιδρά με τον αλγόριθμο κατά τη διάρκεια της εκτέλεσης του και η χρήση του στατιστικού προφίλ του κάθε αλγόριθμου, μπορούν να χρησιμοποιηθούν για τον έλεγχο και την αξιολόγηση του. Πρέπει να σημειωθεί ότι παρόλο που η προτεινόμενη αρχιτεκτονική προνοεί για την δυνατότητα αξιολόγησης

αλγόριθμων, καθώς και ανάκλησης αλγόριθμων με βάση τα χαρακτηριστικά τους και τις επιδιώξεις του χρήστη, αυτά δεν αποτελούν αντικείμενο της συγκεκριμένης εργασίας (μια και από μόνα τους είναι ιδιαίτερα απαιτητικά προβλήματα).

Διαχείριση Δεδομένων Εισόδου

Κάθε πράκτορας εκτέλεσης διαχειρίζεται μία τοπική «κρυφή μνήμη»[1], ένα χώρο ορισμένου μεγέθους, στον οποίο μπορεί να αποθηκεύει τα δεδομένα που έχουν χρησιμοποιηθεί ως είσοδο για διάφορους αλγόριθμους που έχουν εκτελεστεί πρόσφατα από τον συγκεκριμένο πράκτορα εκτέλεσης. Η χρήση του χώρου για αποθήκευση δεδομένων εισόδου σε συνδυασμό με την κατάλληλη ταυτοποίηση τους, επιτρέπει την ελαχιστοποίηση μεταφοράς δεδομένων εισόδου από την πηγή τους στο απομακρυσμένο μηχάνημα όπου έχει δρομολογηθεί η εκτέλεση του αλγόριθμου. Αυτό είναι ιδιαίτερα σημαντικό στις περιπτώσεις όπου τα ίδια δεδομένα χρησιμοποιούνται ως είσοδος σε διάφορους αλγόριθμους, ή σε διαδοχικές εκτελέσεις του ίδιου αλγόριθμου με άλλες κάθε φορά τιμές για παραμέτρους εισόδου.

Ο μηχανισμός της «κρυφής μνήμης» που χρησιμοποιείται παρουσιάζει μεγάλες ομοιότητες με τους μηχανισμούς «κρυφής μνήμης» που χρησιμοποιούνται στους διάφορους Proxy Servers και WWW-Browsers που χρησιμοποιούνται στις τεχνολογίες του Internet. Κάθε αρχείο, εικόνα, κείμενο, κτλ. που μπορεί να προσπελασθεί στο Internet έχει μία μοναδική διεύθυνση (URL [79]) με την οποία μπορεί να ταυτοποιηθεί και να αποθηκευτεί στην «κρυφή μνήμη» του Proxy ή του Browser, ώστε αργότερα να μπορεί να ανακτηθεί τοπικά. Με αντίστοιχο τρόπο ταυτοποιούνται τα δεδομένα (εικόνες) και αποθηκεύονται στον προσωρινό χώρο που προαναφέραμε για μελλοντική ανάκτηση σε περίπτωση που ζητηθεί εκτέλεση αλγόριθμου με δεδομένα τα οποία προ-υπάρχουν τοπικά. Στην συγκεκριμένη όμως περίπτωση η ταυτοποίηση των δεδομένων βασίζεται σε μια μοναδική ψηφιακή υπογραφή που υπολογίζεται για κάθε δεδομένο εισόδου. Η ψηφιακή υπογραφή υπολογίζεται με βάση τον ευρύτατα διαδεδομένο αλγόριθμο ψηφιακών υπογραφών MD5 [80], ο οποίος χρησιμοποιεί μία συνάρτηση κατακερματισμού ενός δρόμου και μπορεί να παράγει από μία ακολουθία δεδομένων εισόδου μία ταυτότητα μήκους 128bit. Ο μηχανισμός με τον οποίο γίνεται η ταυτοποίηση των εικόνων στο σύστημα καθώς και μία μικρή παρουσίαση των συναρτήσεων κατακερματισμού ενός δρόμου περιγράφονται στο Παράρτημα Γ.

[1] Ο όρος «κρυφή μνήμη» έχει επικρατήσει για τη μετάφραση του αγγλικού όρου cache. Ο όρος «κρυφή μνήμη» αρχικά χρησιμοποιήθηκε για να περιγράψει τη γρήγορη μνήμη που βρίσκεται κοντά στον επεξεργαστή και αποθηκεύει τις εντολές προς άμεση εκτέλεση. Σήμερα ο όρος χρησιμοποιείται ευρύτερα για να περιγράψει ένα χώρο ορισμένου μεγέθους στον οποίο αποθηκεύονται προσωρινά αντίγραφα διαφόρων δεδομένων και αρχείων, με μοναδικό σκοπό την γρηγορότερη πρόσβαση τους. Με δεδομένους τους συνειρμούς που προκαλεί ο συγκεκριμένος όρος, κρίθηκε σκόπιμο να χρησιμοποιηθεί και στην παρούσα περίπτωση, όπου προφανώς δεν πρόκειται για «κρυφή μνήμη» με την αυστηρή έννοια του όρου.

Διαχείριση Αποτελεσμάτων Εκτέλεσης

Η εκτέλεση ενός αλγόριθμου μπορεί να χρειάζεται από μερικά δευτερόλεπτα μέχρι μερικές ημέρες. Στην πρώτη περίπτωση, ο χρήστης εν γένει δεν έχει προλάβει να αποσυνδεθεί από το σύστημα και μπορεί να παραλάβει τα αποτελέσματα της εκτέλεσης με τον τερματισμό της. Σε αντίθεση, στη δεύτερη περίπτωση το πιο πιθανό για το χρήστη είναι να έχει αποσυνδεθεί από το σύστημα, οπότε τη χρονική στιγμή που τελειώνει η εκτέλεση του αλγόριθμου, τα δεδομένα δεν μπορούν να προωθηθούν άμεσα στον χρήστη. Για την καλύτερη ικανοποίηση των χρηστών καθώς και για την καλύτερη εκμετάλλευση των υπολογιστικών πόρων, κάθε πράκτορας εκτέλεσης διαχειρίζεται ένα προσωρινό αποθηκευτικό χώρο, στον οποίο αποθηκεύει τα παραγόμενα δεδομένα ενός αλγόριθμου που δεν μπορούν να παραληφθούν άμεσα από τον τελικό χρήστη. Στην επόμενη σύνδεση με το σύστημα, ο χρήστης έχει τη δυνατότητα να επιλέξει εάν θέλει την παραλαβή των δεδομένων ή τη διαγραφή τους. Υπάρχει επίσης η δυνατότητα του προνοητικού πράκτορα, που ειδοποιεί (π.χ. με ηλεκτρονικό μήνυμα) τον χρήστη για την ολοκλήρωση εκτέλεσης ενός αλγόριθμου. Τέλος, τα περιεχόμενα του προσωρινού χώρου αποθήκευσης μπορούν να χρησιμοποιηθούν με τρόπο παρόμοιο με τα περιεχόμενα της κρυφής μνήμης, στις περιπτώσεις που είσοδος σε έναν αλγόριθμο εκτέλεσης είναι η έξοδος από προηγούμενη εκτέλεση άλλου αλγόριθμου. Η συγκεκριμένη ευελιξία του πράκτορα εκτέλεσης στην διαχείριση των αποτελεσμάτων εκτέλεσης αλγόριθμων είναι ιδιαίτερα χρήσιμη στις περιπτώσεις που ο χρήστης συνδέεται στο σύστημα από εξωτερικό οργανισμό μέσω Internet, ή από το σπίτι του, ή από κάποιο κινητό υπολογιστή μέσω ασύρματων επικοινωνιών.

3.4 Διαχειριστής

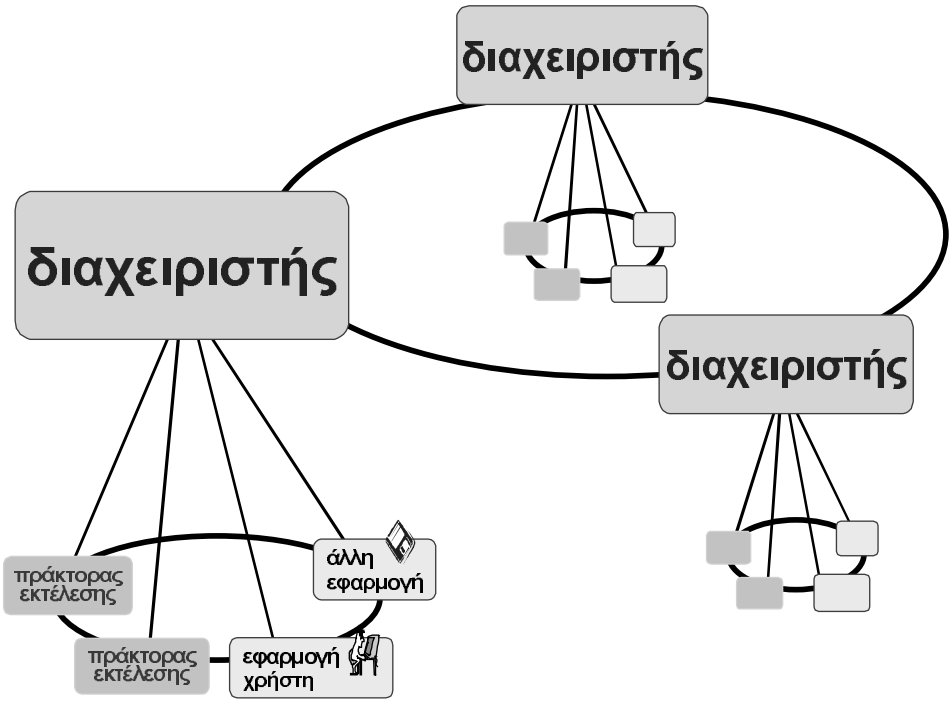
Ο διαχειριστής έχει τους παρακάτω επιμέρους ρόλους:

- αναγνώριση χρηστών και καταγραφή χρεώσεων,
- οργάνωση επικοινωνιών και ανάθεση πόρων,
- διαχείριση κατανεμημένης βάσης αλγόριθμων,
- επίτευξη συνολικών στρατηγικών και πολιτικών για διαχείριση πόρων,
- συνεργασία με άλλους διαχειριστές για την επέκταση του συστήματος.

Ο κύριος ρόλος του διαχειριστή είναι να οργανώνει τις επικοινωνίες των διαφόρων πρακτόρων εκτέλεσης (ΠΕ) και των εφαρμογών των χρηστών. Στις αρμοδιότητές του ανήκει και η ανεύρεση ενός κατάλληλου μηχανήματος για την εκτέλεση του αλγόριθμου που επιθυμεί ο χρήστης. Αυτό είναι το αποτέλεσμα μιας κατανεμημένης διαδικασίας λήψης αποφάσεων στην οποία συμμετέχουν και οι πράκτορες εκτέλεσης. Η διαδικασία λήψης αποφάσεων περιγράφεται αναλυτικότερα στο επόμενο κεφάλαιο. Στις περιπτώσεις

κοστολόγησης των παρεχομένων υπηρεσιών επεξεργασίας, ο διαχειριστής αναλαμβάνει την καταγραφή όλων των χρεώσεων για την έκδοση ενός τελικού λογαριασμού για κάθε χρήστη.

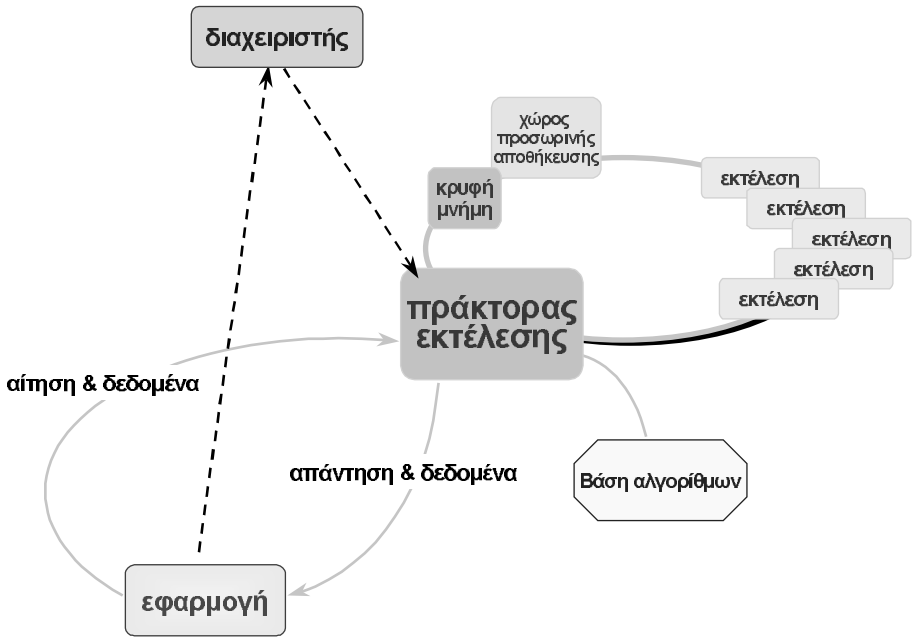
Επιπλέον, ο διαχειριστής είναι υπεύθυνος για θέματα ασφάλειας και ελέγχου πρόσβασης στις υπηρεσίες που παρέχει το σύστημα, μέσω της χρήσης κωδικών πρόσβασης. Εκτός από τους μηχανισμούς για την επίτευξη ενός απλού επιπέδου πρόσβασης, υπάρχει η δυνατότητα για την επίτευξη διαφορετικών επιπέδων και ομάδων πρόσβασης στις παρεχόμενες υπηρεσίες. Σαν αποτέλεσμα, ένας αλγόριθμος μπορεί να είναι κοινός, ελεύθερος για χρήση από όλους τους χρήστες, ή προσωπικός, για χρήση από μεμονωμένους χρήστες ή ομάδες χρηστών. Επίσης, μέσα από τα επίπεδα πρόσβασης για τους διάφορους χρήστες ο διαχειριστής μπορεί να υλοποιήσει διάφορες γενικές πολιτικές τόσο για την πρόσβαση στις υπηρεσίες όσο και για την διαχείριση πόρων και την χρέωση της χρήσης των παρεχόμενων υπηρεσιών.



Σχήμα 7: Επέκταση περιβάλλοντος μέσω δικτύου διαχειριστών.

Τέλος, ο διαχειριστής είναι υπεύθυνος για θέματα ολοκλήρωσης. Κατ' αρχήν όσο αφορά στην τοπική κοινωνία πρακτόρων, μια και ενοποιεί σε ενιαία εικονική βάση τα διάφορα κομμάτια της βάσης των αλγόριθμων που βρίσκονται κατανεμημένα στους πράκτορες εκτέλεσης (κρατώντας αντίγραφα των περιγραφών αλγόριθμων). Επίσης, ο διαχειριστής είναι το στοιχείο κλειδί για την επεκτασιμότητα του προτεινόμενου περιβάλλοντος και των υπηρεσιών που παρέχει. Μία τοπική κοινότητα από πράκτορες εκτέλεσης μπορεί να επεκταθεί περαιτέρω μέσα από ένα δίκτυο διαχειριστών, μέσα στον ίδιο ή σε διαφορετικούς οργανισμούς όπως φαίνεται και στο Σχήμα 7.

Και εδώ πρέπει να τονίσουμε ότι ακολουθώντας το γενικό μοντέλο για την ανάλυση και περιγραφή πρακτόρων (βλέπε δεύτερο κεφάλαιο, σχήμα 2), ο διαχειριστής μπορεί να αναλυθεί σε τρία λειτουργικά μέρη. Καταρχήν, οι αισθητήρες με τους οποίους αντιλαμβάνεται το περιβάλλον του στη συγκεκριμένη περίπτωση είναι μηχανισμοί που του επιτρέπουν να ακούει μηνύματα από άλλους πράκτορες, διαχειριστές ή πράκτορες εκτέλεσης, καθώς και μηχανισμοί με τους οποίους συμβουλεύεται τη βάση αλγόριθμων. Το γνωστικό μέρος αφορά στη δυνατότητα να παίρνει αποφάσεις βασισμένος στα διάφορα σήματα τα οποία λαμβάνει από το περιβάλλον του σε συνδυασμό με τις διάφορες αντιλήψεις που έχει για τον τρόπο με τον οποίο συνεργάζεται με τους υπόλοιπους πράκτορες, τον τρόπο εκτέλεσης της δημοπρασίας, γνώσεις που έχει αποκτήσει για την αξιοπιστία του κάθε πράκτορα, καθώς και πολιτικές τις οποίες πρέπει να εφαρμόσει. Τέλος, το αποτέλεσμα του δεύτερου μέρους καθοδηγεί τα όργανα δράσης, τα οποία στη συγκεκριμένη περίπτωση υλοποιούνται στέλνοντας μήνυματα στους ανάλογους κάθε φορά πράκτορες.



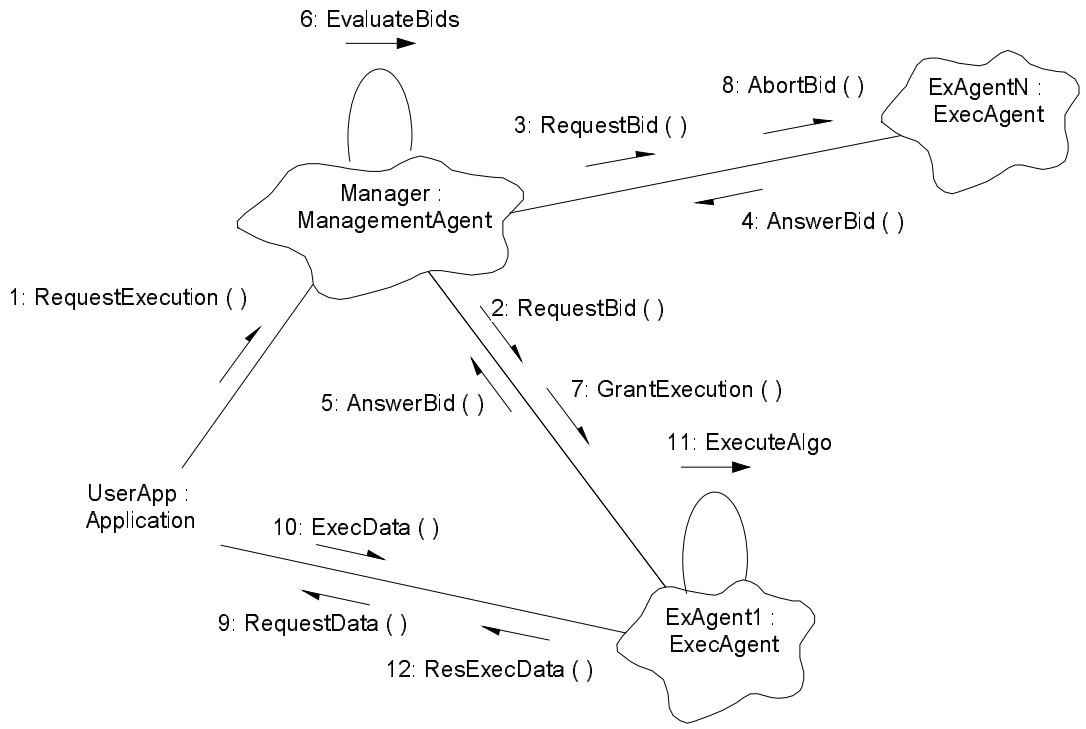
Σχήμα 8: Ακολουθία ανάθεσης της εκτέλεσης αλγόριθμου σε πράκτορα εκτέλεσης.

3.5 Διαδικασία Εκτέλεσης Υπηρεσίας

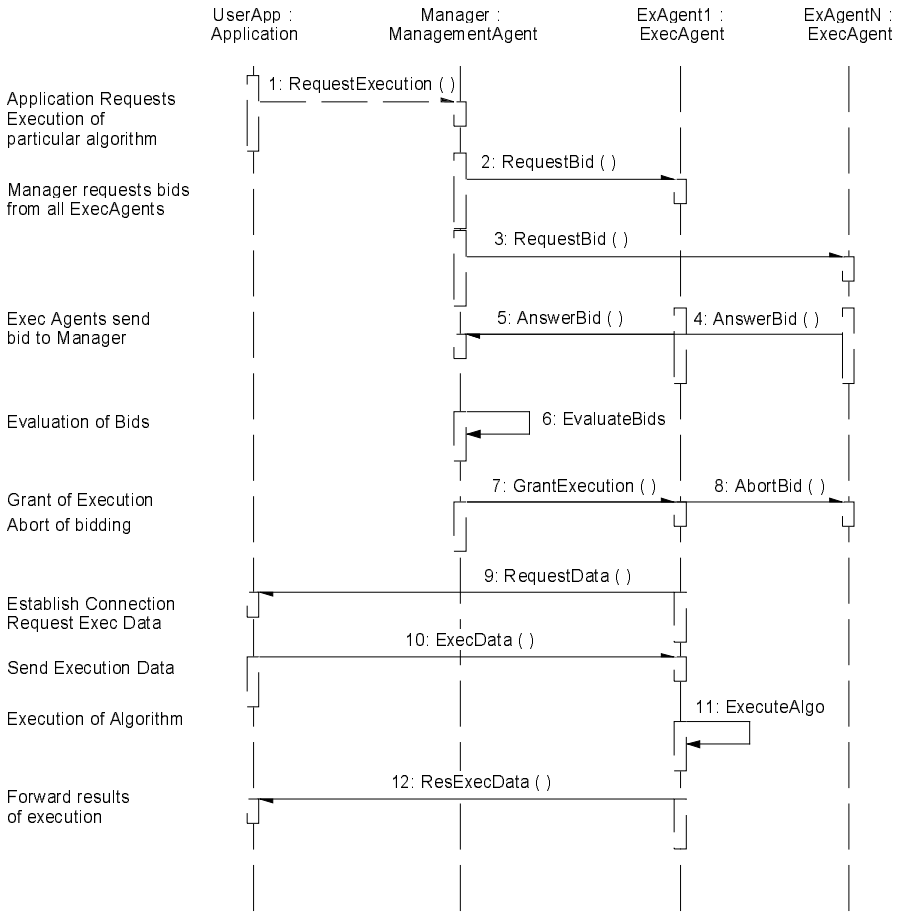
Ο χρήστης (ή κάποια εφαρμογή) εκφράζει την επιθυμία του για την εκτέλεση αλγόριθμου επεξεργασίας δεδομένων πάνω σε ορισμένα δεδομένα, στέλνοντας μία κατάλληλη αίτηση για εκτέλεση στο διαχειριστή. Ο διαχειριστής ξεκινά την διαδικασία ανάθεσης της εκτέλεσης και επικοινωνεί με τους πράκτορες εκτέλεσης (βλ. τέταρτο κεφάλαιο). Αφού παρθεί η απόφαση για την ανάθεση της εκτέλεσης, προωθεί την αίτηση στον αντίστοιχο πράκτορα εκτέλεσης [Σχήμα 8]. Ο επιλεγμένος ΠΕ μπορεί να ανήκει είτε στην κοινότητα των πρακτόρων που διαχειρίζεται ο διαχειριστής, είτε σε κοινότητα κάποιου άλλου διαχειριστή.

Μόλις κάποιος ΠΕ λάβει εντολή για την εκτέλεση ενός αλγόριθμου που βρίσκεται στην τοπική βάση εκτελέσιμων, δημιουργεί ένα κανάλι επικοινωνίας με την εφαρμογή που ζήτησε την εκτέλεση της υπηρεσίας, με σκοπό να παραλάβει επιπλέον πληροφορίες καθώς και τα πιθανά δεδομένα εισόδου που χρειάζονται για την εκτέλεση της (αν αυτά δεν υπάρχουν ήδη στην τοπική «κρυφή μνήμη»). Στη συνέχεια ο εκτελεστής αλγόριθμων του συγκεκριμένου πράκτορα προχωράει αυτόνομα στην εκτέλεση του αλγόριθμου. Οι αλγόριθμοι εκτελούνται σαν διαφορετικές διεργασίες και ο ΠΕ ελέγχει συνεχώς όλες τις διαφορετικές εκτελέσεις που διεξάγονται ανά πάσα χρονική στιγμή. Μέρος των μηνυμάτων που ανταλλάσσονται και των ενεργειών που λαβαίνουν χώρα κατά την διαδικασία[1] που περιγράφεται εδώ φαίνονται στο Σχήμα 9. Η χρονική σειρά με την οποία εκτελούνται οι παραπάνω ενέργειες δίνεται στο Σχήμα 10. Μόλις τελειώσει η εκτέλεση ενός αλγόριθμου, ο πράκτορας εκτέλεσης αναλαμβάνει την προώθηση των παραγόμενων δεδομένων εξόδου στην αντίστοιχη εφαρμογή που ζήτησε την εκτέλεση. Σε περίπτωση που υπάρχει πρόβλημα επικοινωνίας μεταξύ του ΠΕ και της αντίστοιχης εφαρμογής, είτε λόγω καθυστερήσεων δικτύου ή επειδή η αντίστοιχη εφαρμογή δεν είναι πλέον ενεργή, τότε ο ΕΠ αποθηκεύει τα αποτελέσματα στον προσωρινό χώρο αποθήκευσης, όπως προαναφέρθηκε, με σκοπό να τα παραδώσει αργότερα.

[1] Να σημειωθεί ότι για ένα μέρος της σχεδίασης της αρχιτεκτονικής χρησιμοποιήθηκε το Rational Rose, ένα εργαλείο για την οντοκεντρική σχεδίαση και ανάπτυξη εφαρμογών [81]. Ωστόσο, το εργαλείο δεν χρησιμοποιήθηκε για το σύνολο του σχεδιασμού, γιατί είχαμε στην κατοχή μας μόνο την έκδοση "επίδειξης" η οποία υποστηρίζει πολύ περιορισμένο αριθμό κλάσεων. Τα σχήματα 9 και 10 έγιναν με τη βοήθεια αυτού του εργαλείου.



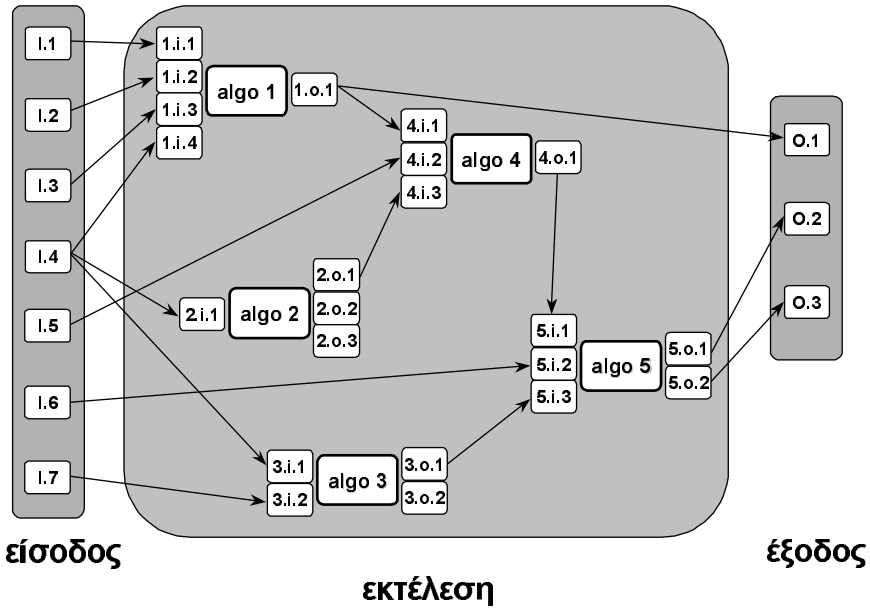
Σχήμα 9 : Ανταλλασσόμενα μηνύματα κατά την ανάθεση εκτέλεσης ανάμεσα στον πράκτορα εκτέλεσης (ExecAgent), τον διαχειριστή (ManagementAgent) και την αιτούσα εφαρμογή (Application).



Σχήμα 10: Χρονική αλληλουχία ανταλλασσόμενων μηνυμάτων κατά την ανάθεση εκτέλεσης

3.6 Ακολουθίες Αλγόριθμων

Η επεξεργασία δεδομένων πολλές φορές περιλαμβάνει την επαναληπτική εκτέλεση μερικών ακολουθιών από μεμονωμένες εκτελέσεις αλγόριθμων. Παράδειγμα αποτελεί η απομόνωση ορισμένων χαρακτηριστικών από ένα μεγάλο σύνολο εικόνων, όπου η ίδια ακολουθία εκτελέσεων πρέπει να εφαρμοστεί σε κάθε εικόνα. Ένα περιβάλλον που υποστηρίζει υπηρεσίες επεξεργασίας δεδομένων πρέπει να παρέχει μηχανισμούς που να απλουστεύουν την εκτέλεση πολύπλοκων ακολουθιών αλγόριθμων, ομαδοποιώντας τους και δημιουργώντας ακολουθίες από αλγόριθμους.



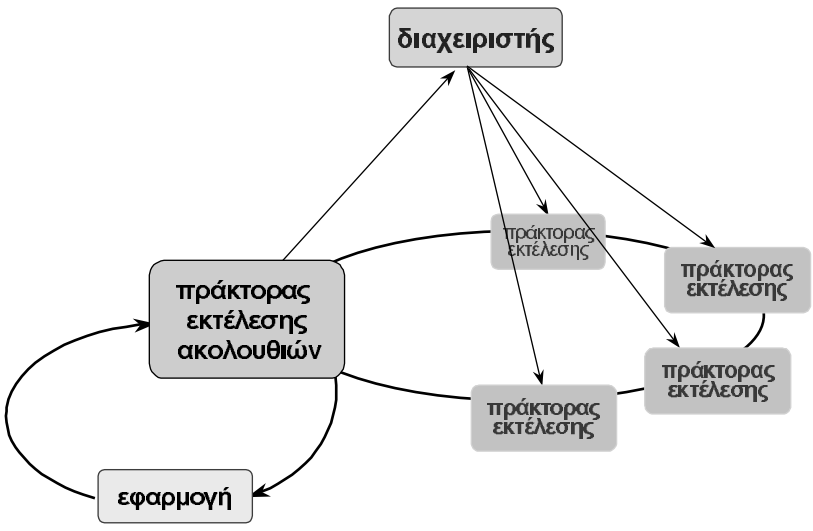
Σχήμα 11: Ένα παράδειγμα ακολουθίας αλγόριθμων όπου φαίνεται μια ποικιλία πολυπλοκότητας σε σχέση με την αλληλουχία των επιμέρους αλγόριθμων και τις αλληλεξαρτήσεις των δεδομένων εισόδου και εξόδου. Οι επιμέρους αλγόριθμοι της ακολουθίας συμβολίζονται με «algo N», όπου N=1, 2, 3,…, ενώ τα δεδομένα εισόδου και εξόδου ενός αλγόριθμου algoN συμβολίζονται με «i.N.k» και «o.N.l», αντίστοιχα (όπου k, l = 1, 2, 3, …). Τέλος, τα δεδομένα εισόδου και εξόδου της ακολουθίας συμβολίζονται με «I.m» και «O.n», αντίστοιχα (όπου m, n = 1, 2, 3, …).

Στην προτεινόμενη αρχιτεκτονική, η ακολουθία εκτελέσεων θεωρείται ως ένα σύνολο από μεμονωμένους αλγόριθμους οι οποίοι μπορούν να εκτελεστούν είτε ανεξάρτητα είτε σειριακά χρησιμοποιώντας τα ίδια ή διαφορετικά σύνολα δεδομένων. Δεν υπάρχει περιορισμός στην πολυπλοκότητα των συνδυασμών των αλγόριθμων και τις εσωτερικές σχέσεις των δεδομένων εισόδων και δεδομένων εξόδων τους. Επίσης, οι διάφοροι αλγόριθμοι μέσα σε μία ακολουθία εκτέλεσης, μπορούν να βρίσκονται σε διαφορετικά μέρη (πλατφόρμες) του κατανεμημένου συστήματος. Γενικά, μία ακολουθία εκτέλεσης,

περιγράφεται με τον ίδιο τρόπο όπως ο μεμονωμένος αλγόριθμος, δηλαδή υπάρχει ένα σύνολο δεδομένων εισόδου, ένα κομμάτι εκτέλεσης και ένα σύνολο δεδομένων εξόδου, όπως φαίνεται στο Σχήμα 11.

Για την καλύτερη διαχείριση της εκτέλεσης των ακολουθιών, εισάγουμε μία καινούργια οντότητα στο σύστημα, τον πράκτορα εκτέλεσης ακολουθιών (ΠΕΑ), όπως φαίνεται στο Σχήμα 12. Αυτή η οντότητα ενεργεί σαν μεσολαβητής των διάφορων ακολουθιών εκτελέσεων. Μοντελοποιεί την ακολουθία αλγόριθμων προς εκτέλεση ως ένα προσανατολισμένο ακυκλικό γράφο, όπου κάθε κόμβος είναι η εκτέλεση ενός αλγόριθμου και κάθε ακμή είναι τα δεδομένα εισόδου και εξόδου, ανάλογα με το αν η ακμή δείχνει σε κάποιον κόμβο ή φεύγει από κάποιο κόμβο. Η εκτέλεση ενός αλγόριθμου (κόμβου) ενεργοποιείται μόλις είναι διαθέσιμα δεδομένα σε κάθε ακμή που καταφθάνει στον συγκεκριμένο κόμβο. Μόλις τελειώσει η εκτέλεση ενός κόμβου, ενημερώνει τους κόμβους που περιμένουν δεδομένα από τον κόμβο που μόλις τελείωσε, και αναλόγως ενεργοποιείται μία νέα εκτέλεση. Παράλληλα ενημερώνεται η αιτούσα εφαρμογή για την πορεία της εκτέλεσης της ακολουθίας των αλγόριθμων, και αν το επιθυμεί επιστρέφονται και τα ενδιάμεσα αποτελέσματα.

Μία ακολουθία εκτέλεσης αλγόριθμων, μπορεί να αποτελείται από τελείως ανεξάρτητα κομμάτια, τα οποία μέσω των μηχανισμών για τη διαχείριση πόρων που παρέχει το σύστημα και αν κάτι τέτοιο είναι εφικτό, μπορούν να εκτελεστούν σε διαφορετικούς κόμβους του κατανεμημένου συστήματος, με άμεσο αποτέλεσμα την μείωση του απαιτούμενου χρόνου εκτέλεσης της ακολουθίας αλγόριθμων.



Σχήμα 12: Πράκτορας εκτέλεσης ακολουθιών (ΠΕΑ).

ΚΕΦΑΛΑΙΟ 4 Διαχείριση Πόρων

Εδώ πρέπει να σημειωθεί ότι τα σημερινά κατανεμημένα συστήματα που σχετίζονται με υπηρεσίες επεξεργασίας δεδομένων διακρίνονται για την πολυπλοκότητά τους, η οποία οφείλεται στους παρακάτω παράγοντες:

- μέγεθος συστήματος (αριθμός χρηστών και μηχανημάτων),
- ετερογενείς εφαρμογές (πολυμέσων, άντλησης πληροφορίας, ψηφιακές βιβλιοθήκες, κλπ.),
- ετερογενείς πόροι (υλικό, λειτουργικά συστήματα, μνήμη, εύρος μεταφοράς δεδομένων),
- συμμετοχή πολλών διαφορετικών οργανισμών στο κατανεμημένο σύστημα,
- ανάγκη για ταυτόχρονη ανάθεση πολλαπλών πόρων στις αντίστοιχες διεργασίες,
- δυναμικότητα του περιβάλλοντος ως προς το μέγεθος, τη σύνθεση, το βαθμό απασχόλησης, κλπ..

Λαμβάνοντας υπόψη την αυξανόμενη πολυπλοκότητα και την δυναμικότητα των συστημάτων που καλείται να υποστηρίξει το προτεινόμενο περιβάλλον, είναι σαφές ότι μια πρακτική και υλοποιήσιμη στρατηγική για την διαχείριση των πόρων πρέπει να είναι κατ' αρχήν δυναμική και κατανεμημένη. Όπως ήδη παρουσιάστηκε στην ανασκόπηση, ένα ευρέως διαδεδομένο μοντέλο για εφαρμογές με παρόμοιες απαιτήσεις είναι το Contract Net και οι διάφορες παραλλαγές σ' αυτό [74,77,62]. Το βασικό όμως πρόβλημα παρόμοιων προσεγγίσεων είναι ότι δεν αντιμετωπίζουν αποτελεσματικά την περίπτωση δυναμικά μεταβαλλόμενης τοπολογίας και σύνθεσης του περιβάλλοντος των πρακτόρων. Δηλαδή, για να επιστρέψουμε στο πρόβλημα της παρούσας εργασίας, άμεση εφαρμογή του παραπάνω μοντέλου για την διαχείριση πόρων θα απαιτούσε η εφαρμογή του κάθε χρήστη να γνωρίζει την τοπολογία των πρακτόρων εκτέλεσης κάθε δεδομένη στιγμή. Αυτό αυξάνει τον φόρτο επικοινωνιών και έχει ιδιαίτερα μεγάλο κόστος στην (συχνή) περίπτωση που ο χρήστης αποσυνδέεται προσωρινά από το σύστημα και επανασυνδέεται για να πάρει τα αποτελέσματα της επεξεργασίας. Επίσης, το παραπάνω πρόβλημα γίνεται ακόμα πιο έντονο στην περίπτωση επέκτασης του συστήματος για να καλύψει περισσότερους του ενός οργανισμούς, όπως επίσης και στην περίπτωση της εξυπηρέτησης εξωτερικού χρήστη. Τέλος, με τις ήδη υπάρχουσες προσεγγίσεις δεν καλύπτεται (με πρακτικό τρόπο) η απαίτηση για διαμόρφωση επιμέρους πολιτικών πρόσβασης, ανάθεσης πόρων και χρέωσης με βάση την εξουσιοδότηση των χρηστών (μεμονωμένα ή σε ομάδες), καθώς και της ανάγκης για τήρηση λογαριασμών στην περίπτωση της χρέωσης του κάθε χρήστη.

Ο μηχανισμός για την διαχείριση πόρων που προτείνει η παρούσα εργασία ακολουθεί κατ' αρχήν τις βασικές αρχές του μοντέλου Contract Net, όσο αφορά στη συνεργασία κατανεμημένων πρακτόρων και την ανάθεση εργασιών με βάση τον μηχανισμό της δημοπρασίας. Ωστόσο, για να καλυφθούν τα παραπάνω προβλήματα και αδυναμίες,

εισάγεται η χρήση του κεντρικού διαχειριστή των δημοπρασιών. Έτσι, η πληροφορία για την πλήρη τοπολογία του συστήματος κάθε δεδομένη στιγμή είναι απαραίτητη μόνο στον κεντρικό πράκτορα, ελαττώνοντας τον φόρτο επικοινωνιών και υποστηρίζοντας δυναμικά μεταβαλλόμενα συστήματα. Επιπλέον, η χρήση του κεντρικού διαχειριστή επιτρέπει την ιεραρχική επέκταση του μηχανισμού, υποστηρίζει διαμόρφωση πολιτικών για επιμέρους χρήστες (ή ομάδες χρηστών), και δίνει την δυνατότητα για εξαγωγή ενιαίας κοστολόγησης υπηρεσιών.

Η βάση του μηχανισμού διαχείρισης πόρων είναι η διενέργεια δημοπρασίας από τον διαχειριστή, στην οποία συμμετέχουν οι κατανεμημένοι πράκτορες εκτέλεσης. Στην σχετική βιβλιογραφία οικονομικών μοντέλων [78] αναφέρονται διάφορα είδη δημοπρασίας, τα κυριότερα από τα οποία παρουσιάζονται παρακάτω. Να σημειωθεί ότι οι παρακάτω μηχανισμοί χρησιμοποιούνται στα ευρύτερα πλαίσια οικονομικών συνδιαλλαγών, αλλά για ευκολία παρουσίασης, περιγράφονται στα πλαίσια του προβλήματος της συγκεκριμένης εργασίας.

Στην δημοπρασία τύπου σφραγισμένης προσφοράς (sealed bid), για κάθε διεργασία προς εκτέλεση, ο διαχειριστής ανακοινώνει την ανάλογη αίτηση. Κάθε πράκτορας υπολογίζει το κόστος για να αναλάβει την διεργασία (αναμενόμενος χρόνος και κόστος επεξεργασίας για την περίπτωσή μας) και στέλνει την προσφορά του στον διαχειριστή. Κανένας πράκτορας δεν γνωρίζει το περιεχόμενο των προσφορών των άλλων πρακτόρων. Ο διαχειριστής αξιολογεί τις προσφορές που έλαβε και παραδίνει την διεργασία στον πράκτορα που έδωσε την καλύτερη προσφορά (με κριτήρια που εξαρτώνται από την δεδομένη εφαρμογή).

Ένα δεύτερο μοντέλο είναι η δημοπρασία ολλανδικού τύπου (Dutch). Στην δημοπρασία αυτή, ο διαχειριστής θέτει μία αναμενόμενη 'τιμή' (χρόνου και κόστους) για την διεργασία και διερευνά να δει αν υπάρχει κάποιος πράκτορας που να μπορεί να ανταπεξέλθει. Αν όχι, σταδιακά αυξάνει την τιμή μέχρι που κάποιος πράκτορας να στείλει μία προσφορά.

Ένα τρίτο μοντέλο είναι η δημοπρασία αγγλικού τύπου (English). Η δημοπρασία αυτή ξεκινά με μια αρχική προσφορά από ένα πράκτορα. Ο διαχειριστής χρησιμοποιεί αυτή την προσφορά για να θέσει την αρχική 'τιμή' για την διεργασία, και ζητά προσφορές από τους υπόλοιπους πράκτορες. Κάθε προσφορά που φτάνει στον διαχειριστή ανακοινώνεται και η διαδικασία συνεχίζεται ώσπου κανένας πράκτορας να μη προτίθεται να κάνει καλύτερη προσφορά από την τρέχουσα.

Στην παρούσα εργασία χρησιμοποιήθηκε μία μορφή της δημοπρασίας σφραγισμένης προσφοράς, γιατί προσφέρει το πλεονέκτημα της χαμηλής επιβάρυνσης του συστήματος σε επικοινωνίες μια και χρειάζεται μόνο ένας γύρος για την ανάθεση της διεργασίας. Οι δημοπρασίες αγγλικού και ολλανδικού τύπου ενδείκνυνται για την πώληση αγαθών ή υπηρεσιών, όπου ο διοργανωτής της δημοπρασίας προσπαθεί να πετύχει την καλύτερη δυνατή τιμή, μέσα από μία αλληλουχία διαδοχικών προσφορών που δέχεται από κάθε ενδεχόμενο αγοραστή (πράκτορας). Παρόλο που οι δημοπρασίες αγγλικού και ολλανδικού τύπου μπορούν να δώσουν καλύτερα αποτελέσματα όσο αναφορά στο τελικό κόστος χρήσης, δεν ενδείκνυται η χρήση τους στην περίπτωση μας γιατί χρειάζονται πολλαπλοί γύροι, οι οποίοι επιβαρύνουν την επικοινωνία και προσθέτουν επιπλέον καθυστερήσεις

στην ανάθεση της κάθε εκτέλεσης (π.χ., μια σύντομη εκτέλεση, θα δέχονταν χρονικές επιβαρύνσεις πολλαπλάσιες του πραγματικού χρόνου εκτέλεσής της). Στις επόμενες παραγράφους παρουσιάζεται αναλυτικά ο μηχανισμός της δημοπρασίας, ο τρόπος υπολογισμού της κάθε προσφοράς και η αξιολόγηση των προσφορών.

4.1 Διαδικασία Ανάθεσης Πόρων

Όπως ήδη αναφέρθηκε, όταν ο χρήστης αποφασίζει την εκτέλεση ενός αλγόριθμου, ο διαχειριστής λαμβάνει μία αίτηση για την συγκεκριμένη εκτέλεση. Η αίτηση περιέχει πληροφορίες για τα δεδομένα που θα χρησιμοποιήσει ο αλγόριθμος στην είσοδό του. Σε κάθε αίτηση αντιστοιχεί ένας μοναδικός αριθμός (ταυτότητα διεργασίας) που την χαρακτηρίζει μοναδικά σε ολόκληρο το περιβάλλον. Καταγράφεται η ακριβής ώρα που ο διαχειριστής παρέλαβε την αίτηση, και η αίτηση προστίθεται σε μία λίστα η οποία περιέχει όλες τις εκκρεμείς αιτήσεις που υπάρχουν σε κάθε χρονική στιγμή στο περιβάλλον. Στη συνέχεια ο διαχειριστής προωθεί την αίτηση σε όλους τους συνδεμένους πράκτορες εκτέλεσης για αξιολόγηση και υπολογισμό της προσωπικής τους προσφοράς. Η αίτηση αυτή περιέχει την ταυτότητα της διεργασίας, το όνομα της υπηρεσίας καθώς και διάφορες πληροφορίες σχετικές με τα δεδομένα εισόδου, οι οποίες θα χρησιμοποιηθούν αργότερα για να ελεγχθεί εάν τα δεδομένα βρίσκονται ήδη στην τοπική κρυφή μνήμη.

Κάθε πράκτορας εκτέλεσης, μόλις λάβει μία αίτηση για προσφορά σχετική με την εκτέλεση ενός αλγόριθμου, την αξιολογεί βασισμένος σε δεδομένα σχετικά με τον αλγόριθμο που έχει αποκτήσει στο παρελθόν καθώς και σε πληροφορίες που μαζεύει από το περιβάλλον του, δηλαδή υπολογίζει τον αναμενόμενο χρόνο εκτέλεσης καθώς και το αναμενόμενο κόστος εκτέλεσης. Οι υπολογισμοί βασίζονται στα παρακάτω:

- πληροφορίες σχετικές με την κατάσταση του δικτύου,
- απόδοση του τοπικού επεξεργαστή,
- τρέχουσα κατάσταση του τοπικού επεξεργαστή,
- πιθανή ύπαρξη των δεδομένων εισόδου στην τοπική κρυφή μνήμη,
- ιστορία εκτέλεσης της ζητούμενης υπηρεσίας (στατιστικό προφίλ),
- αυτοπεποίθηση,
- συγκεκριμένες πολιτικές και στρατηγικές.

Στη συνέχεια ο πράκτορας εκτέλεσης στέλνει την προσφορά του στο διαχειριστή. Η προσφορά περιέχει την ταυτότητα της διεργασίας, τον αναμενόμενο χρόνο εκτέλεσης

καθώς και το αναμενόμενο κόστος της εκτέλεσης. Αν η εκτέλεση της υπηρεσίας χρειάζεται δεδομένα που είναι φυλαγμένα στην τοπική κρυφή μνήμη και ο συγκεκριμένος πράκτορας εκτέλεσης αποφασίζει να κάνει προσφορά, τα δεδομένα αυτά δεσμεύονται ώστε να αποφευχθεί πιθανή διαγραφή τους. Αργότερα, αν η εκτέλεση της υπηρεσίας δεν ανατεθεί στο συγκεκριμένο πράκτορα, τα δεδομένα απελευθερώνονται.

Σε περίπτωση που ο πράκτορας εκτέλεσης δεν μπορεί να χειριστεί την αίτηση είτε λόγω υψηλού φόρτου εργασίας είτε λόγω πολιτικών αποφάσεων, τότε προσθέτει την αίτηση σε μια λίστα από αιτήσεις προς αξιολόγηση. Μόλις τελειώσει την εκτέλεση κάποιου αλγόριθμου ή αν παραλάβει μήνυμα για 'εγκατάλειψη εκτέλεσης' για κάποια εκτέλεση που εκκρεμεί, επαναλαμβάνει την αξιολόγηση των εκκρεμών αιτήσεων και υποβάλει κάποια προσφορά στο διαχειριστή.

Είναι πιθανό το υπολογισμένο κόστος εκτέλεσης να είναι λάθος, λόγω ελλιπούς ή λανθασμένης πληροφορίας (π.χ. προφίλ εκτέλεσης αλγόριθμου που δεν είναι στατιστικά σωστό είτε γιατί δεν υπάρχει ακόμα στατιστικά έγκυρη πληροφορία, είτε γιατί δεν μπορεί να υπολογιστεί στατιστικό προφίλ λόγω εγγενούς τυχαιότητας). Αν το τελικό κόστος είναι μεγαλύτερο από το αρχικά υπολογισμένο, τότε η τελική χρέωση είναι το υπολογισμένο κόστος, ενώ αν είναι μικρότερο τότε χρεώνεται το πραγματικό κόστος. Αυτή η πρακτική στοχεύει στο να αποτρέψει τους πράκτορες να αναπτύξουν μια πολιτική παραπλάνησης, δηλαδή να δίνουν χαμηλές προσφορές για να πάρουν τη δουλειά και αργότερα να χρεώνουν περισσότερα.

Ο διαχειριστής περιμένει ένα συγκεκριμένο προκαθορισμένο χρονικό διάστημα μέσα στο οποίο μαζεύει τις προσφορές που του στέλνουν οι διάφοροι πράκτορες εκτέλεσης. Μόλις περάσει αυτό το χρονικό διάστημα, αξιολογεί όλες τις προσφορές που παρέλαβε και αναθέτει την εκτέλεση της υπηρεσίας στον πράκτορα που έκανε την καλύτερη προσφορά. Στη συνέχεια στέλνει ένα μήνυμα 'ανάθεσης εκτέλεσης' στον πράκτορα που κέρδισε τη δημοπρασία, και ένα μήνυμα 'ακύρωσης εκτέλεσης' στους ηττημένους. Τα κριτήρια επιλογής της καλύτερης προσφοράς μπορούν να οριστούν από τον χρήστη που ζητάει την εκτέλεση της υπηρεσίας ή διαφορετικά μπορούν να οριστούν από την πολιτική και τις στρατηγικές που εφαρμόζει ο διαχειριστής. Σε κάθε περίπτωση τα κριτήρια μπορεί να είναι:

- το μικρότερο αναμενόμενο κόστος,
- ο μικρότερος αναμενόμενος χρόνος εκτέλεσης,
- το μικρότερο αναμενόμενο κόστος και μικρότερο από x μονάδες κόστους,
- ο μικρότερος αναμενόμενος χρόνος εκτέλεσης και μικρότερος από x μονάδες χρόνου,
- ελαχιστοποίηση κάποιας συνάρτησης αναμενόμενου κόστους και χρόνου.

Σε περίπτωση που καμία προσφορά δεν παραλήφθηκε μέσα στον αναμενόμενο χρόνο η εκτέλεση της υπηρεσίας ανατίθεται στον πρώτο πράκτορα που θα δώσει προσφορά για τη

συγκεκριμένη υπηρεσία. Αν πάλι δεν παραληφθεί καμία προσφορά για ακόμη ένα χρονικό διάστημα, τότε ο διαχειριστής στέλνει ξανά αιτήσεις για εκτέλεση. Σε περίπτωση που η αίτηση για εκτέλεση δεν ικανοποιηθεί για ένα αριθμό συνεχόμενων επαναλήψεων, η εκτέλεση ακυρώνεται και η εφαρμογή (και ο χρήστης) ενημερώνεται κατάλληλα.

Στο τέλος της εκτέλεσης ενός αλγόριθμου επεξεργασίας, ο πράκτορας εκτέλεσης αξιολογεί την προσφορά που έκανε, υπολογίζοντας την απόκλιση της προσφοράς (χρόνος και κόστος) από την πραγματικότητα, και ενημερώνει τον συντελεστή αξιοπιστίας του ανάλογα. Ο συντελεστής αξιοπιστίας του μπορεί να χρησιμοποιηθεί για να διορθωθούν μελλοντικά λάθη στους υπολογισμούς και να προσδιοριστεί κάποιο συστηματικό σφάλμα. Η συγκεκριμένη διαδικασία αξιολόγησης δίνει στον πράκτορα εκτέλεσης την δυνατότητα να καλυτερεύσει την απόδοσή του στο μέλλον, βελτιώνοντας την αυτοπεποίθησή του, καθώς και να εξυπηρετήσει καλύτερα τόσο τον πελάτη (προσφέροντας του καλύτερες υπηρεσίες για το κόστος που πληρώνει) όσο και τον φορέα παροχής υπηρεσιών (βεβαιώνοντας ότι η τιμή της προσφερόμενης υπηρεσίας καλύπτει το κόστος της).

Ο πράκτορας εκτέλεσης ενημερώνει τον διαχειριστή για τον τερματισμό της εκτέλεσης μιας υπηρεσίας, συμπεριλαμβάνοντας το πραγματικό κόστος και πραγματικό χρόνο εκτέλεσης. Ο πράκτορας εκτέλεσης ποτέ δεν λέει ψέματα σχετικά με πληροφορίες που του ζητείται να παρέχει (ιδιότητα της ειλικρινείας). Ο διαχειριστής αξιολογεί το αποτέλεσμα της εκτέλεσης βασισμένος σε πληροφορίες που του παρέχονται από τον πράκτορα εκτέλεσης που ανέλαβε την εκτέλεση μιας υπηρεσίας, δηλαδή τις τιμές χρόνου και κόστους της εκτέλεσης σε σχέση με αυτές της προσφοράς, και τροποποιεί κατάλληλα την πεποίθησή του για την αξιοπιστία του συγκεκριμένου πράκτορα. Σε επόμενες δημοπρασίες, ο διαχειριστής αξιολογεί τις προσφορές λαμβάνοντας υπόψη του και την αξιοπιστία κάθε πράκτορα. Χάνοντας τελείως την αξιοπιστία, ένας πράκτορας μπορεί να έχει σαν αποτέλεσμα τη λήψη δραστικών μέτρων εναντίον του, π.χ. σχετικό μήνυμα στους διαχειριστές του συστήματος για ανάγκη αλλαγής ή αναβάθμισης των πόρων που χρησιμοποιεί, επαναξιολόγηση του περιβάλλοντος και της δυνατότητας αντίληψης του, αλλαγή των πολιτικών χρέωσης, κτλ.

Με δεδομένη την γενική περιγραφή της διαδικασίας ανάθεσης πόρων, οι επόμενες παράγραφοι περιγράφουν με περισσότερες λεπτομέρειες τους μηχανισμούς υπολογισμού της κάθε προσφοράς.

4.2 Μοντέλο Συστήματος – Γενικές Παραδοχές

Υποθέτουμε ότι το κατανεμημένο σύστημα που χρησιμοποιούμε αποτελείται από N υπολογιστικούς κόμβους P_1, …, P_N. Για κάθε κόμβο P_i ορίζουμε ένα δείκτη ταχύτητας (CPU_index_i) ο οποίος αντικατοπτρίζει την υπολογιστική ισχύ του κάθε κόμβου. Ο δείκτης αυτός δείχνει πόσες φορές πιο γρήγορος είναι ο υπολογιστικός κόμβος P_i από κάποιον υπολογιστή που έχει δείκτη ταχύτητας ίσο με τη μονάδα. Σαν αποτέλεσμα μία διεργασία η οποία χρειάζεται μ υπολογιστικά δευτερόλεπτα στον επεξεργαστή με δείκτη

ταχύτητας ίσο με τη μονάδα, θα χρειάζεται μ/CPU_index$_i$ δευτερόλεπτα στον κόμβο P_i. Ο δείκτης αυτός μπορεί να υπολογιστεί χρησιμοποιώντας τις μετρήσεις απόδοσης υπολογιστικών συστημάτων SPEC [82,83]. Επιπρόσθετα υποθέτουμε ότι όλοι οι κόμβοι είναι πλήρως συνδεδεμένοι μεταξύ τους με ένα δίκτυο από σημείο σε σημείο (point-to-point) το οποίο παρουσιάζει καθυστέρηση μετάδοσης *d* sec/byte.

4.3 Υπολογισμός Χρόνου

Ο αναμενόμενος συνολικός χρόνος εκτέλεσης για την προσφορά (*total_est_time*) είναι το άθροισμα του αναμενόμενου χρόνου που χρειάζεται για την εκτέλεση του αλγόριθμου (*est_execution_time_final*) και του αναμενόμενου χρόνου που χρειάζεται για τη μεταφορά των δεδομένων εισόδου και εξόδου από και προς τον πράκτορα εκτέλεσης (*est_trasfer_time*):

$$total_est_time = est_transfer_time + est_execution_time_final \quad [1]$$

Ο χρόνος μεταφοράς εξαρτάται από τα χαρακτηριστικά του δικτύου και τα περιεχόμενα της τοπικής κρυφής μνήμης, ενώ ο χρόνος εκτέλεσης εξαρτάται από την υπηρεσία και τα χαρακτηριστικά της μηχανής που θα αναλάβει την εκτέλεσή της καθώς και την κατάσταση στην οποία βρίσκεται.

4.3.1 Υπολογισμός Χρόνου Μετάδοσης στο Δίκτυο

Μόλις ο πράκτορας εκτέλεσης λάβει μία αίτηση για εκτέλεση, υπολογίζει τον αναμενόμενο όγκο δεδομένων εισόδου καθώς και τον αναμενόμενο όγκο των δεδομένων που θα παραχθούν:

$$est_input_data_transferred = \sum_{i=1}^{K} size_input_data_i * data_in_cache(input_data_i) \quad [2]$$

$$est_output_data = average_output_data \quad [3]$$

όπου K είναι ο αριθμός των ορισμάτων εισόδου που έχει ο αλγόριθμος προς εκτέλεση. Η *data_in_cache()* είναι μία λογική συνάρτηση και είναι αληθής (=1) όταν τα δεδομένα $input_data_i$ δεν βρίσκονται στην κρυφή μνήμη του πράκτορα εκτέλεσης, διαφορετικά είναι ψευδής (=0). Η μεταβλητή $size_input_data_i$ είναι το πραγματικό μέγεθος δεδομένων του αντίστοιχου ορίσματος εισόδου. Το γινόμενο της συνάρτησης *data_in_cache() με το* $size_input_data_i$ είναι μηδέν εφόσον τα δεδομένα βρίσκονται στην κρυφή μνήμη και είναι $size_input_data_i$ όταν τα δεδομένα δεν βρίσκονται στην κρυφή μνήμη. Το μέγεθος των δεδομένων εισόδου που υπολογίζουμε μπορεί τελικά να είναι διαφορετικό από το μέγεθος των δεδομένων που θα μεταφερθούν και αυτό οφείλεται κυρίως σε πιθανή συμπίεση των δεδομένων πριν την αποστολή, γεγονός το οποίο δεν μπορεί να είναι γνωστό από πριν. Η μεταβλητή *average_output_data* είναι το μέσο μέγεθος των παραγόμενων δεδομένων εξόδου όπως δίνεται από το στατιστικό προφίλ του κάθε αλγόριθμου (όπως περιγράφεται αργότερα).

Στη συνέχεια, ο πράκτορας εκτέλεσης προχωράει στον υπολογισμό του χρόνου που χρειάζεται για να μεταφερθούν όλα τα δεδομένα από και προς την εφαρμογή του χρήστη. Το αποτελεσματικό εύρος μεταφοράς δεδομένων (effective bandwidth) του δικτύου μεταξύ του πράκτορα εκτέλεσης και της εφαρμογής του χρήστη δίνεται από την παράμετρο *d*, η οποία εξαρτάται από πολλούς παράγοντες: τεχνολογία δικτύου (εύρος μεταφοράς, πρωτόκολλο, τοπολογία), φόρτος στο δίκτυο τη συγκεκριμένη στιγμή, άλλους παράγοντες όπως κάρτες δικτύου, εύρος μεταφοράς δεδομένων διαύλου, λειτουργικό σύστημα, διακόπτες δικτύου, gateways, routers, κλπ. Στην συγκεκριμένη περίπτωση υποθέτουμε ότι έχουμε το ιδανικό δίκτυο, με πλήρη διασύνδεση μεταξύ των κόμβων του και με αποτελεσματικό εύρος μεταφοράς δεδομένων ίσο με το πραγματικό εύρος μεταφοράς δεδομένων. Επομένως, ο χρόνος που χρειάζεται για να μεταφερθούν όλα τα δεδομένα από και προς την εφαρμογή του χρήστη δίνεται στην παρακάτω σχέση:

$$est_transfer_time = \frac{est_input_data_transfered + est_output_data}{d} \qquad [4]$$

4.3.2 Υπολογισμός Χρόνου Εκτέλεσης

Ο αναμενόμενος χρόνος εκτέλεσης υπολογίζεται ως εξής:

$$est_execution_time = \frac{average_execution_time(size_input_data)}{CPU_index_i} \qquad [5]$$

όπου η μεταβλητή *size_input_data* εκφράζει το μέγεθος των συνολικών δεδομένων εισόδου και το αποτέλεσμα της συνάρτησης *average_execution_time(size_input_data)* εκφράζει το χρόνο που χρειάζεται για την εκτέλεση του αλγόριθμου χρησιμοποιώντας το συγκεκριμένο μέγεθος δεδομένων εισόδου και θεωρώντας τον μοναδιαίο υπολογιστή. Η πληροφορία αυτή αντλείται από το στατιστικό προφίλ του αλγόριθμου.

Ο αναμενόμενος χρόνος εκτέλεσης υπολογίζεται σαν να μην εκτελείται καμία άλλη διεργασία στο συγκεκριμένο επεξεργαστή του υπολογιστικού κόμβου. Συνήθως όμως αυτή δεν είναι η πραγματική κατάσταση. Ο ίδιος επεξεργαστής μπορεί να χρησιμοποιείται για την εκτέλεση άλλων διεργασιών, είτε από άλλους χρήστες, είτε από διεργασίες που ο ίδιος πράκτορας εκτελεί. Πρέπει να λάβουμε υπόψη τον πιθανό επιπλέον φόρτο του λειτουργικού συστήματος ώστε να ικανοποιήσει όλες αυτές τις διεργασίες. Διαλέγουμε να αγνοήσουμε τις διεργασίες που εκτελούνται από άλλους χρήστες καθώς και το επιπλέον κόστος του λειτουργικού συστήματος, και επικεντρωνόμαστε μόνο στις διεργασίες που ο συγκεκριμένος πράκτορας εκτελεί. Έτσι για να διορθώσουμε τον αναμενόμενο χρόνο εκτέλεσης, προσθέτουμε τον αναμενόμενο εναπομείναντα χρόνο εκτέλεσης των υπόλοιπων διεργασιών επεξεργασίας που εκτελούνται στο μηχάνημα (*rem_exec_time*).

$$rem_exec_time = \\ = \sum_{i=1}^{M} \begin{cases} est_execution_time_i \le est_execution_time : est_execution_time_i \\ est_execution_time_i > est_execution_time : est_execution_time \end{cases} \quad [6]$$

Δηλαδή, ο τελικός αναμενόμενος χρόνος εκτέλεσης είναι:

$$est_execution_time_final = est_execution_time + rem_exec_time \quad [7]$$

Τέλος, ο συνολικός αναμενόμενος χρόνος εκτέλεσης υπολογίζεται προσθέτοντας το χρόνο μεταφοράς και τον αναμενόμενο χρόνο εκτέλεσης, όπως δίνεται στη σχέση [1].

4.4 Υπολογισμός Κόστους Εκτέλεσης

4.4.1 Υπολογισμός Κόστους Χρήσης Μνήμης

Για κάθε υπηρεσία έχουμε στο στατιστικό προφίλ της ένα συντελεστή χρήσης μνήμης, ο οποίος ορίζει (χονδρικά) το μέγεθος της πραγματικής χρήσης μνήμης τη στιγμή που εκτελείται η υπηρεσία, σε σχέση με τα δεδομένα εισόδου της υπηρεσίας.

$$est_memory_usage = MemUsageFactor * \sum_{i=1}^{N} size_input_data_i \quad [8]$$

όπου MemUsageFactor είναι ένας συντελεστής ο οποίος δείχνει το μέγεθος της μνήμης που χρησιμοποιεί ο κάθε αλγόριθμος συναρτήσει των δεδομένων εισόδου του. Το κόστος της χρήσης της μνήμης υπολογίζεται ως εξής:

$$cost_memory = priceMemory * est_memory_usage \quad [9]$$

όπου *priceMemory* είναι η τιμή που χρεώνει ο διαχειριστής του συστήματος για κάθε byte μνήμης που χρησιμοποιείται από τον αλγόριθμο. Επιλεκτικά, μπορούμε να αγνοήσουμε το κόστος για τη χρήση της μνήμης ή να το αντικαταστήσουμε με ένα διαφορετικό τρόπο υπολογισμού , ανάλογα με την περίπτωση.

4.4.2 Υπολογισμός κόστους μεταφοράς

Το κόστος μεταφοράς δεδομένων από και προς το απομακρυσμένο μηχάνημα, υπολογίζεται ως εξής:

$$cost_transfer = priceNet * (est_input_data_transfered + est_output_data) \quad [10]$$

Όπου *priceNet* είναι η τιμή που το δίκτυο χρεώνει για κάθε byte δεδομένων που μεταφέρεται. Φυσικά η χρέωση μπορεί να μη γίνεται ανά byte αλλά να ακολουθεί διαφορετικό τρόπο, όπως χρέωση ανά σύνδεση ή εφάπαξ χρέωση. Στην περίπτωση αυτή ο κάθε πράκτορας είναι ελεύθερος να χρησιμοποιήσει διαφορετικό τρόπο υπολογισμού του κόστους μεταφοράς.

4.4.3 Υπολογισμός τελικού κόστους εκτέλεσης

Το κόστος της εκτέλεσης της υπηρεσίας υπολογίζεται ως εξής:

$$cost_execution = priceCPU * est_execution_time \quad [11]$$

Όπου *priceCPU* είναι η τιμή που ο διαχειριστής του συστήματος χρεώνει για κάθε δευτερόλεπτο επεξεργασίας. Φυσικά η χρέωση μπορεί να μη γίνει συναρτήσει του χρόνου επεξεργασίας, αλλά να ακολουθεί διαφορετικό τρόπο, ανάλογα με την περίπτωση. Το συνολικό κόστος επεξεργασίας είναι το άθροισμα του κόστους εκτέλεσης, μεταφοράς και μνήμης:

$$total_cost = cost_transfer + cost_execution + cost_memory \quad [12]$$

4.5 Ενημέρωση Στατιστικού Προφίλ

Μόλις τελειώσει η εκτέλεση μιας υπηρεσίας, το μέσο μέγεθος των δεδομένων που παράγει η κάθε υπηρεσία επαναϋπολογίζεται, λαμβάνοντας υπόψη τα δεδομένα που μόλις παράχθηκαν. Ο υπολογισμός αυτός χρησιμοποιεί ένα μοντέλο αυτοπαλινδρομικής (autoregressive) συνάρτησης με βήμα 1 και βάρος πρόσφατου παρελθόντος ίσο με w. Το μοντέλο αυτό μπορεί εύκολα να επεκταθεί (ή να αλλαχθεί) ώστε να χρησιμοποιήσει ένα μοντέλο αυτοπαλινδρομικής συνάρτησης με βήμα N [72].

$$average_output_data = w * real_output_data + (1-w) * average_output_data \quad [13]$$

Αντίστοιχα, ο μέσος χρόνος εκτέλεσης της υπηρεσίας επαναϋπολογίζεται μετά το τέλος της εκτέλεσης της υπηρεσίας. Ο δείκτης *CPU_index* χρησιμοποιείται για να μετατραπεί ο πραγματικός χρόνος στις μονάδες που χρησιμοποιούνται στους υπολογισμούς. Ο υπολογισμός αυτός χρησιμοποιεί ένα μοντέλο αυτοπαλινδρομικής συνάρτησης με βήμα 1 και βάρος πρόσφατου παρελθόντος ίσο με u.

$$average_execution_time =$$

$$= u * real_execution_time * CPU_index_i + (1-u) * average_execution_time \qquad [14]$$

Οι μεταβλητές *average_output_data, average_execution_time* αποθηκεύονται στο στατιστικό προφίλ της κάθε υπηρεσίας. Οι μεταβλητές αυτές μπορούν να υπολογιστούν με δύο τρόπους, είτε ανεξάρτητα από το μέγεθος των δεδομένων εισόδου, είτε ως συνάρτηση του μεγέθους των δεδομένων εισόδου. Για παράδειγμα θα μπορούσαμε να είχαμε μία μεταβλητή που να περιγράφει τον μέσο χρόνο εκτέλεσης ανεξάρτητα από το μέγεθος των δεδομένων εισόδου, και πολλές μεταβλητές, μία για κάθε εύρος δεδομένων που χρησιμοποιούνται στον συγκεκριμένο αλγόριθμο.

ΚΕΦΑΛΑΙΟ 5 Υλοποίηση

Η ανάπτυξη υπηρεσιών επεξεργασίας δεδομένων απαιτεί το περιβάλλον που τις υποστηρίζει να είναι επεκτάσιμο και ευέλικτο. Η λειτουργική και αρχιτεκτονική δομή του περιβάλλοντος όπως παρουσιάστηκε σε προηγούμενα κεφάλαια, δίνει κατ' αρχήν την δυνατότητα να ικανοποιηθούν οι παραπάνω απαιτήσεις, αλλά χρειάζεται και συγκεκριμένη επιπλέον προσπάθεια κατά την υλοποίηση.

5.1 Περιορισμοί - Δυσκολίες

Το υπολογιστικό κατανεμημένο σύστημα στο οποίο απευθύνεται το προτεινόμενο περιβάλλον είναι ετερογενές ως προς ένα πλήθος παραγόντων: από το λειτουργικό σύστημα, και την εσωτερική αρχιτεκτονική του επεξεργαστή, μέχρι τα εργαλεία που χρησιμοποιούνται για την ανάπτυξη της εφαρμογής, τη διεπιφάνεια χρήσης καθώς και τα διάφορα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται στο δίκτυο. Αναλυτικά:

- ***Λειτουργικό Σύστημα***: Υπάρχει μία πληθώρα λειτουργικών συστημάτων και ομάδων λειτουργικών συστημάτων (BSD-UNIX, System-V UNIX, Windows NT) τα οποία καθημερινά εξελίσσονται και παρουσιάζονται σε νέες εκδόσεις. Ο προγραμματιστής πρέπει να ανταπεξέλθει σε διαφορετικούς τρόπους προγραμματισμού του δικτύου, σε διαφορετικά μοντέλα διεργασιών και σχημάτων για την παράλληλη εκτέλεση, σε διαφορετικές προγραμματιστικές βιβλιοθήκες, και σε πολλές άλλες λεπτομέρειες οι οποίες είναι διαφορετικές σε κάθε λειτουργικό σύστημα.

- ***Αρχιτεκτονική Επεξεργαστή***: Ο επεξεργαστής που χρησιμοποιείται μπορεί να έχει διαφορετική αναπαράσταση των δεδομένων (Big Indian, Little Indian) ανάλογα με τον κατασκευαστή του (Intel, SPARC, PowerPC, MIPS). Σαν αποτέλεσμα τα δεδομένα που ανταλλάσσονται μεταξύ υπολογιστών με διαφορετική αρχιτεκτονική επεξεργαστή, πρέπει να υποστούν την κατάλληλη μετατροπή.

- ***Διεπιφάνεια χρήσης***: Η διεπιφάνεια χρήσης ή ο τρόπος με τον οποίο παρουσιάζονται οι διάφορες λειτουργίες στο χρήστη, τις περισσότερες φορές διαφέρει μεταξύ πολλών λειτουργικών συστημάτων. Παραδείγματα διαφορετικών τρόπων παρουσίασης λειτουργιών στο χρήστη χρησιμοποιούν τα Windows-NT, τα X-Windows και τα Open Windows. Σαν αποτέλεσμα ο προγραμματιστής πρέπει να χρησιμοποιήσει διαφορετικά αντικείμενα για να κατασκευάσει την διεπιφάνεια χρήσης, καθώς και πρέπει να λάβει υπόψη του τους διάφορους προτεινόμενους τρόπους παρουσίασης λειτουργιών που επιθυμεί το κάθε περιβάλλον, ώστε η λειτουργία της εφαρμογής να μην ξενίζει τον χρήστη.

- ***Μεταφραστές (compilers)***: Ακόμη, τα εργαλεία και οι μεταφραστές που πρέπει να χρησιμοποιήσει ο χρήστης είναι διαφορετικά στα διάφορα περιβάλλοντα. Για παράδειγμα, ακόμα και η διαδικασία της απλής μετάφρασης ενός προγράμματος μπορεί να προκαλεί προβλήματα λόγω της διαφορετικής υλοποίησης του πρότυπου της C++ που χρησιμοποιείται.

Εκτός από τους παραπάνω περιορισμούς που θέτει η ανομοιογένεια του υπολογιστικού συστήματος, οι επιθυμητές λειτουργίες του περιβάλλοντος καθώς και γενικότερες αρχές προγραμματισμού δημιουργούν τις παρακάτω απαιτήσεις:

- ***Ευκολία επέκτασης***: Το περιβάλλον πρέπει να είναι επεκτάσιμο με μεγάλη ευκολία ως προς την λειτουργικότητα, δηλαδή πρέπει να υποστηρίζεται η προσθήκη νέων λειτουργικών οντοτήτων με ελάχιστες κατά το δυνατόν αλλαγές στα υπόλοιπα κομμάτια.

- ***Δυνατότητα διασύνδεσης***: Υπάρχει η απαίτηση για δυνατότητα διασύνδεσης και συνεργασίας με άλλα πληροφοριακά συστήματα, άλλες εφαρμογές και περιβάλλοντα, χωρίς να πρέπει να εκτεθούν οι εσωτερικές λεπτομέρειες της υλοποίησης.

- ***Επαναχρησιμοποίηση πηγαίου κώδικα***: Είναι επιθυμητή η δυνατότητα μιας αρχικής γενικής υλοποίησης βασικών στοιχείων και μηχανισμών, τα οποία να επαναχρησιμοποιούνται με τις ελάχιστες κατά το δυνατόν αλλαγές για την υλοποίηση του περιβάλλοντος.

- ***Ευκολία Ανεύρεσης Σφαλμάτων***: Η ανεύρεση σφαλμάτων κώδικα σε ένα κατανεμημένο σύστημα είναι μία πολύ δύσκολη διαδικασία, και απαιτεί χρήση ειδικών μηχανισμών για την διευκόλυνσή της.

5.2 Επιλογές Υλοποίησης

Για να απομονωθεί η υλοποίηση του προτεινόμενου περιβάλλοντος από τους διάφορους παράγοντες που αναφέρονται στην προηγούμενη ενότητα, έγιναν διάφορες επιλογές όπως περιγράφονται παρακάτω.

5.2.1 Γλώσσα Προγραμματισμού

Με δεδομένη την προτεινόμενη κατανεμημένη αρχιτεκτονική πρακτόρων, επιβάλλεται η χρήση γλώσσας προγραμματισμού που να επιτρέπει την άμεση εφαρμογή οντοκεντρικών σχεδίων [84] με φυσικό τρόπο, να επιτρέπει τη χρήση προχωρημένων τεχνικών προγραμματισμού, όπως χρήση σχεδιαστικών προτύπων [85,86], καθώς και να υποστηρίζει την ανάπτυξη μεγάλων σε μέγεθος εφαρμογών οι οποίες κατά κανόνα παρουσιάζουν μεγάλες δυσκολίες υλοποίησης [87]. Με δεδομένα την απαίτηση του περιβάλλοντος για

άμεση αλληλεπίδραση και έλεγχο του επεξεργαστή, καθώς και τον αριθμό και μέγεθος των μηνυμάτων που ανταλλάσσονται, η γλώσσα ανάπτυξης πρέπει να επιτρέπει προγραμματισμό του λειτουργικού συστήματος σε χαμηλό επίπεδο τόσο σε επίπεδο δικτύου [88], αλλά και γενικότερα [89].

Χαμηλού επιπέδου προγραμματισμό προσφέρει η ευρέως διαδεδομένη γλώσσα C [90], που δεν καλύπτει όμως τις υπόλοιπες προϋποθέσεις. Οι περισσότερες από τις προϋποθέσεις ικανοποιούνται από την Java [91]. Η Java ωστόσο, αν και επιτρέπει προγραμματισμό δικτύου, δεν επιτρέπει χαμηλού επιπέδου προγραμματισμό λόγω των διαφόρων περιορισμών που θέτει λόγω των μηχανισμών που χρησιμοποιεί για την ασφάλεια των προγραμμάτων. Η τελική επιλογή ήταν η γλώσσα C++, η οποία ικανοποιεί πλήρως όλες τις παραπάνω απαιτήσεις [92]. Να σημειωθεί ότι η Java προσφέρει σημαντικές δυνατότητες, όπως ανεξαρτησία από το λειτουργικό σύστημα και βοήθεια διάφορων έτοιμων βιβλιοθηκών, και θα μπορούσε να χρησιμοποιηθεί στο μέλλον για την ανάπτυξη συγκεκριμένων κομματιών του συστήματος (κυρίως με δεδομένη την ιδιαίτερα γρήγορη εξέλιξη και σταθεροποίησή της). Οι ευκολίες που προσφέρονται από την Java, αντικαταστάθηκαν στη συγκεκριμένη C++ υλοποίηση με την χρήση εργαλείων ανάπτυξης όπως το ACE, τη zApp, τα Tools++, που περιγράφονται αναλυτικότερα στις επόμενες παραγράφους.

5.2.2 Ανεξαρτησία από Λειτουργικό Σύστημα και Τρόπο Προγραμματισμού του Δικτύου

Για την ανάπτυξη του συστήματος και της επικοινωνίας των διαφόρων οντοτήτων που το αποτελούν με τρόπο ανεξάρτητο από όλους τους παράγοντες που σχετίζονται με το λειτουργικό σύστημα που χρησιμοποιείται, η υλοποίηση βασίστηκε στο πλαίσιο εργασίας ADAPTIVE Communication Environment (ACE) [93]. Το ACE υλοποιεί ένα σύνολο από βασικά σχεδιαστικά πρότυπα [85,86] τα οποία απλουστεύουν την ανάπτυξη επικοινωνιακού λογισμικού [93,94]. Επίσης, παρέχει μία πλούσια συλλογή από wrappers, διάφορες κατηγορίες κλάσεων καθώς και πλαίσια εργασίας που υλοποιούν συχνά εμφανιζόμενες διεργασίες επικοινωνίας σε ένα ευρύ φάσμα λειτουργικών συστημάτων. Στη συγκεκριμένη υλοποίηση, χρησιμοποιήθηκαν οι μηχανισμοί που παρέχει το ACE για τη δημιουργία και διαχείριση των διεργασιών ώστε να απαλειφθούν οι διαφοροποιήσεις που εμφανίζονται στα διάφορα λειτουργικά συστήματα [95]. Χρησιμοποιήθηκαν επίσης σχεδιαστικά πρότυπα όπως το Reactor [96], το Acceptor και το Connector [97] για να μειωθεί η πολυπλοκότητα των επικοινωνιών μεταξύ των διαφόρων οντοτήτων του συστήματος.

5.2.3 Επαναχρησιμοποίηση Πηγαίου Κώδικα

Τα σχεδιαστικά πρότυπα Acceptor και Connector χρησιμοποιήθηκαν για την απομόνωση του τρόπου διασύνδεσης της επικοινωνίας των διαφόρων κομματιών της κατανεμημένης εφαρμογής από το λειτουργικό τους μέρος. Έτσι, ο διαχειριστής, ο πράκτορας εκτέλεσης και η εφαρμογή του χρήστη μοιράζονται τον ίδιο μηχανισμό για τη διασύνδεση και

επικοινωνία ενώ έχουν διαφορετικό λειτουργικό μέρος. Με τη χρήση των συγκεκριμένων προτύπων, αναπτύχθηκε ένα πολύ πιο επαναχρησιμοποιήσιμο και επεκτάσιμο επικοινωνιακό λογισμικό. Η χρήση όλων αυτών των σχεδιαστικών προτύπων, καθώς και άλλων [85,86] (που δεν αναφέρονται εδώ αναλυτικά), είχε σαν αποτέλεσμα πολλά από τα κομμάτια του λογισμικού που αναπτύχθηκαν να χρησιμοποιηθούν αυτούσια σε αρκετά διαφορετικά σημεία της υλοποίησης. Παράδειγμα χρήσης των σχεδιαστικών προτύπων παρουσιάζεται στο Δείγμα Κώδικα 1, όπου ορίζεται ο γενικότερος τρόπος με τον οποίο θα γίνεται η προσπέλαση σε ένα χειριστή μηνυμάτων, κάτι που επαναχρησιμοποιείται ευρύτατα σε κάθε κομμάτι του συστήματος που χρειάζεται δυνατότητες επικοινωνίας.

```
class DMessageHandler :
public ACE_Svc_Handler<ACE_SOCK_STREAM, ACE_NULL_SYNCH>
{
public:
      DMessageHandler(void);
      ~DMessageHandler();

      // Hooks for opening and closing handlers.
      //
      virtual int open (void * = 0);
      virtual int close (u_long);
      virtual int handle_timeout
            (const ACE_Time_Value &tv, const void *arg);

protected:
      virtual int handle_input (ACE_HANDLE);
      virtual int handle_close
            (ACE_HANDLE fd, ACE_Reactor_Mask close_mask );
      virtual void destroy (void);
};

// Message Handlers Lists
//
typedef RWTPtrDlist<DMessageHandler> DMSG_HANDLER_LIST;
typedef RWTPtrDlistIterator<DMessageHandler> DMSG_HANDLER_LIT;

//  Acceptor/Connector using Message_Handler and SOCK_Acceptor
//
typedef ACE_Acceptor <DMessageHandler, ACE_SOCK_ACCEPTOR>
      DMessageAcceptor;
typedef ACE_Connector <DMessageHandler, ACE_SOCK_CONNECTOR>
      DMessageConnector;

// Our global Reactor Singleton.
//
typedef ACE_Singleton<ACE_Reactor, ACE_Null_Mutex> REACTOR;
```

Δείγμα Κώδικα 1: Παράδειγμα χρήσης των διαφόρων σχεδιαστικών προτύπων.

5.2.4 Επεκτασιμότητα και Διασύνδεση Ετερογενών Εφαρμογών

Μία σημαντική πρόκληση για τα σημερινά πληροφοριακά συστήματα, είναι η επίλυση του προβλήματος της διασύνδεσης μεταξύ ετερογενών εφαρμογών. Λύσεις οι οποίες έχουν προταθεί περιλαμβάνουν μια πληθώρα διαφορετικών αρχιτεκτονικών για κατανεμημένο προγραμματισμό, όπως CORBA, OLE, SOM, OpenDoc, DCOM, DCE (για μια επισκόπηση της σημερινής κατάστασης κοιτάξτε το [98]). Μια από τις πιο πολλά υποσχόμενες αρχιτεκτονικές είναι η CORBA (Common Object Request Broker Architecture) [98,99], μία εξελισσόμενη ανοιχτή αρχιτεκτονική που γίνεται πρότυπο από τον οργανισμό OMG (Object Management Group) [100]. Η αρχιτεκτονική CORBA αυτοματοποιεί πολλές κοινές λειτουργίες οι οποίες γίνονται κατά τον προγραμματισμό του δικτύου, όπως η εγγραφή αντικειμένων, ανεύρεση και ενεργοποίηση, απο-πολύπλεξη αιτήσεων, χειρισμός σφαλμάτων, μετατροπή παραμέτρων και εκτέλεση λειτουργιών.

Το υπόδειγμα του πρότυπου Reactor που χρησιμοποιήθηκε στην υλοποίηση υποστηρίζει την απο-πολύπλεξη και διανομή γεγονότων σε πολλαπλούς χειριστές γεγονότων (event handlers), οι οποίοι ενεργοποιούνται ταυτόχρονα με τη λήψη πολλαπλών γεγονότων. Παραδείγματα τέτοιων χειριστών γεγονότων είναι χειριστές για γεγονότα που παράγονται από το επίπεδο του δικτύου (socket), για γεγονότα από τη διεπιφάνεια χρήσης, για γεγονότα από μετρητές (timers) και διάφορες ειδοποιήσεις (signals), καθώς και για γεγονότα από το CORBA. Έτσι δίνεται η δυνατότητα για άμεση αντιστοίχηση των μηνυμάτων που παρέχει το CORBA στα εσωτερικά μηνύματα του περιβάλλοντος και αντίστροφα, ώστε η υλοποίηση να είναι συμβατή με CORBA και συνεπώς άμεσα ολοκληρώσιμη με άλλες εφαρμογές συμβατές με αυτό το πρότυπο.

Τέλος, χρησιμοποιώντας το υπόδειγμα Reactor μπορούμε να συμπεριλάβουμε στο μέλλον γεγονότα από διαφορετικές πηγές, όπως εργαλειοθήκες για σύνδεση με DICOM και HL7 [101], για την υποστήριξη επεξεργασίας ιατρικών δεδομένων σε ένα δίκτυο υπηρεσιών τηλε-ακτινολογίας [13].

5.2.5 Εσωτερική Επικοινωνία και Ανταλλασσόμενα Μηνύματα

Τα χαρακτηριστικά του πρότυπου CORBA το καθιστούν ιδανική υποψήφια μέθοδο και για την εσωτερική επικοινωνία μεταξύ των διαφόρων κομματιών του συστήματος. Ωστόσο, υπάρχουν δύο σημαντικοί ανασταλτικοί παράγοντες:

- Το CORBA δεν είναι υλοποιημένο σε όλα τα λειτουργικά συστήματα που θα μπορούσαμε να συναντήσουμε σε ένα κατανεμημένο υπολογιστικό σύστημα επεξεργασίας εικόνων (για παράδειγμα λειτουργικά συστήματα υπερ-υπολογιστών).

- Υπάρχουν πολλές υλοποιήσεις του πρότυπου CORBA, είτε εμπορικές είτε για κοινή χρήση. Οι εκδόσεις κοινής χρήσης δεν είναι μεταφέρσιμες σε πολλά λειτουργικά συστήματα και η υποστήριξη τους είναι αμφίβολη. Σε αντίθεση, οι εμπορικές εκδόσεις είναι μεταφέρσιμες σε περισσότερα λειτουργικά συστήματα, αλλά με μεγάλο κόστος για

την ανάπτυξη και τη χρήση τους, μια που επιβάλλουν κόστος για άδεια χρήσης σε κάθε μηχάνημα που θα χρησιμοποιήσει εφαρμογή που χρειάζεται διασύνδεση μέσω CORBA.

Σαν αποτέλεσμα δεν χρησιμοποιήθηκε το CORBA για την μεταφορά των διαφόρων μηνυμάτων που ανταλλάσσουν τα εσωτερικά τμήματα της εφαρμογής. Αντίθετα, τα διάφορα μηνύματα που ανταλλάσσονται έχουν υλοποιηθεί σαν μία εσωτερική ιεραρχία κλάσεων. Στο Δείγμα Κώδικα 2 παρουσιάζεται η γενική κλάση μηνυμάτων από την οποία κληρονομούν οι υπόλοιπες, ενώ στο Δείγμα Κώδικα 3 παρουσιάζεται η υλοποίηση ενός συγκεκριμένου μηνύματος.

```
class DMsg
{
public:
      DMsg();
      virtual ~DMsg();
      virtual int Read(char *str, NET_STREAM *peer = 0) = 0;
      virtual void Write (NET_STREAM *peer)  = 0;

      short Type(void);
      long  MsgId(void);
      sessionID &FromId ();
      sessionID &DestId ();
      void * &UserData ();

. . . . . .

protected:
      WORD type;
      LONG msgId;       // message Id. Identifies each message
      sessionID fromId; // Initiator of message
      sessionID destId; // Destination of message

private:
      void *user_data;         // user data assigned to message
      static long msgId_count; // counter of generated messages
};
```

Δείγμα Κώδικα 2: Η κλάση DMsg παρέχει τους βασικούς μηχανισμούς για την ανταλλαγή μηνυμάτων.

```
class DMsgReqAlgoInfo : public DMsg
{
public:
      DMsgReqAlgoInfo();
      ~DMsgReqAlgoInfo();

      virtual int Read(char *str, NET_STREAM *peer = 0);
      virtual void Write (NET_STREAM *peer);

      sessionID &Id ();                    { return id; }
      void SetAlgoId (long id)             { algo_id = id; }
      long GetAlgoId(void)                 { return algo_id; }

      void SetAlgoIdName (const char *fn);
      char *GetAlgoIdName(void)            { return algo_name; }

protected:
      sessionID id;
      long algo_id;
      char algo_name[FILE_NAMELEN + 1];
};
```

Δείγμα Κώδικα 3: Παράδειγμα ορισμού μηνύματος για την αίτηση πληροφορίας σχετικής με έναν αλγόριθμο.

5.2.6 Μηχανισμός Καταγραφής Γεγονότων

Για να διευκολύνουμε τη διαδικασία της διόρθωσης σφαλμάτων χρησιμοποιήθηκε ένας μηχανισμός κατανεμημένης καταγραφής που παρέχει, σε ένα κεντρικό σημείο, την ακριβή σειρά με την οποία συνέβησαν όλα τα γεγονότα. Με αυτόν το τρόπο μπορούμε να βρούμε πολύ εύκολα την πηγή του πιθανού προβλήματος. Ο ίδιος μηχανισμός για την καταγραφή γεγονότων κατά την διόρθωση σφαλμάτων, μπορεί να χρησιμοποιηθεί και στην τελική έκδοση του συστήματος, με διαφορετικό σκοπό, την παρακολούθηση του περιβάλλοντος. Ο μηχανισμός καταγραφής χρησιμοποιεί τους ίδιους μηχανισμούς για επικοινωνία που χρησιμοποιούν και τα υπόλοιπα κομμάτια.

5.2.7 Διεπιφάνεια Χρήσης

Η διεπιφάνεια χρήσης έχει υλοποιηθεί χρησιμοποιώντας μια βιβλιοθήκη για τον προγραμματισμό διεπιφανειών χρήσης, την zApp [102]. Η βιβλιοθήκη zApp είναι μία

πλήρως οντοκεντρική βιβλιοθήκη που μοντελοποιεί τα διάφορα κομμάτια της διεπιφάνειας χρήσης με τρόπο αφαιρετικό. Επιτρέπει στο χρήστη να χρησιμοποιήσει λειτουργίες που υπάρχουν στα πιο γνωστά περιβάλλοντα διεπιφανειών χρήσης (Microsoft Windows [103], Motif και OpenWindows [104]), χωρίς ο χρήστης-προγραμματιστής να γνωρίζει σε πιο περιβάλλον αναφέρεται. Με αυτόν τον τρόπο ο χρήστης ασχολείται μόνο με λειτουργίες που του παρέχει η zApp και αφήνει τη βιβλιοθήκη είτε να κάνει την αντιστοίχηση των λειτουργιών στο πραγματικό περιβάλλον (εφόσον υπάρχει αντιστοίχηση), είτε να υλοποιήσει τη συγκεκριμένη λειτουργία στο ζητούμενο περιβάλλον. Η εμπειρία μας με τη χρήση της zApp δείχνει ότι είναι μία πολλά υποσχόμενη βιβλιοθήκη η οποία δίνει τα αναμενόμενα αποτελέσματα σε όλα τα λειτουργικά συστήματα που υποστηρίζει, πολλές φορές με τρόπο ευκολότερο από τη χρήση της βιβλιοθήκης που παρέχει κάποιο συγκεκριμένο λειτουργικό περιβάλλον. Η zApp παρουσιάζει πολλές ομοιότητες με το AWT (Abstract Windowing Toolkit), την αντίστοιχη βιβλιοθήκη της Java [91], μια που και τα δύο είναι οντοκεντρικά και παρέχουν μία αφαίρεση στην τελικά χρησιμοποιημένη διεπιφάνεια χρήσης. Το αποτέλεσμα είναι να δίνεται η δυνατότητα για άμεση και εύκολη μελλοντική υλοποίηση της διεπιφάνειας χρήσης σε περιβάλλον Java.

5.2.8 Βάση Αλγόριθμων

Θεωρήσαμε ότι η χρήση ενός κοινού συστήματος διαχείρισης βάσης δεδομένων (DBMS) όπως της ORACLE [105] και της Sybase [106], θα ήταν υπερβολική για χρήση στο συγκεκριμένο περιβάλλον. Θα είχε μεγάλο κόστος αγοράς, εγκατάστασης και συντήρησης, και θα χρειαζόταν σημαντικά περισσότερους υπολογιστικούς πόρους για να χρησιμοποιηθεί (περιορίζοντας έτσι τους άμεσα απαραίτητους πόρους για την επεξεργασία δεδομένων). Η υλοποίηση ακολούθησε μια απλή λύση, όπου η τοπική συλλογή αλγόριθμων και οι πληροφορίες που τους συνοδεύουν οργανώθηκαν σε αρχεία, τα οποία διαχειρίζεται το υποσύστημα διαχείρισης αρχείων του κάθε λειτουργικού συστήματος. Συγκεκριμένα, τα αρχεία της περιγραφής λειτουργίας των αλγόριθμων (wrapper) είναι απλά αρχεία κειμένου ώστε να μπορεί ο κάθε χρήστης με ένα οποιαδήποτε πρόγραμμα απλού κειμενογράφου να επέμβει και να διορθώσει ή να προσθέσει παραμέτρους. Η δομή που εφαρμόζεται σε κάθε αρχείο περιγραφής είναι αντίστοιχη με τη δομή που εφαρμόζουν τα Microsoft Windows στα αρχεία τύπου .INI που χρησιμοποιούν για τον ορισμό των διαφόρων παραμέτρων. Η πρόσβαση στα συγκεκριμένα αρχεία υλοποιήθηκε με τρόπο αντίστοιχο με τα Microsoft Windows και σε περιβάλλον UNIX, ώστε η εφαρμογή να είναι πλήρως μεταφερόμενη και να μην έχει εξαρτήσεις από το συγκεκριμένο λειτουργικό σύστημα. Παράδειγμα ενός αρχείου περιγραφής παρουσιάζεται στα Δείγματα Κώδικα Δείγμα Κώδικα 4 και Δείγμα Κώδικα 5. Στο πρώτο παρουσιάζεται ο τρόπος με τον οποίο περιγράφεται η δομή ενός κόμβου του δένδρου των αλγόριθμων και στο δεύτερο το αρχείο περιγραφής ενός συγκεκριμένου αλγόριθμου.

```
[Algorithm]
Type=1
Name=PBM Plus Algorithms
DescN=1
Desc1=Algorithms from the PBMPlus package
Count=6
Node1=pgmcleanup.ini
Node2=pgmedge.ini
Node3=pgmenhance.ini
Node4=pnmgamma.ini
Node5=pgmoil.ini
Node6=pgmgaussian.ini
```

Δείγμα Κώδικα 4: Δείγμα αρχείου περιγραφής κόμβου αλγόριθμων.

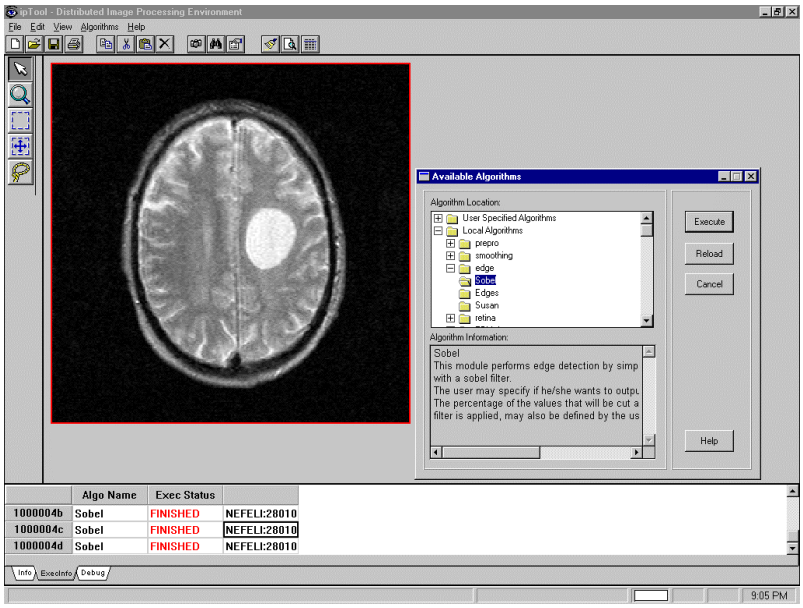
```
[Algorithm]
Type=2
Id=301
Name=Gaussian
DescN=5
Desc1=Performs gaussian smoothing.
Desc2=The size and the sigma value
Desc3=of the filter are entered,
Desc4=and the image is convolved with
Desc5=the related gaussian filter.

[Exe]
Numparams=4
Exepath=Algos\\Gaussian $IMAGE_IN
                        $IMAGE_OUT
                        $FILTER_SIZE
                        $FILTER_VARIANCE
[Param1]
Name=IMAGE_IN
Type=IMAGE
Format=RAW
Required=1
Input=1
```

```
[Param2]
Name=IMAGE_OUT
Type=IMAGE
Format=RAW
Required=1
Input=0

[Param3]
Name=FILTER_SIZE
Type=FLOAT
Format=FLOAT
Required=1
Input=1

[Param4]
Name=FILTER_VARIANCE
Type=FLOATFormat=FLOAT
Required=1
Input=1
```

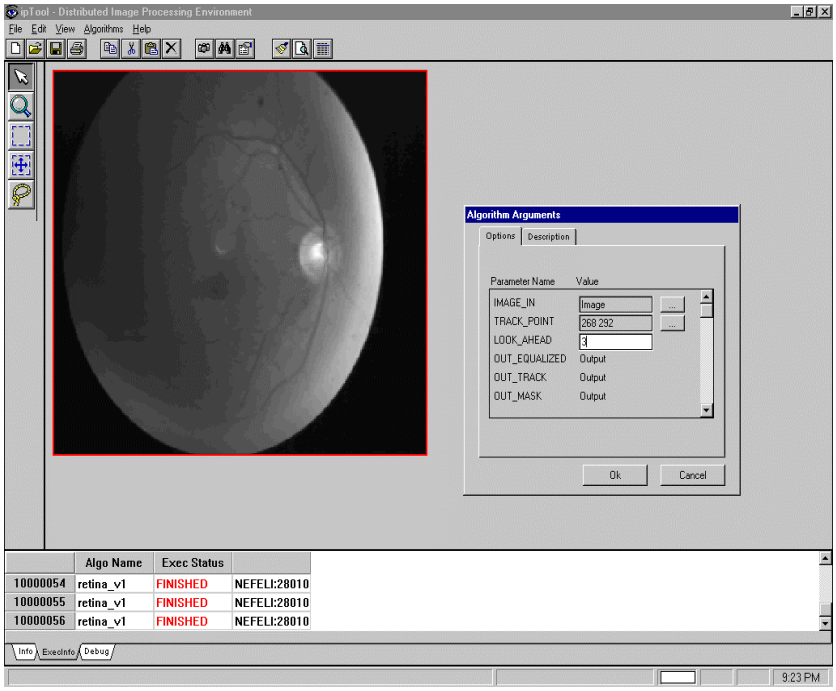
Δείγμα Κώδικα 5: Δείγμα αρχείου προφίλ αλγόριθμου (pgmgaussian.ini).

5.3 Αποτελέσματα

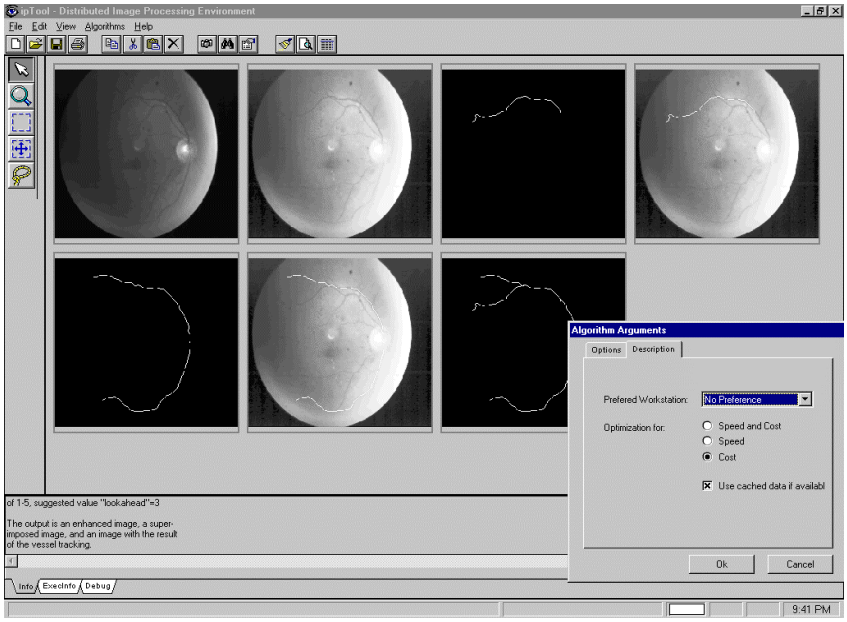
Η πρωτότυπη υλοποίηση του συστήματος παρέχει μία ειδική εφαρμογή για το χρήστη η οποία του επιτρέπει τη χρησιμοποίηση στο μέγιστο των υπηρεσιών που παρέχει το περιβάλλον. Στα Σχήματα 13-17 δίνονται μερικές οθόνες ενδεικτικές της λειτουργίας του περιβάλλοντος. Η εφαρμογή έχει ενσωματωμένες λειτουργίες για τις πιο συνηθισμένες εκτελέσεις αλγόριθμων που χρειάζονται απόκριση σε πραγματικό χρόνο, και σαν αποτέλεσμα δεν θεωρείται σκόπιμη η ανακατεύθυνση της εκτέλεσης τους σε ένα πράκτορα εκτέλεσης (όπως άλλωστε υποδεικνύουν και τα πειράματα προσομοίωσης που περιγράφονται στο Παράρτημα Α). Τέτοια παραδείγματα είναι διάφοροι απλοί αλγόριθμοι προ-επεξεργασίας (Negate, Contrast Enhance, Brightness Enhance). Στο περιβάλλον μέχρι σήμερα έχουν ενσωματωθεί αλγόριθμοι από πολλές διαφορετικές πηγές, μέσω της χρήσης του εκτελεστή αλγόριθμων και χωρίς τροποποίηση του κώδικά τους. Συγκεκριμένα, έχουν ενσωματωθεί αλγόριθμοι που παρέχουν βιβλιοθήκες όπως το PBMPlus [21] και το NETPBM [22], καθώς και αλγόριθμοι από συλλογές όπως το ORT (Object Recognition Toolkit) [107], και το Susan [108]. Τέλος, έχει ενσωματωθεί και αλγόριθμος που έχει αναπτυχθεί στα πλαίσια διπλωματικής εργασίας του Τμήματος Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης [109].



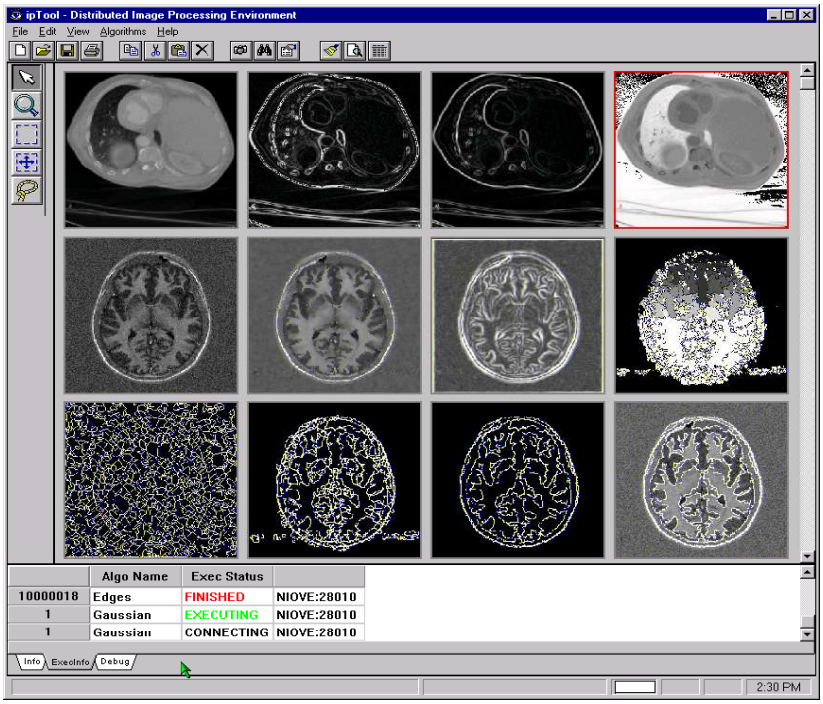
Σχήμα 13: Παράδειγμα χρήσης του συστήματος. Στο δεξί μέρος της οθόνης φαίνεται το παράθυρο επιλογής αλγόριθμων, όπου στο πάνω μέρος διακρίνεται το δένδρο της δομής των αρχείων ενώ στο κάτω μέρος φαίνονται οι πληροφορίες που συνοδεύουν τον επιλεγμένο αλγόριθμο.



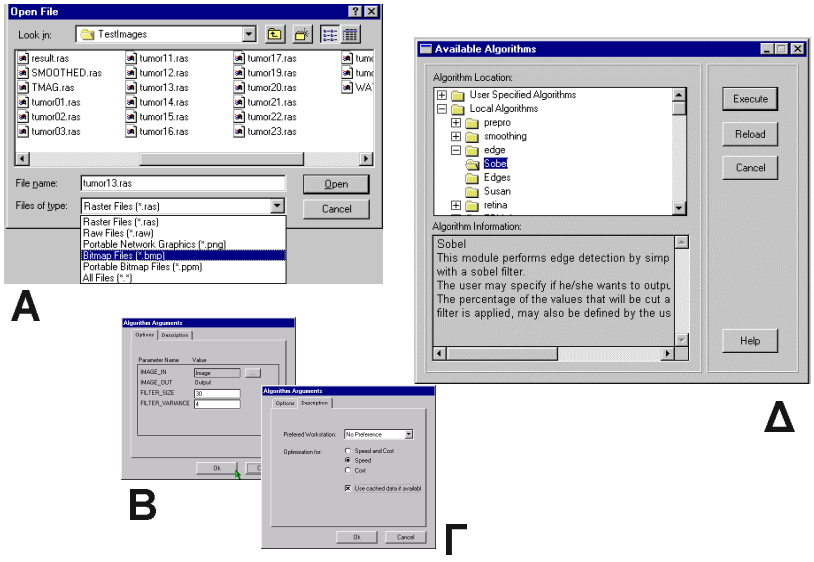
Σχήμα 14: Παράδειγμα χρήσης του συστήματος. Στο δεξί μέρος της οθόνης φαίνεται το παράθυρο εκκίνησης εκτέλεσης ενός αλγόριθμου, και συγκεκριμένα η σελίδα για την εισαγωγή τιμών στα ορίσματα εισόδου του αλγόριθμου.



Σχήμα 15: Παράδειγμα χρήσης του συστήματος. Στο κάτω-δεξί μέρος της οθόνης φαίνεται το παράθυρο εκκίνησης εκτέλεσης ενός αλγόριθμου, και συγκεκριμένα η σελίδα για την εισαγωγή κριτηρίων για την ανάθεση εκτέλεσης του αλγόριθμου.



Σχήμα 16: Παράδειγμα χρήσης του συστήματος. Ο αριθμός των εικόνων που εμφανίζονται ταυτόχρονα στην οθόνη είναι μεταβλητός και επιλέγεται από τον χρήστη. Στο κάτω μέρος της οθόνης φαίνεται το παράθυρο πληροφοριών, στην σελίδα που αντιστοιχεί σε πληροφορίες για την κατάσταση των εκτελέσεων. Συγκεκριμένα, διακρίνεται πληροφορία για μια εκτέλεση που έχει ολοκληρωθεί, έναν αλγόριθμο που εκτελείται, και μια ακόμα εκτέλεση που βρίσκεται στο στάδιο της αρχικής επικοινωνίας.



Σχήμα 17: Μερικά από τα βοηθητικά παράθυρα της εφαρμογής χρήστη. Επιλογή εικόνας (Α), παράθυρο εκκίνησης εκτέλεσης στη σελίδα εισαγωγής τιμών στα ορίσματα εισόδου του αλγόριθμου (Β) και στη σελίδα εισαγωγής κριτηρίων ανάθεσης εκτέλεσης (Γ), και επιλογή αλγόριθμου (Δ).

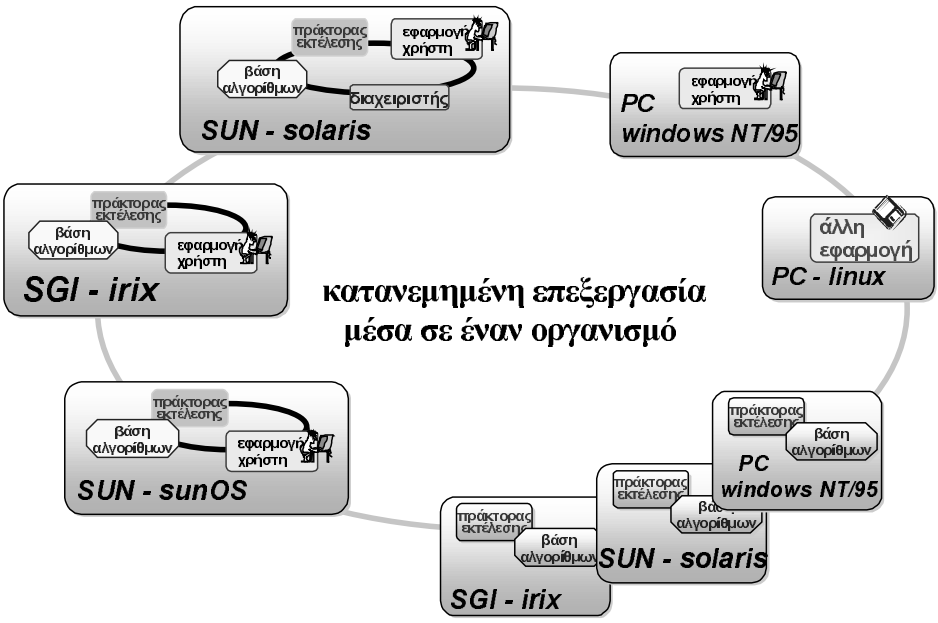
ΚΕΦΑΛΑΙΟ 6 Συμπεράσματα και Μελλοντική Εργασία

Είναι προφανές ότι η βιομηχανία λογισμικού βρίσκεται σε μία φάση σημαντικών αλλαγών. Μετατρέπεται από μία βιομηχανία προϊόντων σε μία βιομηχανία υπηρεσιών. Με αυτό το σκεπτικό, σήμερα μεγάλος αριθμός έργων ασχολείται με την ανάπτυξη νέων υπηρεσιών πληροφορικής για το ευρύ κοινό.

Στην εργασία αυτή παρουσιάζεται μία αρθρωτή αρχιτεκτονική αυτόνομων συνεργατικών πρακτόρων οι οποίοι μπορούν να χρησιμοποιηθούν για την ανάπτυξη δικτύων παροχής υπηρεσιών επεξεργασίας δεδομένων για μια πληθώρα χώρων εφαρμογής. Η βασική οντότητα της εφαρμογής είναι η κοινωνία των πρακτόρων εκτέλεσης. Κάθε πράκτορας εκτέλεσης αντιστοιχεί σε έναν από τους υπολογιστικούς πόρους του κατανεμημένου συστήματος. Ο πράκτορας εκτέλεσης είναι μια σύνθετη οντότητα με πολλαπλούς ρόλους. Η κύρια υπευθυνότητά του είναι η τοπική εκτέλεση διεργασιών. Ανάλογα με την εφαρμογή, άλλοι ρόλοι που προβλέπονται και μπορούν να ενισχυθούν κατάλληλα περιλαμβάνουν τον έλεγχο της κατάστασης των τοπικών πόρων και διαμόρφωση τοπικής πολιτικής για την χρήση και διάθεσή τους, έξυπνη διαχείριση τοπικής συλλογής εκτελέσιμων και σχετικής πληροφορίας, παρακολούθηση εκτέλεσης διεργασιών και εξαγωγή στατιστικών συμπερασμάτων, έξυπνη διαχείριση αποτελεσμάτων διεργασιών. Συνοψίζοντας, ο πράκτορας εκτέλεσης αναλαμβάνει να απομονώσει κάθε τι σχετικό με τους τοπικούς πόρους και τις τοπικές εκτελέσεις διεργασιών, τόσο από το υπόλοιπο περιβάλλον όσο και από τον χρήστη. Ανάμεσα στον χρήστη και στον πράκτορα εκτέλεσης υπάρχει (στις περιπτώσεις που χρειάζεται) μια άλλη οντότητα, ο πράκτορας εκτέλεσης ακολουθιών. Αυτός αναλαμβάνει την ανάλυση σύνθετων διεργασιών και την αποσύνθεσή τους στα δομικά κομμάτια που μπορούν να εκτελεστούν από μεμονωμένους πράκτορες εκτέλεσης. Επίσης, παρακολουθεί την πορεία της εκτέλεσης μιας σύνθετης διεργασίας και προγραμματίζει την επανεκτέλεση κάποιου συγκεκριμένου μέρους σε περίπτωση απρόσμενου σφάλματος σε κάποιον από τους πόρους του συστήματος. Οι παραπάνω πράκτορες εκτέλεσης (ακολουθίας ή διεργασίας) οργανώνονται από έναν κεντρικό διαχειριστή. Η κεντρική αυτή οντότητα εισάγεται για να υποστηρίξει μηχανισμούς αναγνώρισης χρηστών, διαμόρφωσης γενικής πολιτικής διαχείρισης πόρων και χρέωσης, έκδοσης λογαριασμών, κλπ. Η ανάθεση εκτέλεσης διεργασιών στην κατανεμημένη κοινωνία των πρακτόρων είναι ουσιαστικά μια κατανεμημένη διαδικασία και υλοποιείται μέσω δημοπρασίας. Τα κριτήρια για την δημιουργία της προσφοράς στη δημοπρασία, καθώς και τα κριτήρια για την ανάθεση της διεργασίας εξαρτώνται από την συγκεκριμένη εφαρμογή και βασίζονται στην (πιθανά) διαφοροποιημένη πολιτική του κάθε πράκτορα εκτέλεσης καθώς και σε γενικότερες πολιτικές που επιβάλλονται από τον διαχειριστή.

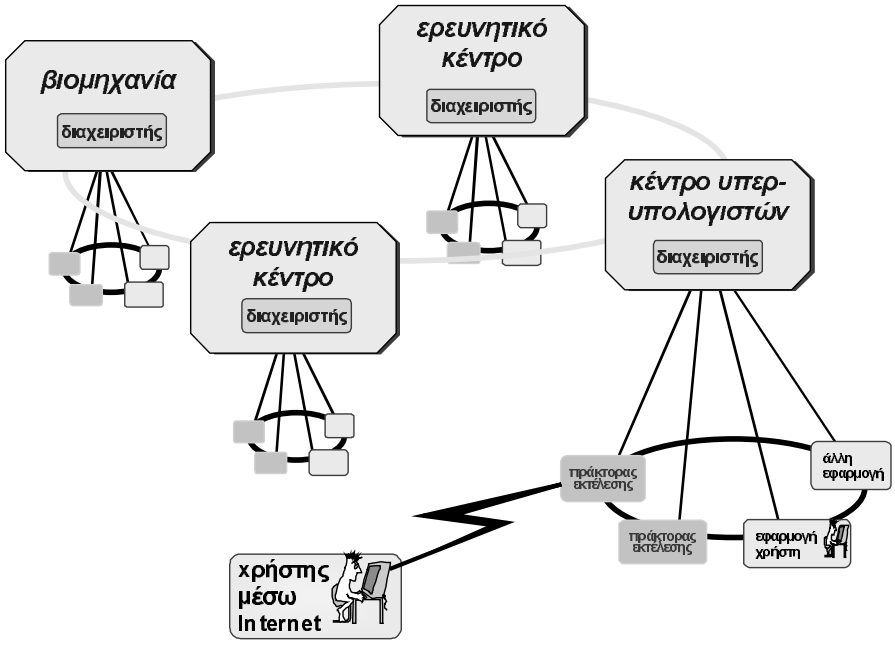
Στα πλαίσια της εργασίας, με βάση την προτεινόμενη αρχιτεκτονική, αναπτύχθηκε ένα περιβάλλον για παροχή υπηρεσιών επεξεργασίας εικόνας. Το περιβάλλον αυτό αναμένεται να έχει μία πληθώρα εφαρμογών. Ένα πρώτο σενάριο υποθέτει ένα δίκτυο μέσα σε ένα οργανισμό που αποτελείται από μία πληθώρα υπολογιστών με διαφορετικό υλικό και λειτουργικό σύστημα (Σχήμα 18). Κάθε υπολογιστής που είναι ικανός να εκτελέσει αλγόριθμους επεξεργασίας εικόνων, έχει έναν πράκτορα εκτέλεσης και μία βάση με τα

αντίστοιχα εκτελέσιμα. Οι χρήστες μπορούν να προσπελάσουν τις υπηρεσίες μέσω της εφαρμογής, η οποία υπάρχει σε διάφορους υπολογιστές στο δίκτυο. Κάθε χρήστης, ανεξάρτητα από τη τοποθεσία του μηχανήματός του μπορεί να προσπελάσει καθένα από τους αλγόριθμους που βρίσκονται στο δίκτυο και να τον εκτελέσει με διαφανή τρόπο, χωρίς να χρειάζεται να ξέρει την ακριβή τοποθεσία, ή τις απαιτήσεις σε υλικό και λογισμικό του κάθε εκτελέσιμου. Επιπλέον, η εκτέλεση ανατίθεται στον καλύτερο δυνατό επεξεργαστή στο δίκτυο (από όσους μπορούν να εκτελέσουν το συγκεκριμένο εκτελέσιμο). Μερικά μηχανήματα που προορίζονται για ειδική χρήση (π.χ. παράλληλες μηχανές) μπορεί να μην έχουν κάποιο χρήστη τοπικά, ενώ άλλα μηχανήματα δεν έχουν πράκτορα εκτέλεσης (όπως χαμηλών επιδόσεων μηχανές ή μηχανές με αποκλειστική χρήση για άλλες εφαρμογές). Στο σενάριο αυτό, ο οργανισμός μπορεί να είναι ένα ερευνητικό ινστιτούτο, εργαστήριο ενός πανεπιστήμιου, ή άλλος οργανισμός που χρησιμοποιεί ευρέως επεξεργασία εικόνας (π.χ. το σχεδιαστικό τμήμα μιας βιομηχανίας). Σε όλες αυτές τις εφαρμογές ένα σημαντικό πλεονέκτημα του προτεινόμενου περιβάλλοντος είναι η δυνατότητα χρήσης λογισμικού τρίτων κατασκευαστών και η ενοποίηση κάτω από το ίδιο περιβάλλον ανομοιογενών πόρων, καθώς και εκτελέσιμων με ανομοιογενείς απαιτήσεις όσο αναφορά στο λογισμικό και στο υλικό. Επιπρόσθετα, η διαχείριση πόρων είναι ένα πολύ σημαντικό εργαλείο για την καθημερινή εργασία σε παρόμοιους οργανισμούς, μια και ένας σημαντικός αριθμός από τους πόρους συνήθως χρησιμοποιείται για διεργασίες εκτός από επεξεργασία εικόνων, οι οποίες μπορεί να είναι λιγότερο απαιτητικές σε υπολογιστική ισχύ. Τέλος, η χρέωση βασισμένη σε εικονικά χρήματα μπορεί να βοηθήσει τον επαναπροσδιορισμό διεργασιών και την ανακατανομή πόρων μέσα στον οργανισμό.



Σχήμα 18: Το προτεινόμενο περιβάλλον ολοκληρώνει ανομοιογενείς υπολογιστικούς πόρους και εκτελέσιμα με διαφορετικές απαιτήσεις σε υλικό και λογισμικό, και παρέχει διάφανη εκτέλεση σε κατανεμημένο σύστημα ενός οργανισμού αξιοποιώντας όλους τους διαθέσιμους πόρους.

Ένα άλλο σημαντικό χαρακτηριστικό που υποστηρίζεται από το περιβάλλον είναι η δυνατότητα της εικονικής επέκτασης του οργανισμού ώστε να επιτρέπει σε χρήστες να εργάζονται εκτός του οργανισμού (π.χ. στο σπίτι τους). Έτσι ένας χρήστης που έχει την εφαρμογή στον υπολογιστή του (στο σπίτι του ή σε ένα φορητό) και που χρησιμοποιεί οποιαδήποτε σύνδεση (ασύρματη ή ενσύρματη) με τον οργανισμό, μπορεί να ξεκινήσει μία διεργασία που απαιτεί πολύ χρόνο ή να επεξεργαστεί αποτελέσματα μιας προηγούμενης αίτησης του. Επιπλέον, το περιβάλλον επιτρέπει το σχηματισμό εικονικών οργανισμών, επιτρέποντας σε διαφορετικούς οργανισμούς που αναπτύσσουν συμπληρωματικά συστήματα επεξεργασίας δεδομένων και συνήθως έχουν διαφορετικό εξοπλισμό, να μοιραστούν τους πόρους τους με έναν ευέλικτο τρόπο πολιτικών χρήσης και χρέωσης (Σχήμα 19). Αυτό μπορεί να αποδειχθεί πολύ χρήσιμο για την κοινή χρήση επιστημονικών αποτελεσμάτων και την αξιολόγηση νέων και υπαρκτών αλγόριθμων μεταξύ διαφορετικών οργανισμών χωρίς την ανάγκη για πρόσβαση στον πηγαίο κώδικα ή ακόμη στο εκτελέσιμο ενός αλγόριθμου. Ο εικονικός υπερ-οργανισμός μπορεί να περιέχει μέλη με ειδικό υλικό και το αντίστοιχο λογισμικό (όπως κέντρα υπερ-υπολογιστών με δίκτυα ακριβών παράλληλων υπολογιστών) που θα νοικιάζουν τους πόρους τους στους υπόλοιπους χρήστες. Ένα σημαντικό πλεονέκτημα του περιβάλλοντος είναι η δυνατότητα της διαφοροποίησης των διαφορετικών διαχειριστών ώστε να χειρίζονται τις εξωτερικές αιτήσεις με διαφορετική πολιτική όσον αναφορά την πρόσβαση, την κατανομή πόρων και τη χρέωση, έτσι ώστε κάθε φορά να αντικατοπτρίζουν τους συγκεκριμένους διακανονισμούς μεταξύ δύο οργανισμών.



Σχήμα 19: Δημιουργία εικονικού οργανισμού μέσω του προτεινόμενου περιβάλλοντος και παροχή υπηρεσιών σε εξωτερικούς χρήστες.

Παρόμοιες υπηρεσίες μπορεί να αποτελέσουν τη βάση για εκτενή πειράματα υλικού και λογισμικού σε πιθανούς πελάτες, χωρίς τα μειονεκτήματα της σημερινής πρακτικής όπου ο προμηθευτής πρέπει να δώσει στον πελάτη ακριβά μηχανήματα και το αντίστοιχο λογισμικό για εκτενή πειράματα στους χώρους του. Σε πολλές περιπτώσεις, μπορεί να είναι επιθυμητή η παροχή υπηρεσιών επεξεργασίας σε μη ειδικευμένους χρήστες στο Internet. Παραδείγματα θα μπορούσαν να αποτελέσουν ένα μαγαζί φωτογραφιών που παρέχει υπηρεσίες επεξεργασίας εικόνων στους πολίτες, ή μια εταιρεία που σκοπεύει να παρέχει σε χρήστες στον τομέα της διαφήμισης, έκδοσης και ψυχαγωγίας τη δυνατότητα ανάκλησης και επεξεργασίας εικόνων, όπως στο παράδειγμα [18].

Το περιβάλλον στην τωρινή του μορφή αποτελεί μια πρώτη επίδειξη της αρχιτεκτονικής που προτείνει η εργασία. Υπάρχουν αρκετές επεκτάσεις που έχουν σχεδιαστεί για υλοποίηση στο άμεσο ή απώτερο μέλλον. Οι κυριότερες απ' αυτές αναφέρονται στις επόμενες παραγράφους.

Προσθήκη Γλώσσας για Επεκτάσεις (Scripting και Plugins)

Ένα περιβάλλον επεξεργασίας δεδομένων είναι πολύ χρήσιμο να μπορεί να προσαρμόζεται δυναμικά σε χρήση δεδομένων για τα οποία δεν έχει σχεδιαστεί. Απώτερος στόχος μας είναι να μπορέσουμε να μετατρέψουμε το προτεινόμενο σύστημα σε μία πλατφόρμα πάνω στην οποία θα μπορούμε να χτίσουμε διαφορετικούς τρόπους επεξεργασίας πληροφορίας, αποτελεσματικά, με τρόπο διαφανή και εύχρηστο για τον τελικό χρήστη, χωρίς μεγάλη πολυπλοκότητα και με την καλύτερη δυνατή χρήση των διάφορων διαθέσιμων πόρων. Γι' αυτήν την περίπτωση είναι πολύ χρήσιμο να μπορεί ο κάθε χρήστης-προγραμματιστής να προσθέτει λειτουργίες στο περιβάλλον, δηλαδή να το προγραμματίσει (ως ένα βαθμό). Παραδείγματα τέτοιων περιπτώσεων είναι η εισαγωγή ενός νέου τύπου δεδομένων στο περιβάλλον (π.χ. καρδιογράφημα), το οποίο χρειάζεται διαφορετικό τρόπο αναπαράστασης στην οθόνη καθώς και διαφορετικό τρόπο ανάγνωσης και αποθήκευσης σε αρχείο. Ένα δεύτερο εξίσου σημαντικό παράδειγμα είναι η προσθήκη ενός αλγόριθμου που θα επικοινωνεί με το χρήστη μέσω ενός διαλόγου.

Η ανάγκη για μια τέτοια λειτουργικότητα οδήγησε σε ένα πρώτο σχεδιασμό και μελέτη μηχανισμών για την εύκολη επέκταση και μορφοποίηση των λειτουργιών του συστήματος. Ο προγραμματισμός των επεκτάσεων πρέπει να γίνει σε κάποια γλώσσα προγραμματισμού που να επιτρέπει τη συγγραφή αρκετά πολύπλοκων προγραμμάτων. Τα προγράμματα αυτά πρέπει να είναι γραμμένα με τρόπο που να επιτρέπει την εκτέλεση τους κάτω από διαφορετικές πλατφόρμες και θα πρέπει να μπορούν να μεταφερθούν μέσω δικτύου με εύκολο τρόπο, καθώς και θα πρέπει να μπορούν να ενσωματωθούν μέσα σε ένα πρόγραμμα γραμμένο σε κάποια άλλη γλώσσα (C, C++, Java), ή ακόμη και να παραχθούν δυναμικά [110]. Τέλος, τα προγράμματα που θα γραφτούν σε αυτή τη γλώσσα πρέπει να μπορούν να αλληλεπιδράσουν με το υπόλοιπο περιβάλλον με χρήση διαφόρων προκαθορισμένων συναρτήσεων.

Παίρνοντας υπόψη τις διάφορες τάσεις για επιλογή μιας γλώσσας προγραμματισμού για τον προγραμματισμό των επεκτάσεων μεγάλων συστημάτων, καθώς και την κατάσταση στην

οποία βρίσκεται η υλοποίηση κάθε γλώσσας, επιλέχθηκαν δύο υποψήφιες γλώσσες, η Tcl [111] και η Scheme [112] (ή κάποια διαλεκτός της). Η Tcl είναι μία γλώσσα που δίνει θαυμάσιες λύσεις για την επίλυση μερικών τύπων προβλημάτων (όπως διαμόρφωση περιβάλλοντος - configuration), απλό έλεγχο και μικρές προσθήκες σε πολύ μεγαλύτερα κομμάτια από υπάρχοντα κώδικα. Έχει σχεδιαστεί με σκοπό να μπορεί να ενσωματωθεί μέσα σε C/C++ κώδικα. Σε συνδυασμό με το Tk, επιτρέπει τη δημιουργία γραφικών διαλόγων με το χρήστη. Η Scheme είναι μία γλώσσα με πολύ μεγάλη παράδοση, αρκετές δυνατότητες (περισσότερες δυνατότητες από την Tcl), αλλά δεν έχει υλοποιηθεί (ακόμη) κάποια διαλεκτός της με τρόπο που να επιτρέπει την αποτελεσματική ενσωμάτωση της μέσα σε ένα άλλο σύστημα γραμμένο σε C/C++. Έχει αρχίσει μία προσπάθεια από τη GNU για την κατασκευή μίας γλώσσας βασισμένης στη Scheme, που να μπορεί να χρησιμοποιηθεί για τον προγραμματισμό επεκτάσεων των διαφόρων συστημάτων που κατασκευάζονται κάτω από τη GNU. Η προσπάθεια αυτή είναι πολύ σημαντική και αναμένεται να φέρει τρομερές αλλαγές στον τρόπο διασύνδεσης διαφόρων προγραμμάτων καθώς και στον τρόπο προγραμματισμού διαφόρων συστημάτων.

Συνοψίζοντας, έχει προγραμματιστεί για το άμεσο μέλλον η ανάπτυξη μηχανισμών για την εύκολη επέκταση του περιβάλλοντος. Η αρχική επιλογή γλώσσας για τον προγραμματισμό των επεκτάσεων του συστήματος είναι η Tcl, ενώ παρακολουθούμε τις εξελίξεις στο πεδίο της Scheme, ώστε κάποια στιγμή στο μέλλον να ενσωματωθεί και αυτή στο σύστημα.

Ενεργοποίηση Internet

Η δυνατότητες που δίνει το περιβάλλον για ενοποίηση και αξιοποίηση πόρων ενός οργανισμού για την παροχή υπηρεσιών επεξεργασίας σε χρήστες εκτός οργανισμού (εξωτερικούς χρήστες), θα μπορούσε να αποτελέσει μια ενδιαφέρουσα λύση για όσους θέλουν να δημιουργήσουν με εύκολο τρόπο νέες υπηρεσίες για το Internet, στον χώρο της σχετικά απαιτητικής επεξεργασίας δεδομένων. Σημαντικό πλεονέκτημα του περιβάλλοντος για μια τέτοια χρήση είναι η δυνατότητα χρέωσης που παρέχει.

Οι χρήστες αν και μπορούν να προσπελάσουν το υπάρχον περιβάλλον και τις υπηρεσίες που παρέχει από το Internet, δεν μπορούν να το προσπελάσουν μέσω ενός κοινού WWW-browser. Άμεσος στόχος είναι η μεταφορά της εφαρμογής που αλληλεπιδρά ο χρήστης με το σύστημα, σε περιβάλλον και τρόπο κατάλληλο για αλληλεπίδραση μέσω ενός κοινού WWW-browser χρησιμοποιώντας τη γλώσσα προγραμματισμού Java [91]. Επιπλέον, οι μηχανισμοί χρέωσης πρέπει να επεκταθούν ώστε να χρησιμοποιούν με ασφαλές τρόπο διάφορες μορφές ψηφιακών χρημάτων, σύμφωνα με τις γενικότερες αρχές και πρακτικές που έχουν αρχίσει να εμφανίζονται στον χώρο του ηλεκτρονικού εμπορίου [113,114].

Ακολουθίες Αλγόριθμων Επεξεργασίας

Το περιβάλλον στην τωρινή του μορφή επιτρέπει στο χρήστη να διαχειριστεί προκαθορισμένες περιγραφές ακολουθιών επεξεργασίας. Για να μπορέσει ο χρήστης να κατασκευάσει τη δικιά του ακολουθία χρειάζεται αρκετός κόπος και είναι μία διαδικασία

τρωτή σε σφάλματα. Λειτουργικότητες όπως δυνατότητα για εποπτικό έλεγχο της ακολουθίας και δυνατότητα για προγραμματισμό χωρίς ιδιαίτερη γνώση πολλών λεπτομερειών, είναι προφανή πλεονεκτήματα που παρέχονται από οπτικά περιβάλλοντα επεξεργασίας εικόνας [41,42,43,45]. Μέρος μελλοντικής υλοποίησης αποτελεί ένα οπτικό περιβάλλον προγραμματισμού ακολουθιών επεξεργασίας, αντίστοιχο με το Cantata του Khoros [44]. Το περιβάλλον πρέπει να παρέχει μία γραφική αναπαράσταση της εκτέλεσης για κάθε χρονική στιγμή και να επιτρέπει στο χρήστη να επεξεργαστεί, εφόσον το θέλει, ενδιάμεσα αποτελέσματα της επεξεργασίας.

Επίσης, είναι επιθυμητό να υπάρχουν μηχανισμοί για την εύκολη ανάκτηση αλγόριθμων, βάση προσωπικών ή άλλων κριτηρίων του χρήστη, έτσι ώστε να προτείνονται πιθανοί αλγόριθμοι για τα διάφορα βήματα της ακολουθίας. Γενικά, όταν οι αλγόριθμοι ενός συστήματος επεξεργασίας ξεπερνούν τον αριθμό των μερικών δεκάδων, ο κοινός χρήστης αντιμετωπίζει σοβαρό πρόβλημα στην επιλογή ενός αλγόριθμου κατάλληλου για μια δεδομένη μορφή επεξεργασίας συγκεκριμένης μορφής δεδομένων εισόδου. Για την διάδοση του περιβάλλοντος για χρήση από μη εξειδικευμένους χρήστες, είναι επιθυμητό να υπάρχουν μηχανισμοί για την έξυπνη περιγραφή και ανάκληση αλγόριθμων. Το συγκεκριμένο θέμα έχει αρκετό ερευνητικό ενδιαφέρον και δεν είναι εύκολο να αποτελέσει άμεσο στόχο για γρήγορη επέκταση.

Προς την Ανάπτυξη Πλαισίου Εργασίας

Το πλαίσιο εργασίας (framework) [115] είναι ένα σύνολο από προκατασκευασμένα κομμάτια λογισμικού που οι προγραμματιστές μπορούν να χρησιμοποιήσουν, επεκτείνουν ή τροποποιήσουν για να αναπτύξουν συγκεκριμένες υπολογιστικές λύσεις. Ένα πλαίσιο εργασίας εσωκλείνει την προγραμματιστική εμπειρία που χρειάζεται για να λυθεί μία συγκεκριμένη κλάση προβλημάτων. Σαν αποτέλεσμα, η κατασκευή του διαφέρει πολύ από την κατασκευή μιας αυτόνομης εφαρμογής. Ένα επιτυχές πλαίσιο εργασίας λύνει προβλήματα τα οποία επιφανειακά είναι αρκετά διαφορετικά από το πρόβλημα για το οποίο αρχικά δημιουργήθηκε. Η λύση του προβλήματος πρέπει να αποτυπωθεί με τέτοιο τρόπο που να είναι ανεξάρτητη από το αρχικό πρόβλημα και τις μελλοντικές λύσεις στις οποίες θα χρησιμοποιηθεί. Με τη χρήση διαφόρων πλαισίων εργασίας, οι κατασκευαστές λογισμικού δεν χρειάζεται να αρχίζουν από το μηδέν κάθε φορά που γράφουν μία εφαρμογή.

Η προτεινόμενη αρχιτεκτονική έχει εφαρμοστεί πρόσφατα σε ένα άλλο τομέα, για την ανάπτυξη συστήματος διαχείρισης ροής εργασίας στον χώρο της υγείας [19]. Η εμπειρία από την άμεση εφαρμογή της αρχιτεκτονικής για την ανάπτυξη δύο συστημάτων σε δύο διαφορετικούς τομείς εφαρμογών και το γεγονός ότι τα περισσότερα τμήματα τόσο του σχεδιασμού όσο και της υλοποίησης αποδείχθηκαν επαναχρησιμοποιήσιμα, δείχνει ότι το προτεινόμενο περιβάλλον έχει το δυναμικό να αποτελέσει την βάση για την ανάπτυξη ενός πλαισίου εργασίας. Σχετική μελέτη έχει ήδη ξεκινήσει και αποτελεί το αντικείμενο μακροπρόθεσμης συνέχειας της παρούσας εργασίας._

ΠΑΡΑΡΤΗΜΑ Α

Διαχείριση Πόρων: Προσομοιωτής - Πειράματα

Ένα υπολογιστικό περιβάλλον, όπως αυτό που περιγράφεται στην παρούσα εργασία, αποτελείται από διάφορες οντότητες οι οποίες αλληλεπιδρούν μεταξύ τους (πράκτορες, επεξεργαστές, δίκτυο). Οι αλληλοεπιδράσεις αυτές είναι πολλές φορές τόσο πολύπλοκες και σε μια πραγματική εφαρμογή του περιβάλλοντος δεν ακολουθούν κάποιο προκαθορισμένο μοντέλο (δυναμικό περιβάλλον), πράμα που κάνει δύσκολη (έως και αδύνατη) μια θεωρητική απόδειξη της σωστής λειτουργίας του περιβάλλοντος και της καλής συμπεριφοράς του μηχανισμού ανάθεσης πόρων. Ο έλεγχος για τη σωστή λειτουργία του συνήθως γίνεται διεξάγοντας ένα πλήθος πειραμάτων σε πραγματικές συνθήκες, αλλά και σε αυτή την περίπτωση, η ορθότητα δεν αποδεικνύεται πλήρως, μια που είναι σχετικά δύσκολη (και ακριβή) η εκτέλεση εκτενών πειραμάτων σε ένα πραγματικό κατανεμημένο σύστημα με πλήρως ελεγχόμενο τρόπο.

Για τον έλεγχο της συμπεριφοράς του προτεινόμενου μηχανισμού για την ανάθεση υπολογιστικών πόρων αναπτύχθηκε ένας προσομοιωτής. Η προσομοίωση του περιβάλλοντος επιτρέπει τον έλεγχο ιδεών και πιθανών επιλογών, πριν αυτές περάσουν στο τελικό σύστημα, μέσω ενός μοντέλου του συστήματος. Επιπλέον επιτρέπει την διενέργεια πειραμάτων με χαμηλό κόστος ώστε να ελεγχθεί η λειτουργία του συστήματος σε διάφορες, ακραίες συνήθως, συνθήκες. Στο σχεδιασμό μιας προσομοίωσης, η πιο βασική παράμετρος είναι το τι ακριβώς χρειάζεται να μετρηθεί με την συγκεκριμένη προσομοίωση, ποια είναι τα μεγέθη των οποίων μας ενδιαφέρει η μέτρηση, καθώς και τα χαρακτηριστικά του συστήματος που τα επηρεάζουν. Πολύ βασικός επίσης είναι ο σχεδιασμός του ίδιου του πειράματος καθώς και των μηχανισμών συλλογής των αποτελεσμάτων. Σε κάθε βήμα του σχεδιασμού, γίνονται κάποιες απαραίτητες απλουστεύσεις ώστε να βοηθηθεί η υλοποίηση και η μελέτη του συστήματος μέσω της προσομοίωσης. Παρόλα αυτά, η ακρίβεια των αποτελεσμάτων εξαρτάται από την ισχύ των αρχικών θεωρήσεων και απλουστεύσεων.

Η υλοποίηση του συγκεκριμένου προσομοιωτή βασίστηκε στο C++Sim [117], μία οντοκεντρική βιβλιοθήκη που χρησιμοποιεί τη γλώσσα προγραμματισμού C++. Η βιβλιοθήκη παρέχει τους βασικούς μηχανισμούς για την κατασκευή ενός προσομοιωτή, όπως είναι οι διάφορες έτοιμες οντότητες για τη δημιουργία διεργασιών, για τον προγραμματισμό των διεργασιών, καθώς και οι διάφορες γεννήτριες κατανομών και τυχαίων αριθμών. Οι οντότητες αυτές έχουν μία προκαθορισμένη συμπεριφορά, που μπορεί να τροποποιηθεί και επεκταθεί χρησιμοποιώντας τη δυνατότητα της κληρονόμησης που παρέχει η γλώσσα προγραμματισμού C++. Στη συνέχεια περιγράφεται το μοντέλο του συστήματος που χρησιμοποιήθηκε καθώς και οι διάφορες οντότητες που το αποτελούν, τα μετρούμενα μεγέθη, και τα πειράματα που έγιναν καθώς και τα αποτελέσματα τους.

Α.1 Μοντέλο συστήματος

Το σύστημα αποτελείται από τις εξής οντότητες που αλληλεπιδρούν δυναμικά μεταξύ τους και πρέπει να μοντελοποιηθούν για την ανάπτυξη του προσομοιωτή:

- δίκτυο,
- επεξεργαστές,
- χρήστες,
- πράκτορας διαχειριστής,
- πράκτορες εκτέλεσης,
- αλγόριθμοι.

Δίκτυο

Για την προσομοίωση της επικοινωνίας μεταξύ των διαφόρων διεργασιών που εκτελούνται στα διάφορα μηχανήματα του συστήματος χρειάζεται μια μοντελοποίηση του δικτύου που ενώνει αυτά τα μηχανήματα. Επιλέξαμε τη χρήση ενός πολύ απλού μοντέλου δικτύου τύπου bus. Η μετάδοση στο δίκτυο μοντελοποιείται χρησιμοποιώντας μία ουρά στην οποία εισέρχονται τα μηνύματα προς μετάδοση, περιμένουν κάποιο χρονικό διάστημα και παραδίνονται στον παραλήπτη τους. Η καθυστέρηση της μετάδοσης ενός μηνύματος εξαρτάται μόνο από την ταχύτητα του δικτύου (η οποία ορίζεται στην αρχή της προσομοίωσης). Μία επιπλέον καθυστέρηση στη μετάδοση προστίθεται σε κάθε μήνυμα εφόσον τη στιγμή που πρόκειται να αρχίσει η μετάδοση του, υπάρχουν και άλλα μηνύματα που αναμένουν μετάδοση. Η καθυστέρηση αυτή έχει σκοπό να μοντελοποιήσει την αντίστοιχη καθυστέρηση που εμφανίζεται σε δίκτυα τύπου bus (π.χ. Ethernet) όταν πολλοί πομποί (κάρτες δικτύου) προσπαθούν να στείλουν ταυτόχρονα ένα μήνυμα.

Επεξεργαστές

Το σύστημα αποτελείται από ένα σύνολο υπολογιστών οι οποίοι έχουν τη δυνατότητα να επικοινωνήσουν μεταξύ τους μέσω ενός δικτύου. Στην περίπτωση του συστήματος μας, για τον κάθε υπολογιστή μας ενδιαφέρει η μοντελοποίηση του επεξεργαστή του, ώστε να μπορούμε να μετρήσουμε τη χρήση καθώς και την απόδοση του.

Ο κάθε επεξεργαστής έχει ένα σύνολο διεργασιών προς εκτέλεση, όπως οι διεργασίες που αντιστοιχούν στους πράκτορες του περιβάλλοντος καθώς και οι διεργασίες που αφορούν στην εκτέλεση των αλγόριθμων. Σε κάθε χρονική στιγμή μόνο μία διεργασία εκτελείται. Για την επίτευξη πολυεπεξεργασίας, σε κάθε διεργασία ανατίθεται ένα μικρό χρονικό διάστημα μέσα στο οποίο μπορεί να εκτελεστεί. Ο επεξεργαστής χρησιμοποιεί ένα προγραμματιστή διεργασιών τύπου round robin [118], όπου όλες οι διεργασίες (αλγόριθμοι) έχουν την ίδια προτεραιότητα εκτός από το διαχειριστή και τον πράκτορα εκτέλεσης που

έχουν μεγαλύτερη. Το αποτέλεσμα είναι η επίτευξη «ψευδο-παράλληλης» επεξεργασίας των διαφόρων διεργασιών με τρόπο αντίστοιχο με αυτό που συμβαίνει στα σύγχρονα λειτουργικά συστήματα. Πρέπει να αναφερθεί ότι για κάθε διεργασία μοντελοποιείται μόνο ο χρόνος που χρειάζεται η εκτέλεση της και όχι πιθανές καθυστερήσεις λόγω εισόδου και εξόδου δεδομένων.

Σε ένα κατανεμημένο σύστημα πολλών υπολογιστών, ο κάθε υπολογιστής έχει διαφορετικές δυνατότητες και ταχύτητες επεξεργασίας. Κάθε επεξεργαστής μπορεί να εκτελέσει τον ίδιο αλγόριθμο με διαφορετικό συνολικό χρόνο εκτέλεσης. Αυτό μπορεί να οφείλεται σε χαρακτηριστικά του επεξεργαστή (τύπος, ταχύτητα ρολογιού) καθώς και σε πιθανό φόρτο λόγω εκτέλεσης διεργασιών ανεξάρτητων από το συγκεκριμένο περιβάλλον. Η μοντελοποίηση της σχετικής ικανότητας κάθε μηχανήματος γίνεται με τη χρήση ενός δείκτη (*speed index*) που δείχνει πόσες φορές πιο γρήγορα (ή πιο αργά) μπορεί να εκτελέσει ένας επεξεργαστής μία διεργασία, από κάποιο μηχάνημα το οποίο θεωρούμε ότι έχει δείκτη ίσο με τη μονάδα. Κάθε διεργασία κανονικοποιείται με βάση τον συγκεκριμένο δείκτη και έτσι υπολογίζεται ο χρόνος που χρειάζεται για την εκτέλεση της σε κάθε επεξεργαστή.

Χρήστες

Οι χρήστες αποτελούν ένα εξωτερικό παράγοντα για το σύστημα και στην πραγματικότητα είναι η μοναδική μη υπολογιστική οντότητα που θέλουμε να προσομοιώσουμε. Η μοντελοποίηση του τρόπου αλληλοεπίδρασης του κάθε χρήστη με το σύστημα είναι πολύ δύσκολη μέχρι και αδύνατη. Αυτό συμβαίνει γιατί η σωστή μοντελοποίηση ενός χρήστη προϋποθέτει τη μοντελοποίηση των νοητικών διεργασιών του, της σκέψης του, των εμπειριών του, της ψυχολογικής και σωματικής του κατάστασης, σε ένα καλά καθορισμένο χώρο εργασίας και όταν η αποστολή του είναι καθορισμένη και μοντελοποιήσιμη. Ωστόσο, αυτό δεν συμβαίνει στη συγκεκριμένη περίπτωση, μια και το προτεινόμενο περιβάλλον αφορά στην υποστήριξη υπηρεσιών παροχής επεξεργασίας εικόνας, ένα τομέα με πολλές διαφορετικές εφαρμογές και μεγάλο εύρος σε ανάγκες και συνήθειες χρηστών. Σαν αποτέλεσμα διάφορες παραδοχές και απλουστεύσεις είναι απαραίτητο να γίνουν, ώστε να υπάρχει κάποιο ικανοποιητικό μοντέλο των χρηστών που να αντιπροσωπεύει τον ευρύτερο χώρο.

Έτσι, για τα πειράματα θεωρήσαμε ότι οι αλγόριθμοι που χρησιμοποιούν οι χρήστες ανήκουν σε τέσσερις κατηγορίες, ανάλογα με το χρόνο που χρειάζονται για την εκτέλεση τους:

- 0.1 - 1 min (πολύ γρήγοροι)
- 1 - 5 min (γρήγοροι)
- 5 - 10 min (μέτριοι)
- 10 - 30 min (αργοί)

Τα παραπάνω χρονικά διαστήματα επιλέχθηκαν γιατί αντικατοπτρίζουν σε ένα ικανοποιητικό βαθμό την σημερινή πραγματικότητα στον χώρο και κατά κάποιο τρόπο αντιστοιχούν στην σημερινή ψυχολογική κλίμακα (από 'πολύ γρήγορο' έως 'αργό') για περιπτώσεις εκτέλεσης αλγόριθμων επεξεργασίας εικόνας. Υποθέτουμε ότι στην γενική περίπτωση η υποδομή ενός οργανισμού σε υπολογιστική ισχύ είναι ανάλογη με τις ανάγκες των χρηστών του και τις απαιτήσεις των αλγόριθμων που αυτοί χρησιμοποιούν, ώστε σε γενικές γραμμές να ικανοποιείται η παραπάνω ψυχολογική κλίμακα για την διάρκεια της εκτέλεσης.

Επιπλέον, θεωρήσαμε τα μεγέθη της εισόδου και της εξόδου του κάθε αλγόριθμου, να ανήκουν ανάλογα σε τρεις κατηγορίες:

- 256K – 1 M (μικρός όγκος δεδομένων),
- 1 – 4 M (μέτριος όγκος δεδομένων)
- 4 – 8 M (μεγάλος όγκος δεδομένων)

Ο χρήστης μοντελοποιείται χρησιμοποιώντας μία διεργασία η οποία είναι αδρανής (δηλαδή ο χρήστης σκέφτεται ή ασχολείται με άλλες εργασίες) για ένα ορισμένο χρονικό διάστημα και μόλις ενεργοποιηθεί (τελειώσει τη σκέψη του ή άλλες εργασίες και αποφασίσει την εκτέλεση αλγόριθμου) στέλνει μία αίτηση για την εκτέλεση στον διαχειριστή και στη συνέχεια αδρανοποιείται και πάλι.

Διαχειριστής - Πράκτορες Εκτέλεσης

Ο διαχειριστής αναλαμβάνει την επεξεργασία των αιτήσεων των χρηστών, την διεξαγωγή της δημοπρασίας καθώς και την ανάθεση της εκτέλεσης μιας διεργασίας στον κατάλληλο πράκτορα εκτέλεσης, όπως ακριβώς έχει περιγραφεί στην αρχιτεκτονική του συστήματος, στο τρίτο και τέταρτο κεφάλαιο. Για την ανάθεση μίας διεργασίας χρησιμοποιεί το κριτήριο του ελάχιστου αναμενόμενου χρόνου εκτέλεσης της διεργασίας.

Ο πράκτορας εκτέλεσης συμμετέχει στις δημοπρασίες που διεξάγει ο διαχειριστής. Δέχεται την αίτηση για προσφορά που του στέλνει ο διαχειριστής, υπολογίζει τον υπολειπόμενο χρόνο που χρειάζονται οι διεργασίες που εκτελούνται στον επεξεργαστή του, συνυπολογίζει το χρόνο που χρειάζεται ο αιτούμενος αλγόριθμος και στέλνει την προσφορά του στον διαχειριστή. Ακολουθεί μία πολύ απλή πολιτική όσο αναφορά την συμμετοχή σε δημοπρασίες για την εκτέλεση ενός αλγόριθμου, δηλαδή στέλνει απάντηση για κάθε αίτηση για εκτέλεση που δέχεται, ανεξαρτήτως αν η ανάληψη της εκτέλεσης της διεργασίας θα είχε σαν αποτέλεσμα την καθυστέρηση διεργασιών που εκτελούνται. Στα πλαίσια της προσομοίωσης ο πράκτορας εκτέλεσης δεν έχει τοπική κρυφή μνήμη για την προσωρινή αποθήκευση δεδομένων.

Αλγόριθμοι

Κάθε αλγόριθμος μοντελοποιείται με μία διεργασία η οποία χρειάζεται ένα χρονικό διάστημα για την εκτέλεσή της και επιπλέον αναμένει ένα μέγεθος δεδομένων για είσοδο και παράγει ένα μέγεθος δεδομένων για έξοδο. Ο κάθε επεξεργαστής αναλαμβάνει την εκτέλεση των διαφόρων αλγόριθμων που του έχουν ανατεθεί για εκτέλεση. Η εκτέλεση κάθε αλγόριθμου προσομοιώνεται έχοντας τον επεξεργαστή σε αδράνεια για ένα μικρό χρονικό διάστημα κάθε φορά, το οποίο αφαιρείται από το συνολικό χρόνο εκτέλεσης του κάθε αλγόριθμου. Ο χρόνος που χρειάζεται για την εκτέλεση του ενός αλγόριθμου, καθώς και το μέγεθος των δεδομένων εισόδου και εξόδου του, θεωρούνται γνωστά κατά την αποστολή της αίτησης για την εκτέλεσή του.

Α.2 Μετρούμενα Μεγέθη

Κατά τη διάρκεια εκτέλεσης μιας προσομοίωσης πολλά διαφορετικά μεγέθη μπορούν να μετρηθούν, είτε σε αναλυτική μορφή για περαιτέρω επεξεργασία είτε σε συγκεντρωτική μορφή. Ο προσομοιωτής που κατασκευάστηκε επιτρέπει την απόκτηση αναλυτικών αποτελεσμάτων για μία πληθώρα γεγονότων, όπως: αίτηση για εκτέλεση αλγόριθμου από χρήστη, δημιουργία διεργασίας, ανάθεση διεργασίας σε επεξεργαστή, τερματισμός διεργασίας, αποστολή μηνύματος, έναρξη δημοπρασίας, μηνύματα προσφορών, τέλος δημοπρασίας, μήνυμα ακύρωσης προσφοράς. Τα γεγονότα αυτά καταγράφονται σε ένα αρχείο με αναλυτική πληροφορία για κάθε γεγονός, με τη χρονική σειρά που συμβαίνουν. Το αρχείο αυτό μπορεί πολύ εύκολα να εισαχθεί σε μία βάση δεδομένων όπως η Microsoft Access ώστε να γίνει κάποια πρώτης μορφής επεξεργασία και να χρησιμοποιηθεί ένα στατιστικό πακέτο, όπως το Microsoft Excel, για εξαγωγή γραφημάτων και άλλων στατιστικών στοιχείων. Τέλος, υπάρχει η δυνατότητα καταγραφής σε αρχείο των ενεργειών (αιτήσεις για εκτέλεση αλγόριθμου) που έκαναν οι χρήστες, ώστε να χρησιμοποιηθούν σαν είσοδο σε επόμενο πείραμα, στο οποίο θέλουμε να μελετήσουμε μία διαφορετική πολιτική από την πλευρά του συστήματος με την ίδια πάντα αντιμετώπιση από τους χρήστες. Αυτή η δυνατότητα είναι χρήσιμη για την πειραματική επιλογή κατάλληλης πολιτικής των πρακτόρων στην περίπτωση που είναι δεδομένος ένας οργανισμός με καθορισμένο το μοντέλο χρηστών και συγκεκριμένη άποψη για την ποιότητα υπηρεσιών.

Ο προσομοιωτής μπορεί να αναθέσει την εκτέλεση κάθε αλγόριθμου με τους παρακάτω τρόπους:

- τοπική ανάθεση: κάθε αλγόριθμος εκτελείται τοπικά στο μηχάνημα του χρήστη που ζήτησε την εκτέλεση του
- τυχαία ανάθεση: κάθε αλγόριθμος εκτελείται τυχαία σε κάποιο μηχάνημα που ανήκει στο σύστημα
- με μηχανισμό δημοπρασίας: κάθε αλγόριθμος εκτελείται βάση του μηχανισμού δημοπρασίας που έχει περιγραφεί στο τέταρτο κεφάλαιο.

Τα μεγέθη που ενδιαφέρουν στα συγκεκριμένα πειράματα είναι ο αριθμός των διεργασιών που εκτελέστηκαν ανά επεξεργαστή, ο μέσος χρόνος απόκρισης του κάθε επεξεργαστή, ο ενεργός χρόνος του κάθε επεξεργαστή, ο μέσος πραγματικός χρόνος εκτέλεσης κάθε αλγόριθμου σε σχέση τον μέσο αναμενόμενο χρόνο εκτέλεσης, η τυπική απόκλιση του πραγματικού χρόνου εκτέλεσης κάθε αλγόριθμου, καθώς και ο όγκος των δεδομένων που μετακινήθηκαν στο δίκτυο. Για μεγαλύτερη ευκολία στην συγκριτική επισκόπηση των αποτελεσμάτων, ο προσομοιωτής υπολογίζει τα παραπάνω αποτελέσματα συγκεντρωτικά, χωρίς να χρειάζεται για την εξαγωγή τους η επεξεργασία των επιμέρους καταγραμμένων γεγονότων.

Α.3 Πειράματα

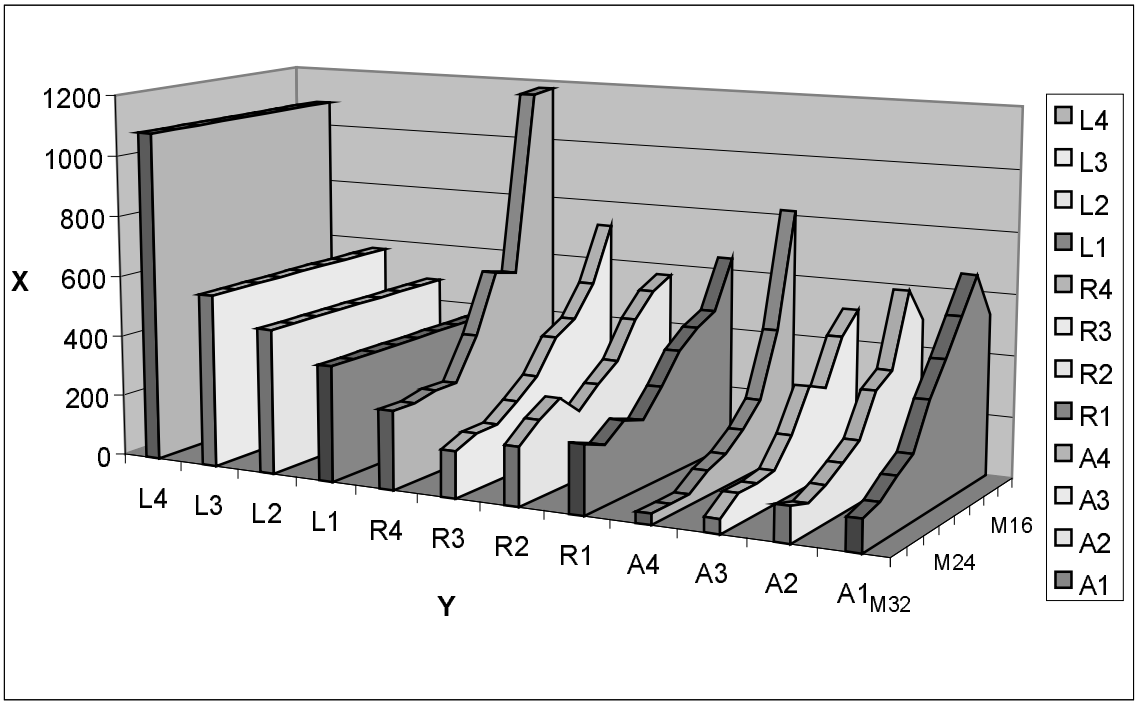
Η έλλειψη ενός ικανοποιητικού μοντέλου των χρηστών του συστήματος, ώστε να χρησιμοποιηθεί στον σχεδιασμό των πειραμάτων, είχε σαν αποτέλεσμα την θεώρηση διαφορετικών υποθετικών περιπτώσεων χρήσης του συστήματος. Καταρχήν το σύστημα δοκιμάστηκε σε διάφορες ακραίες συνθήκες όσο αναφορά στο συνολικό φόρτο εργασίας, και τέλος δοκιμάστηκε σε μία περίπτωση ενός υποθετικού οργανισμού.

1ο Πείραμα

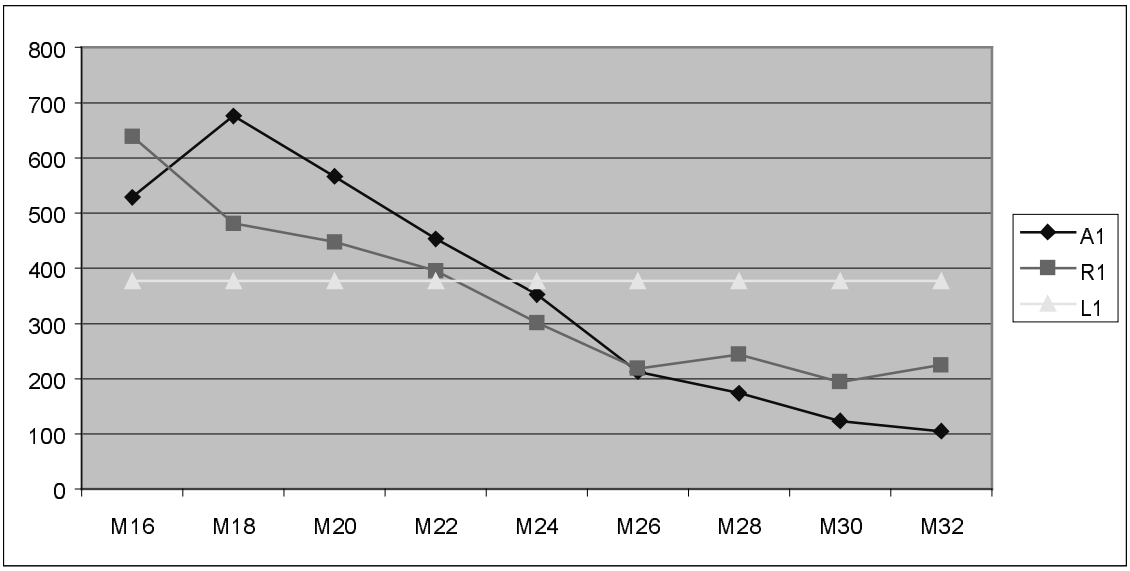
Ένας ικανοποιητικός αριθμός χρηστών (16) συμμετέχει στο πείραμα, αρχικά χρησιμοποιώντας ισάριθμο αριθμό μηχανημάτων (16) που σταδιακά αυξάνεται (μέχρι 32). Η επιλογή των 16 χρηστών και των 16 μηχανημάτων έγινε με σκοπό τη μοντελοποίηση μιας συνηθισμένης ομάδας χρηστών και μηχανημάτων που χρησιμοποιούν το ίδιο τοπικό δίκτυο. Επίσης, τα μηχανήματα αυξάνονται σταδιακά σε σχέση με τους χρήστες για να μελετηθεί η απόκριση του συστήματος σε περίπτωση που υπάρχουν κάποια ελεύθερα μηχανήματα στο δίκτυο, κάτι που είναι αρκετά συχνό σε έναν μεγάλο οργανισμό. Οι χρήστες εμφανίζουν το ίδιο προφίλ όσο αναφορά στις εκτελέσεις αλγόριθμων. Συγκεκριμένα, όλοι οι χρήστες εκτελούν αλγόριθμους από όλες τις κατηγορίες (πολύ γρήγορους, μέχρι και αργούς) με την ίδια πιθανότητα. Ο κάθε χρήστης ζητά την εκτέλεση ενός αλγόριθμου με συχνότητα κάθε πέντε λεπτά. Ο συνολικός αριθμός αλγόριθμων που χρησιμοποιούνται για τις μετρήσεις είναι 300 και οι μετρήσεις αρχίζουν αφού το σύστημα φορτωθεί ικανοποιητικά (έχει ζητηθεί και δρομολογηθεί η εκτέλεση από τουλάχιστον 100 αλγόριθμους). Οι χρήστες παράγουν συνεχώς αιτήσεις για εκτέλεση αλγόριθμων μέχρι να τερματίσουν οι αλγόριθμοι που έχουν επιλεγεί προς μέτρηση. Το πείραμα επαναλαμβάνεται για μέτριο και μεγάλο όγκο δεδομένων χρησιμοποιώντας ισάριθμο αριθμό μηχανημάτων με τους χρήστες (16).

Λόγω της ιδιαιτερότητας του συγκεκριμένου πειράματος, το αποτέλεσμα που ενδιαφέρει είναι η επί τις εκατό αύξηση στο χρόνο εκτέλεσης κάθε αλγόριθμου, για την κάθε κατηγορία αλγόριθμων όσο αναφορά τον απαιτούμενο χρόνο εκτέλεσης. Στα διαγράμματα που ακολουθούν χρησιμοποιείται η παραπάνω μέτρηση για την παρουσίαση των αποτελεσμάτων.

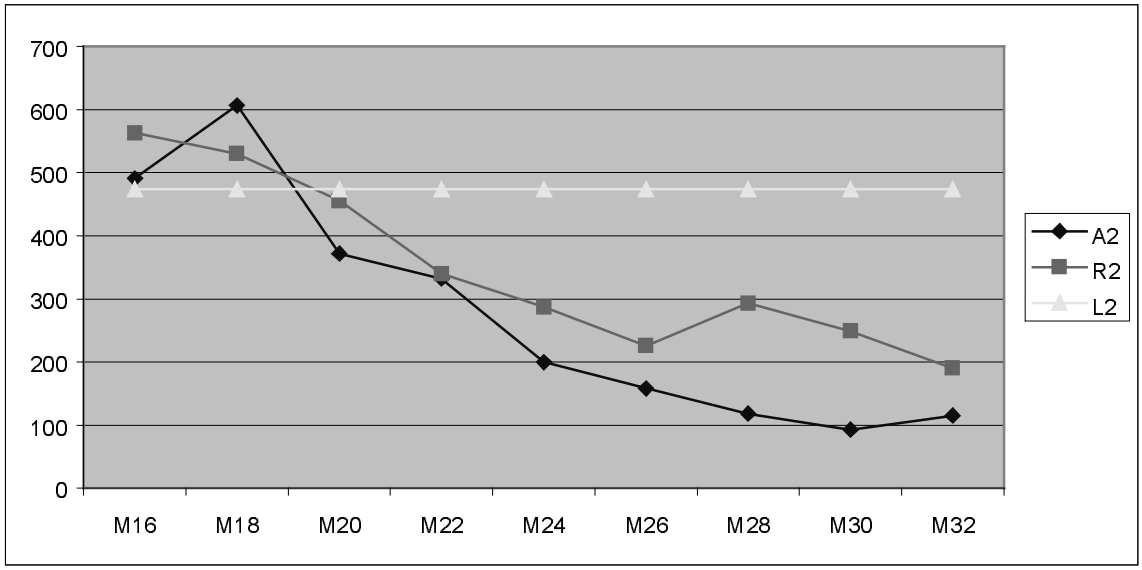
Συγκεκριμένα, στο ραβδοδιάγραμμα που ακολουθεί παρουσιάζεται η μέση % αύξηση στο χρόνο εκτέλεσης των αλγόριθμων για την κάθε κατηγορία χρόνου εκτέλεσης, με τους τρείς διαφορετικούς μηχανισμούς ανάθεσης εκτέλεσης, για κάθε κατηγορία αλγόριθμων και ως συνάρτηση του αριθμού των μηχανημάτων στο δίκτυο. Η κάθε ράβδος φέρει ένδειξη δύο χαρακτήρων. Ο πρώτος χαρακτήρας υποδεικνύει τον μηχανισμό ανάθεσης εκτέλεσης, L(=τοπική εκτέλεση), R(= τυχαία ανάθεση), A(=ανάθεση με δημοπρασία). Ο δεύτερος χαρακτήρας υποδεικνύει το είδος αλγόριθμου ως προς τον απαιτούμενο χρόνο εκτέλεσης, 1(=πολύ γρήγορος), 2(=γρήγορος), 3(=μέτριος), 4(=αργός). Δηλαδή, η ένδειξη A3 σημαίνει μέτριος αλγόριθμος και μηχανισμός ανάθεσης εκτέλεσης με δημοπρασία. Τα αποτελέσματα του πειράματος παρουσιάζονται συγκεντρωτικά στο ραβδοδιάγραμμα του Διαγράμματος 1, καθώς επίσης και αναλυτικότερα στα Διαγράμματα 2-5, όπου δίνονται τα αποτελέσματα για κάθε κατηγορία αλγόριθμων χωριστά.



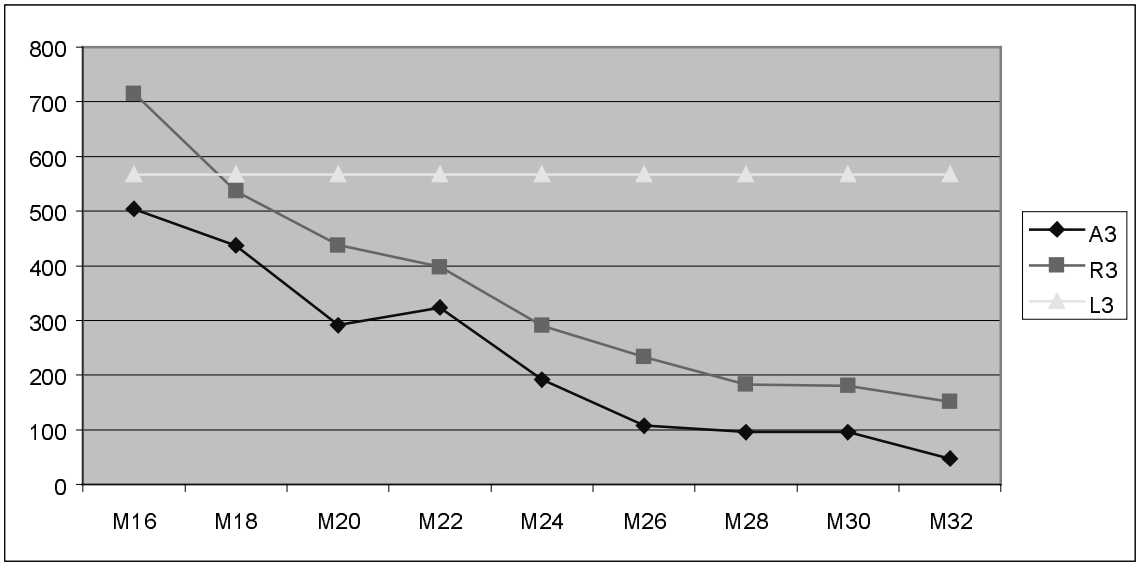
Διάγραμμα 1: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για κάθε κατηγορία αλγόριθμων ως προς την ταχύτητα εκτέλεσης τους και για διάφορους μηχανισμούς ανάθεσης εκτέλεσης αλγόριθμων, όπως περιγράφεται στο 1ο πείραμα. Ο κάθετος άξονας (X) αντιστοιχεί στις τιμές μέσης % αύξησης στο χρόνο εκτέλεσης. Κάθε μία από τις τρισδιάστατες ράβδους αντιστοιχεί σε ένα είδος αλγόριθμου και ένα μηχανισμό ανάθεσης εκτέλεσης. Συγκεκριμένα, οι ράβδοι L1-L4 αντιστοιχούν σε τοπική εκτέλεση, οι R1-R4 σε τυχαία κατανομή, και οι A1-A4 σε ανάθεση με δημοπρασία. Τέλος, για κάθε ράβδο ο αριθμός των μηχανημάτων αυξάνεται σταδιακά από 16 (στο βάθος) μέχρι 32 (μπροστά). Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M).



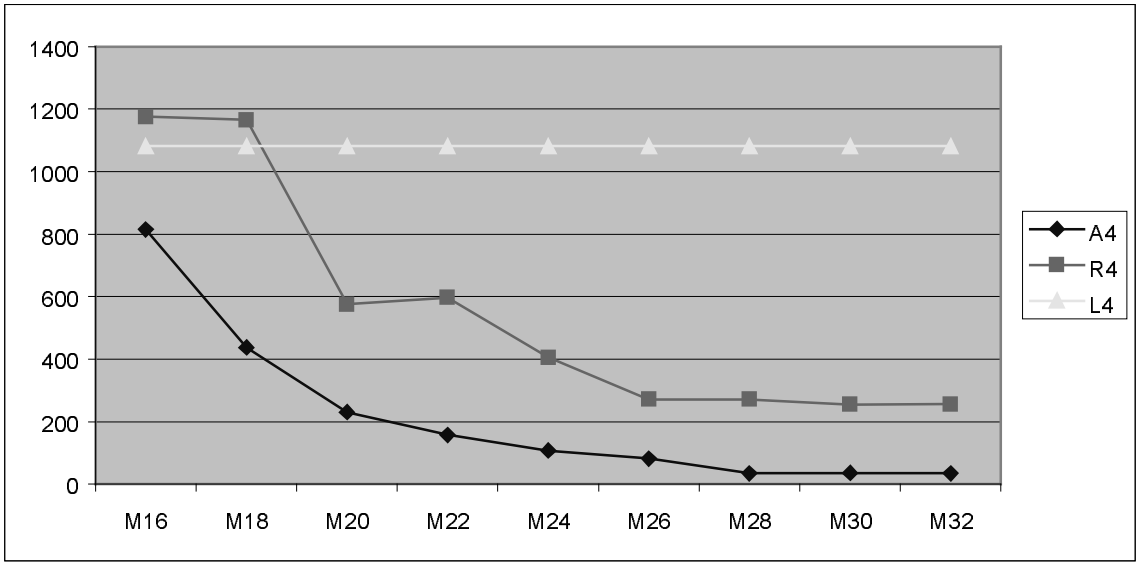
Διάγραμμα 2: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για τοπική εκτέλεση (L1), για τυχαία ανάθεση (R1) και για ανάθεση με δημοπρασία (A1), για πολύ γρήγορους αλγόριθμους (αναμενόμενος χρόνος εκτέλεσης 0.1-1 min). Ο κάθετος άξονας (Y) αντιστοιχεί στις τιμές μέσης % αύξησης του χρόνου εκτέλεσης, ενώ στον οριζόντιο άξονα παρουσιάζεται ο αριθμός των μηχανημάτων στο δίκτυο. Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M).



Διάγραμμα 3: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για τοπική εκτέλεση (L2), για τυχαία ανάθεση (R2) και για ανάθεση με δημοπρασία (A2), για γρήγορους αλγόριθμους (αναμενόμενος χρόνος εκτέλεσης 1-5 min). Ο κάθετος άξονας (Y) αντιστοιχεί στις τιμές μέσης % αύξησης του χρόνου εκτέλεσης, ενώ στον οριζόντιο άξονα παρουσιάζεται ο αριθμός των μηχανημάτων στο δίκτυο. Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M).



Διάγραμμα 4: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για τοπική εκτέλεση (L3), για τυχαία ανάθεση (R3) και για ανάθεση με δημοπρασία (A3), για μέρτιους αλγόριθμους (αναμενόμενος χρόνος εκτέλεσης 5-10 min). Ο κάθετος άξονας (Y) αντιστοιχεί στις τιμές μέσης % αύξησης του χρόνου εκτέλεσης, ενώ στον οριζόντιο άξονα παρουσιάζεται ο αριθμός των μηχανημάτων στο δίκτυο. Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M).



Διάγραμμα 5 : Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για τοπική εκτέλεση (L4), για τυχαία ανάθεση (R4) και για ανάθεση με δημοπρασία (A4), για αργούς αλγόριθμους (αναμενόμενος χρόνος εκτέλεσης 10-30 min). Ο κάθετος άξονας (Y) αντιστοιχεί στις τιμές μέσης % αύξησης του χρόνου εκτέλεσης, ενώ στον οριζόντιο άξονα παρουσιάζεται ο αριθμός των μηχανημάτων στο δίκτυο. Όλες οι εκτελέσεις υποθέτουν μικρό όγκο δεδομένων εισόδου και εξόδου (256K-1M).

Παρατηρούμε τα εξής:

- Οι χρόνοι εκτέλεσης ενός αλγόριθμου, χρησιμοποιώντας την πολιτική της τοπικής ανάθεσης, όπως άλλωστε είναι φυσικό, δεν εξαρτώνται από τον αριθμό των μηχανημάτων.

- Για αργούς και μέτριους σε ταχύτητα αλγόριθμους, η πολιτική της δημοπρασίας είναι πολύ καλύτερη από τις άλλες δύο πολιτικές, ανεξάρτητα από τον αριθμό των μηχανημάτων που συμμετέχουν.

- Για τους πολύ γρήγορους και γρήγορους αλγόριθμους, σε ένα φορτωμένο σύστημα όπως αυτό του πειράματος, η καθυστέρηση λόγω της μεταφοράς στο δίκτυο για απομακρυσμένη εκτέλεση είναι σημαντική. Γενικά παρατηρούμε ότι η εκτέλεση πολύ γρήγορων αλγόριθμων καλυτερεύει για περισσότερα από είκοσι τέσσερα μηχανήματα, ενώ για τους γρήγορους αλγόριθμους για περισσότερα από είκοσι μηχανήματα.

- Η καθυστέρηση αυτή είναι μεγάλη αναλογικά με τον απαιτούμενο χρόνο εκτέλεσης για πολύ γρήγορους και γρήγορους αλγόριθμους, ενώ ελαττώνεται για πιο αργούς αλγόριθμους. Αυτό ακριβώς φαίνεται και στο διάγραμμα 1, για την περίπτωση των 32 μηχανημάτων, συγκρίνοντας τους αλγόριθμους A1 – A4. Οι πιο αργοί αλγόριθμοι εμφανίζουν αναλογικά πιο μικρή επί τις εκατό επιβράδυνση.

- Για την πολιτική δημοπρασίας, όταν μεταβάλλεται ο αριθμός των μηχανημάτων από δέκα έξι σε δέκα οκτώ, για τους πολύ γρήγορους και γρήγορους αλγόριθμους, εμφανίζεται μία μεγαλύτερη καθυστέρηση. Αυτό οφείλεται στο γεγονός ότι αυξάνεται ο αριθμός των μηνυμάτων στο δίκτυο, με αποτέλεσμα μεγαλύτερες καθυστερήσεις λόγω δικτύου, χωρίς να είναι πολύ σημαντική η μεταβολή στον αριθμό των μηχανημάτων ώστε να επέλθει ισοστάθμιση της καθυστέρησης λόγω μεγαλύτερης παράλληλης επεξεργασίας διαφορετικών αλγόριθμων.

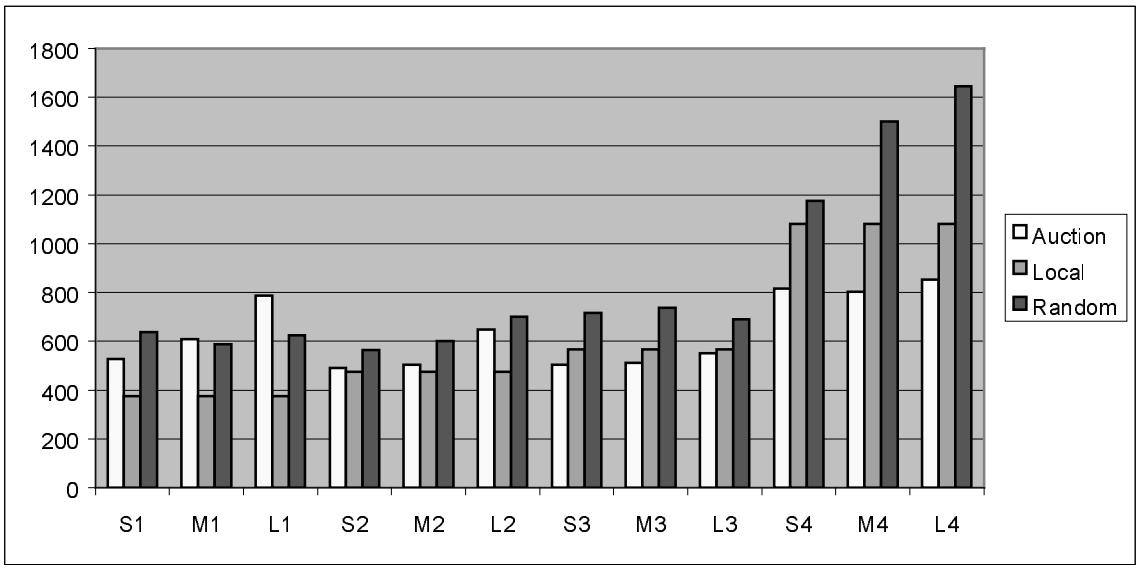
Το παραπάνω πείραμα επαναλήφθηκε με τις ίδιες παραμέτρους αλλά μόνο για την περίπτωση 16 χρηστών σε ισάριθμα μηχανήματα, για μέτριο και μεγάλο όγκο δεδομένων. Τα αποτελέσματα φαίνονται στα παρακάτω ραβδοδιαγράμματα. Κάθε ράβδος φέρει μια ένδειξη που αποτελείται από δύο χαρακτήρες. Ο πρώτος είναι S(=μικρός), M(=μέτριος), L(=μεγάλος) ανάλογα με τον όγκο δεδομένων κάθε φορά, και ο δεύτερος είναι 1(=πολύ γρήγορος), 2(=γρήγορος), 3(=μέτριος), 4(=αργός) ανάλογα με το απαιτούμενο χρόνο εκτέλεσης του κάθε αλγόριθμου. Έτσι η ένδειξη M3 σημαίνει μέτριος όγκος δεδομένων με αλγόριθμο μέτριου χρόνου εκτέλεσης.

Παρατηρούμε τα εξής:

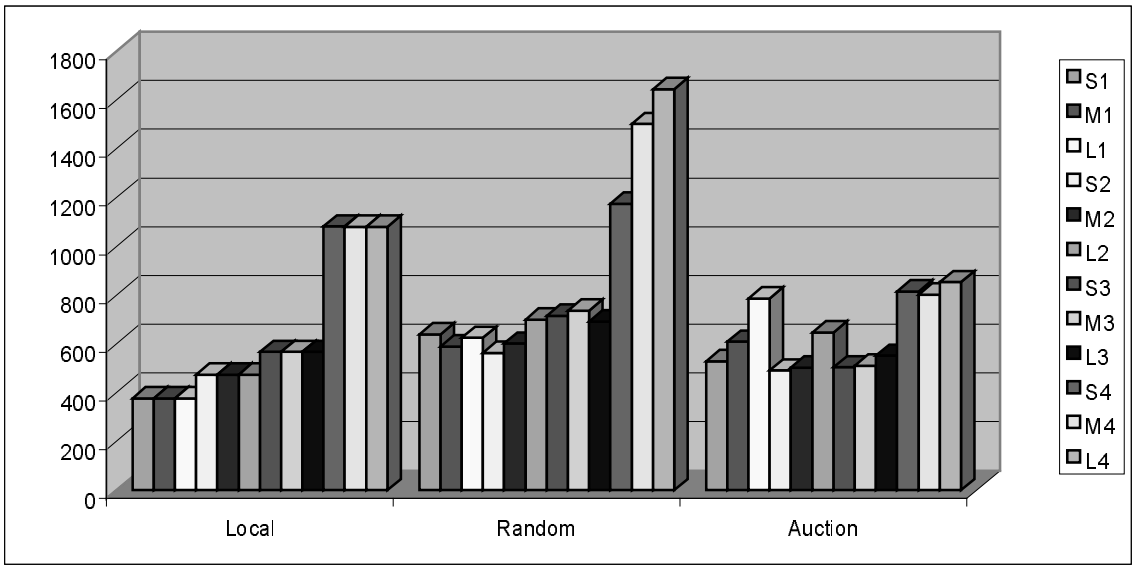
- Για τους πολύ γρήγορους και γρήγορους αλγόριθμους η εκτέλεση τους τοπικά είναι καλύτερη από την εφαρμογή των άλλων μηχανισμών, και η διαφορά γίνεται μεγαλύτερη όσο ο όγκος των δεδομένων μεγαλώνει.

- Για αλγόριθμους μέσου και μεγάλου χρόνου εκτέλεσης ο μηχανισμός της δημοπρασίας δίνει πολύ καλύτερα αποτελέσματα από τις άλλες δύο, ανεξάρτητα από τον όγκο δεδομένων.

- Η τυχαία ανάθεση δίνει άσχημα αποτελέσματα σε σχέση με τους άλλους μηχανισμούς, και χειροτερεύει όσο ο όγκος των δεδομένων μεγαλώνει.



Διάγραμμα 6: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για κάθε κατηγορία αλγόριθμων ως προς την ταχύτητα εκτέλεσης τους, για τους 3 διαφορετικούς μηχανισμούς ανάθεσης εκτέλεσης, με διαφορετικό όγκο δεδομένων εισόδου και εξόδου [μικρός (S), μέτριος (M), μεγάλος (L)] και διαφορετικές χρονικές απαιτήσεις [πολύ γρήγορος (1), γρήγορος (2), μέτριος (3), αργός (4)].



Διάγραμμα 7: Μέση % αύξηση χρόνου εκτέλεσης αλγόριθμων για κάθε κατηγορία αλγόριθμων ως προς την ταχύτητα εκτέλεσης τους, για τους 3 διαφορετικούς μηχανισμούς ανάθεσης εκτέλεσης, με διαφορετικό όγκο δεδομένων εισόδου και εξόδου [μικρός (S), μέτριος (M), μεγάλος (L)] και διαφορετικές χρονικές απαιτήσεις [πολύ γρήγορος (1), γρήγορος (2), μέτριος (3), αργός (4)].

2ο Πείραμα

Θεωρούμε ένα ικανοποιητικό αριθμό χρηστών (16) με ισάριθμο αριθμό μηχανημάτων (16). Δύο μηχανήματα έχουν καλύτερες επιδόσεις από τα υπόλοιπα (τρεις φορές πιο γρήγορα). Οι χρήστες εκτελούν αλγόριθμους από όλες τις κατηγορίες όσο αναφορά στο χρόνο (αργούς μέχρι πολύ γρήγορους), της ίδιας όμως κατηγορίας όσο αναφορά στον όγκο δεδομένων. Πρέπει να τονιστεί ότι ο αναμενόμενος χρόνος για την εκτέλεση των αλγόριθμων υπολογίζεται με βάση τα μηχανήματα με την μικρότερη υπολογιστική ισχύ.

Οι χρήστες χωρίζονται σε δύο κατηγορίες, τους απαιτητικούς χρήστες (8) που παράγουν σχετικά συχνές αιτήσεις (κάθε πέντε λεπτά) για εκτέλεση αλγόριθμων αποτελούμενες από αλγόριθμους με μέτρια προς μεγάλη χρονική διάρκεια εκτέλεσης, και τους απλούς χρήστες (8) χωρίς μεγάλες απαιτήσεις, οι οποίοι εκτελούν αλγόριθμους με μικρή χρονική διάρκεια εκτέλεσης, με τον ίδιο ρυθμό.

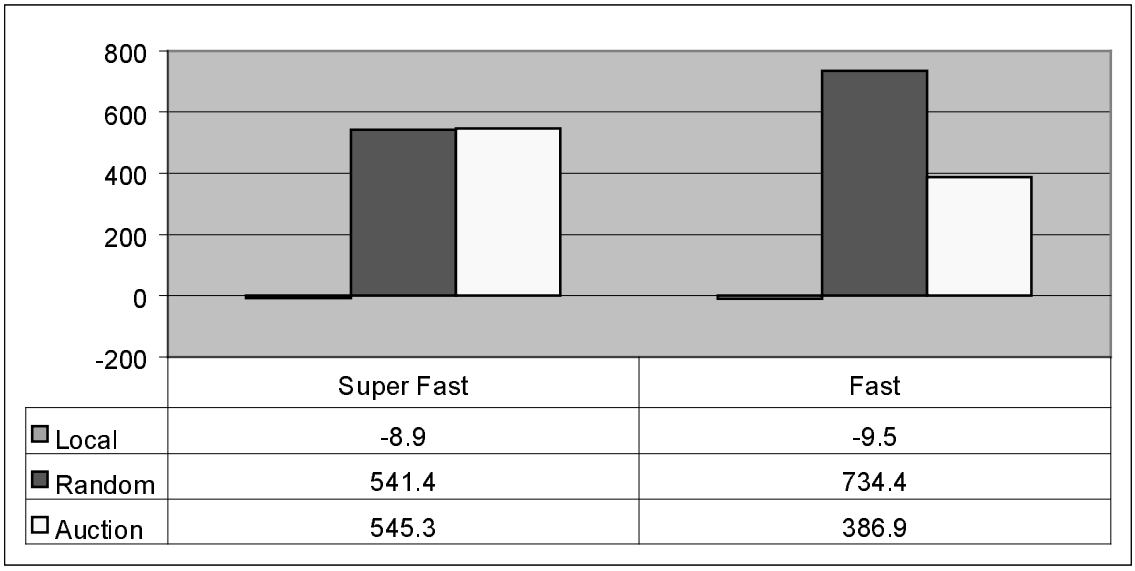
Τα αποτελέσματα των πειραμάτων φαίνονται στα ακόλουθα διαγράμματα. Τα 'γρήγορα' μηχανήματα είναι τα υπ' αριθμόν 8 και 16. Ένας χρήστης από την πρώτη κατηγορία χρησιμοποιεί το πρώτο μηχάνημα και ένας χρήστης από τη δεύτερη κατηγορία χρησιμοποιεί το δεύτερο μηχάνημα.

Παρατηρούμε τα εξής:

- Για πολύ γρήγορους και γρήγορους αλγόριθμους, η τοπική ανάθεση δίνει συνολικά πολύ καλύτερους χρόνους. Αυτό συμβαίνει γιατί οι πολύ γρήγοροι και γρήγοροι αλγόριθμοι εκτελούνται σε λιγότερο χρόνο από το χρονικό διάστημα που μεσολαβεί μέχρι ο χρήστης να ζητήσει νέα εκτέλεση.

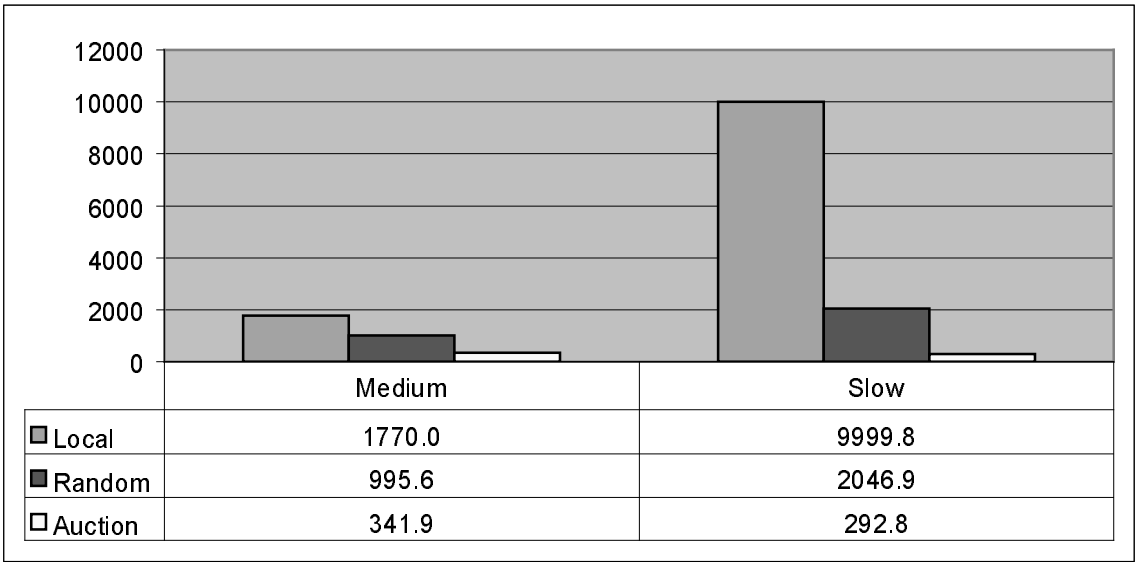
- Για τους μέτριους και αργούς αλγόριθμους, ο μηχανισμός ανάθεσης με δημοπρασία δίνει πολύ καλύτερους χρόνους, με πολύ καλή βελτίωση στην περίπτωση των αργών αλγόριθμων.

- Η συνολική χρήση των μηχανημάτων παρουσιάζει διακυμάνσεις στην τοπική και τυχαία ανάθεση, ενώ στον μηχανισμό ανάθεσης με δημοπρασία η συνολική χρήση παραμένει σταθερή.



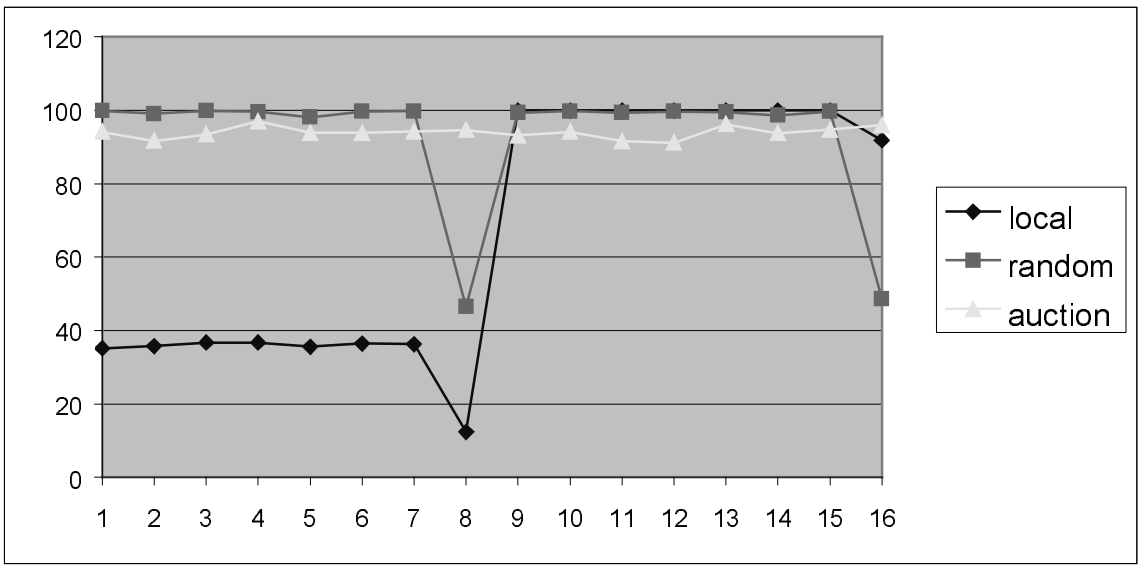
	Super Fast	Fast
Local	-8.9	-9.5
Random	541.4	734.4
Auction	545.3	386.9

Διάγραμμα 8: Μέση % αύξηση χρόνου εκτέλεσης για τους πολύ γρήγορους και γρήγορους αλγόριθμους για τους 3 μηχανισμούς ανάθεσης εκτέλεσης. Ενδιαφέρον παρουσιάζει η αρνητική τιμή για την τοπική εκτέλεση, που οφείλεται στο γεγονός ότι ο αναμενόμενος χρόνος για την εκτέλεση των αλγόριθμων υπολογίζεται με βάση τα μηχανήματα με την μικρότερη υπολογιστική ισχύ, ενώ στο συγκεκριμένο πείραμα το δίκτυο διαθέτει 2 μηχανήματα με τριπλάσια υπολογιστική ισχύ.



	Medium	Slow
Local	1770.0	9999.8
Random	995.6	2046.9
Auction	341.9	292.8

Διάγραμμα 9: Μέση % αύξηση χρόνου εκτέλεσης για τους μέτριους και αργούς αλγόριθμους για τους 3 μηχανισμούς ανάθεσης εκτέλεσης.



Διάγραμμα 10: % χρήση του κάθε μηχανήματος που συμμετέχει στο δίκτυο για τους τρεις μηχανισμούς ανάθεσης εκτέλεσης. Να σημειωθεί ότι το διάγραμμα έχει νόημα μόνο για σημεία – οι γραμμές σχεδιάστηκαν για ευκολία παρουσίασης.

Α.4 Συζήτηση

Λόγω της έλλειψης ενός ικανοποιητικού μοντέλου των χρηστών του συστήματος, ώστε να χρησιμοποιηθεί στον σχεδιασμό των πειραμάτων, τα πειράματα που παρουσιάστηκαν χρησιμοποίησαν ένα υποθετικό μοντέλο χρηστών που αντιστοιχεί σε ακραίες καταστάσεις όσο αναφορά στη χρήση του περιβάλλοντος. Η μοντελοποίηση του τρόπου αλληλοεπίδρασης του κάθε χρήστη με το σύστημα είναι πολύ δύσκολη μέχρι και αδύνατη. Αυτό συμβαίνει γιατί η σωστή μοντελοποίηση ενός χρήστη προϋποθέτει τη μοντελοποίηση των νοητικών διεργασιών του, της σκέψης του, των εμπειριών του, της ψυχολογικής και σωματικής του κατάστασης, σε ένα καλά καθορισμένο χώρο εργασίας και όταν η αποστολή του είναι καθορισμένη και μοντελοποιήσιμη. Ωστόσο, αυτό δεν συμβαίνει στη συγκεκριμένη περίπτωση, μια και το προτεινόμενο περιβάλλον αφορά στην υποστήριξη υπηρεσιών παροχής επεξεργασίας εικόνας, ένα τομέα με πολλές διαφορετικές εφαρμογές και μεγάλο εύρος σε ανάγκες και συνήθειες χρηστών. Σαν αποτέλεσμα, διάφορες παραδοχές και απλουστεύσεις είναι απαραίτητο να γίνουν, ώστε να καταλήξουμε σε ένα μοντέλο χρηστών που να αντιπροσωπεύει μια γενικότερη περίπτωση με ακραίες για το περιβάλλον συνθήκες, για χρήση στα πειράματα προσομοίωσης στα πλαίσια αυτής της εργασίας.

Πρέπει λοιπόν να τονίσουμε ότι τα αποτελέσματα των παραπάνω πειραμάτων πρέπει να συζητηθούν μόνο ποιοτικά, μια και ποσοτικά συμπεράσματα αφορούν μόνο το συγκεκριμένο μοντέλο χρηστών και όχι την γενικότερη περίπτωση. Επομένως, τα

πειράματα προσομοίωσης που παρατίθονται σ' αυτό το Παράρτημα είναι κυρίως ενδεικτικά της συμπεριφοράς του μηχανισμού διαχείρισης πόρων σε κάποιες ακραίες συνθήκες (είναι χαρακτηριστικό ότι τα πειράματα προϋποθέτουν ένα φορτωμένο στο σύνολό του σύστημα επεξεργαστών και δεν χρησιμοποιούν καθόλου έλεγχο τοπικής κρυφής μνήμης για πιθανή ύπαρξη δεδομένων εισόδου). Κατά συνέπεια, η συζήτηση που ακολουθεί είναι κατά κύριο λόγο ποιοτική.

Στα παραπάνω πειράματα παρατηρούμε, λοιπόν, ότι η πολιτική της δημοπρασίας έχει σαν αποτέλεσμα την καλύτερη αξιοποίηση των πόρων του συστήματος. Όλα τα μηχανήματα, ανεξαρτήτως δυνατοτήτων χρησιμοποιούνται στο μέγιστο των δυνατοτήτων τους, χωρίς να υπάρχουν μηχανήματα που υπολειτουργούν. Άμεσο αποτέλεσμα είναι η περάτωση περισσότερων διεργασιών σε μικρότερο χρονικό διάστημα. Οι διάφορες διεργασίες έχουν πολύ καλύτερο χρόνο απόκρισης από το σύστημα. Οι χρήστες του συστήματος συνολικά, είναι καλύτερα ικανοποιημένοι από τη χρήση της πολιτικής της δημοπρασίας.

Πρέπει να σημειωθεί ότι στην περίπτωση των πολύ γρήγορων διεργασιών, οι χρόνοι εκτέλεσης με μηχανισμό δημοπρασίας είναι κάπως χειρότεροι από την τοπική εκτέλεση, λόγω του πολύ μεγάλου φόρτου του συστήματος και του δικτύου, πράγμα αναμενόμενο μια που η ανάθεση με δημοπρασία δεν προσπαθεί να καλυτερεύσει μεμονωμένες διεργασίες αλλά την συνολική απόδοση του συστήματος. Η χρήση ενός καλύτερου δικτύου, θα είχε σαν αποτέλεσμα την ελάττωση του χρόνου που χρειάζονται τα δεδομένα να μεταφερθούν στο δίκτυο, με άμεσο αποτέλεσμα την καλυτέρευση της απόδοσης του μηχανισμού δημοπρασίας έναντι της τοπικής εκτέλεσης ακόμη και για τους πολύ γρήγορους αλγόριθμους (εφόσον ένα μηχάνημα είναι ανενεργό και ο χρόνος για να εκτελεστεί μία διεργασία σε αυτό είναι μηδαμινός, τότε είναι πολύ πιθανό η καλύτερη λύση να είναι η ανάθεση εκτέλεσης σ' αυτό το μηχάνημα).

Ο πράκτορας εκτέλεσης στον προσομοιωτή αναλαμβάνει την εκτέλεση διεργασιών, ανεξάρτητα από τον αριθμό των διεργασιών που ήδη εκτελεί και την καθυστέρηση που επιφέρει μία νέα διεργασία στις υπόλοιπες. Η εφαρμογή διαφόρων πολιτικών για την προσφορά στην δημοπρασία μπορεί να δώσει πολύ καλύτερα αποτελέσματα στο χρόνο αναμονής μιας συγκεκριμένης διεργασίας στο σύστημα, αλλά θα έχει σαν αποτέλεσμα την απόρριψη ή τη σημαντική καθυστέρηση άλλων διεργασιών.

Τα πειράματα που παρουσιάζονται σ' αυτό το Παράρτημα έγιναν υποθέτοντας διάφορες κατηγορίες αλγόριθμων όσο αναφορά στον αναμενόμενο χρόνο εκτέλεσης, όπως φαίνεται και στην ανάλογη κατηγοριοποίηση της παραγράφου Α1-Χρήστες. Οι χρόνοι αυτοί αναμένεται να ελαττωθούν όσο η τεχνολογία των επεξεργαστών εξελίσσεται. Ωστόσο, η μείωση του χρόνου εκτέλεσης μιας διεργασίας συνοδεύεται κατά κανόνα με αύξηση του όγκου των δεδομένων προς επεξεργασία καθώς και των αιτήσεων για επεξεργασία, με αποτέλεσμα η αύξηση της υπολογιστικής δύναμης να 'απορροφάται' συντομότατα από τους χρήστες και τις απαιτήσεις τους. Παράλληλα, εκτός από την ελάττωση που θα επέλθει στους αναμενόμενους χρόνους εκτέλεσης, θα επέλθει ελάττωση και στο χρόνο μεταφοράς των δεδομένων στο δίκτυο, χρησιμοποιώντας διαφορετικές τεχνολογίες δικτύων (π.χ. ATM), με αποτέλεσμα την (πολλές φορές) καλυτέρευση της απομακρυσμένης εκτέλεσης, ακόμη και στην περίπτωση των πολύ γρήγορων (κατ' απόλυτη τιμή) αλγόριθμων. Η χρήση

του προσομοιωτή, μεταβάλλοντας τις διάφορες παραμέτρους ταχύτητας, δικτύου, υπολογιστών, χρόνων εκτέλεσης αλγόριθμων, θα μπορέσει να δώσει απαντήσεις σε μια υποθετική εφαρμογή ενός τέτοιου σεναρίου. Οι απαντήσεις αυτές μπορούν να χρησιμοποιηθούν για τη λήψη σχεδιαστικών αποφάσεων για το σύστημα προς κατασκευή, για τον καθορισμό των διάφορων πολιτικών ανάθεσης πόρων, καθώς και για τον καθορισμό των διάφορων πολιτικών χρέωσης.

Η πραγματική αξία του προσομοιωτή και της χρήσης του για ανάλογα πειράματα αναδεικνύεται σε περίπτωση μελέτης για την εγκατάσταση του περιβάλλοντος σε κάποιον πραγματικό οργανισμό. Σε έναν υπαρκτό οργανισμό με γνωστή τη συμπεριφορά των χρηστών, ο προσομοιωτής μπορεί να χρησιμοποιηθεί για το σχεδιασμό του μηχανισμού κατανομής πόρων με τρόπο που να συμπεριλάβει τις απαιτήσεις των χρηστών, διάφορους πιθανούς τοπικούς περιορισμούς, καθώς και να λάβει υπόψη του τα πιθανά σχέδια χρήσης και υπαρκτές ανάγκες επεξεργασίας δεδομένων που έχουν οι χρήστες του συγκεκριμένου οργανισμού. Σε μια τέτοια περίπτωση υπάρχει ένα συγκεκριμένο μοντέλο χρηστών (που προέρχεται είτε από πειραματική καταγραφή είτε από θεωρητική προσέγγιση) και, κυρίως, υπάρχει συγκεκριμένη άποψη για τον ορισμό της ποιότητας υπηρεσιών επεξεργασίας δεδομένων ανάλογα με τις ανάγκες και τις ιδιαιτερότητες του οργανισμού. Έτσι σ' αυτή την περίπτωση, οι παράμετροι του προσομοιωτή αρχικοποιούνται ώστε να αντικατοπτρίζουν το συγκεκριμένο μοντέλο των χρηστών και διεξάγονται πειράματα ανάλογα με τα παραπάνω. Τα αποτελέσματα των πειραμάτων συγκρίνονται με την επιθυμητή συμπεριφορά (όπως αυτή ορίζεται για τον κάθε οργανισμό) και αν είναι απαραίτητο γίνεται κάποια ανα-διοργάνωση των πολιτικών ανάθεσης εκτέλεσης των πρακτόρων ή ακόμα και μια ανα-διοργάνωση των πόρων του οργανισμού, και τα πειράματα επαναλαμβάνονται. Η διαδικασία επαναλαμβάνεται ώσπου να αναδειχθεί μια σύνθεση πολιτικών ανάθεσης εκτέλεσης που να ικανοποιεί τις απαιτήσεις του οργανισμού για ποιότητα υπηρεσιών. Έτσι, με βάση τις ενδείξεις των πειραμάτων προσομοίωσης αρχικοποιείται ο πληθυσμός των πρακτόρων στην συγκεκριμένη εγκατάσταση του περιβάλλοντος. Τέλος, ο προσομοιωτής μπορεί να χρησιμοποιηθεί για τον καθορισμό των διαφόρων τιμών και πολιτικών χρέωσης των παρεχόμενων υπηρεσιών χρησιμοποιώντας υποθετικά σενάρια χρήσης και υπολογίζοντας κάθε φορά τα αναμενόμενα κέρδη από τη χρήση του συστήματος. Με αυτόν τον τρόπο χρήσης ο προσομοιωτής μπορεί να υποστηρίξει την εύκολη παραμετροποίηση του προτεινόμενου περιβάλλοντος και να γίνει ένα εργαλείο για την αναδιοργάνωση οργανισμών στον τομέα παροχής υπηρεσιών επεξεργασίας δεδομένων.

ΠΑΡΑΡΤΗΜΑ Β

Συστήματα Επεξεργασίας Εικόνων

Β.1 Εργαλειοθήκες – Βιβλιοθήκες

PBMPlus [21]: Συλλογή εργαλείων διαχείρισης εικόνων. Χρησιμοποιείται πολύ συχνά στο UNIX για μετατροπή εικόνων μεταξύ διαφορετικών τύπων αρχείων.

NetPBM [22]: Αποτελεί μία βελτιωμένη έκδοση του PBMPlus με πολλές επιπρόσθετες λειτουργίες και τύπους αρχείων εικόνων.

Utah Raster Toolkit [23]: Είναι μία συλλογή εντολών στο UNIX καθώς και βιβλιοθηκών σε C, για πρόσβαση και απλή επεξεργασία εικόνων βάθους 8 bit/pixel. Αποτελεί μία από τις πρώτες προσπάθειες κατασκευής μιας εργαλειοθήκης για χρήση στην επεξεργασία εικόνων.

Visualization Toolkit (VTK) [24]: Είναι μία βιβλιοθήκη γραμμένη σε C++ για την κατασκευή τρισδιάστατων γραφικών και προγραμμάτων γραφικής απεικόνισης δεδομένων. Μπορεί να χρησιμοποιηθεί μέσα από τη γλώσσα προγραμματισμού Tcl με αποτέλεσμα την πολύ γρήγορη κατασκευή πρότυπων εφαρμογών

ImageGear [25]: Εμπορικό προϊόν της εταιρείας AccuSoft. Παρέχει βιβλιοθήκη για ανάγνωση και εγγραφή διαφορετικών τύπων αρχείων εικόνων, καθώς και δυνατότητα εκτύπωσης εικόνων, και κάποιες βασικές λειτουργίες επεξεργασίας.

ImageVision [26]: Είναι μία βιβλιοθήκη για τη δημιουργία, επεξεργασία και παρουσίαση εικόνων στα μηχανήματα της Silicon Graphics. Η βιβλιοθήκη παρέχει στους κατασκευαστές εφαρμογών επεξεργασίας εικόνας ένα πλήρες πλαίσιο για τη διαχείριση εικόνων.

LeadTools [27]: Εμπορικό προϊόν της εταιρείας LEAD Technologies. Παρέχει μία πολύ καλή συλλογή από συναρτήσεις επεξεργασίας εικόνας (μετασχηματισμούς, φίλτρα, περιοχές ενδιαφέροντος, ζωγραφική).

Β.2 Ολοκληρωμένα Προγράμματα Επεξεργασίας Εικόνων

Adobe Photoshop [30]: Το πιο γνωστό ολοκληρωμένο πρόγραμμα επεξεργασίας εικόνας από την εταιρεία Adobe Systems Inc., San Jose, CA, USA. Χρησιμοποιείται κυρίως

από γραφίστες για την επεξεργασία εικόνων με προορισμό την εκτύπωση σε περιοδικά βιβλία ή την έκδοσή τους στο Internet. Παρέχει τη δυνατότητα στους χρήστες να προσθέσουν λειτουργίες στο βασικό πρόγραμμα.

Corel Photopaint [31]: Ο κυρίως ανταγωνιστής του Adobe Photoshop από την εταιρεία Corel Corporation, Ottawa, Canada. Χρησιμοποιείται για αντίστοιχους σκοπούς και παρέχει αντίστοιχες λειτουργίες.

Paint Shop Pro [32]: Ένα από τα πιο καλά γενικής χρήσης προγράμματα επεξεργασίας εικόνων που διατίθεται για κοινή χρήση, με ελάχιστο κόστος, από την εταιρεία Jasc Inc., Eden Prairie, MN, USA. Παρέχει αντίστοιχες δυνατότητες με το Adobe Photoshop.

XV [33]: Το πιο γνωστό πρόγραμμα παρουσίασης εικόνων που χρησιμοποιείται σε ακαδημαϊκούς χώρους και γενικότερα στο περιβάλλον UNIX/X11. Παρέχει πολύ βασικές λειτουργίες (διόρθωση χρωμάτων, αποκοπή και συγκόλληση κομματιών) και υποστηρίζει τους πιο γνωστούς τύπους αρχείων εικόνων. Αναπτύχθηκε από το Computer and Information Science Dept., University of Pennsylvania, USA.

GIMP [34]**:** Ένα πολύ καλό πρόγραμμα επεξεργασίας εικόνας το οποίο διατίθεται για κοινή χρήση δωρεάν. Συγκρίνεται με αντίστοιχα εμπορικά προγράμματα και παρέχει όλα τα εργαλεία και τα φίλτρα που κάποιος μπορεί να βρει σε αυτά. Πολύ σημαντική είναι η δυνατότητα που παρέχει για την επέκταση της λειτουργικότητας μέσω της χρήσης εξωτερικών υπό-προγραμμάτων (plugins). Η αρχική ανάπτυξη του προγράμματος έγινε στο University of California at Berkeley, USA.

ImageMagick [35]: Το δεύτερο πιο γνωστό πρόγραμμα παρουσίασης εικόνων που χρησιμοποιείται σε ακαδημαϊκούς χώρους και γενικότερα στο περιβάλλον UNIX. Υποστηρίζει αρκετούς τύπους αρχείων εικόνων και μπορεί να εκτελεστεί είτε σε γραφικό περιβάλλον (GUI) είτε όχι (γραμμή εντολών). Αρχικά αναπτύχθηκε από τους E. I. du Pont de Nemours and Company, Delaware, USA.

OSIRIS [37]**:** Είναι ένα πρόγραμμα που χρησιμοποιείται για την παρουσίαση, επεξεργασία και σχολιασμό ιατρικών εικόνων. Υποστηρίζει ιατρικές εικόνες που προέρχονται από διαφορετικές διαγνωστικές μονάδες. Χρησιμοποιείται ευρύτατα στον ιατρικό χώρο.

BIPS-HELIOS [38]: Αποτελεί μία προσπάθεια για την ενοποίηση εργαλείων ανάλυσης και επεξεργασίας εικόνας σε ένα νοσοκομειακό περιβάλλον, το οποίο αναπτύχθηκε στα πλαίσια του ευρωπαϊκού προγράμματος AIM. Υπάρχει μία βάση αλγόριθμων και λειτουργιών οι οποίες μπορούν να προσπελαστούν από το χρήστη μέσω ενός ειδικού προγράμματος. Ο χρήστης μπορεί να κατασκευάσει μία ακολουθία από διάφορες λειτουργίες και να ζητήσει την εκτέλεση τους. Η εκτέλεση των διαφόρων λειτουργιών είναι μία εργασία την οποία αναλαμβάνει ένα ειδικό κομμάτι της εφαρμογής. Υποστηρίζεται μια μορφή κατανεμημένης εκτέλεσης, χωρίς όμως να χρησιμοποιείται κάποια μορφής βέλτιστης ανάθεσης εργασιών.

Β.3 Οπτικά Περιβάλλοντα Επεξεργασίας Εικόνων

KBVision: [41] Είναι ένα σύνολο εργαλείων που λύνουν προβλήματα αναγνώρισης και κατανόησης εικόνων. Παρέχει ένα οπτικό περιβάλλον προγραμματισμού, εργαλεία αναπαράστασης δεδομένων καθώς και μία μεγάλη βιβλιοθήκη διαφόρων λειτουργιών. Ο χρήστης κατασκευάζει οπτικά προγράμματα ορίζοντας διάφορες λειτουργίες στο επίπεδο που τις ενώνει μεταξύ τους χρησιμοποιώντας βέλη. Το αποτέλεσμα είναι μία αναπαράσταση ροής δεδομένων ως γράφος. Οι γράφοι μπορούν να αποθηκευτούν και να χρησιμοποιηθούν αναδρομικά, δηλαδή μέσα από άλλους γράφους. Αποτελεί εμπορικό προϊόν της εταιρείας Amerinex Applied Imaging Inc., USA

Aphelion [42]: Είναι ένα εργαλείο που επιτρέπει τη λύση προβλημάτων αναγνώρισης και κατανόησης εικόνων. Υποστηρίζει μία πληθώρα ειδικών περιφερειακών και μηχανημάτων σχετικών με την απόκτηση εικόνων. Παρέχει ένα οπτικό περιβάλλον προγραμματισμού όμοιο με το αντίστοιχο του Khoros καθώς και μία μεγάλη συλλογή από τελεστές και λειτουργίες επεξεργασίας εικόνων. Αποτελεί εξέλιξη του KBVision και σε συνδυασμό με το γεγονός ότι η εταιρεία που το κατασκευάζει (Amerinex Applied Imaging Inc.) είναι ο συντονιστής του προγράμματος IUE [28], εγγυάται τη συνεχή εξέλιξη του προγράμματος με τις τελευταίες εξελίξεις και ανακαλύψεις στο χώρο της κατανόησης και αναγνώρισης εικόνων.

SCIL-Image: [43] Αποτελεί ένα συνδυασμό ενός περιβάλλοντος για την αλληλεπίδραση με το χρήστη (SCIL) και μιας βιβλιοθήκης για επεξεργασία εικόνων (Image). Το περιβάλλον μπορεί να τροποποιηθεί και να χρησιμοποιηθεί σε πολλούς διαφορετικούς χώρους, εκτός από τον χώρο της επεξεργασίας εικόνας. Ο χρήστης μπορεί να προγραμματίσει το περιβάλλον μέσω μίας γλώσσας προγραμματισμού C ή οποία μεταφράζεται δυναμικά (interpreted). Η γλώσσα προγραμματισμού συνδέεται με τη βιβλιοθήκη μέσω ειδικών αρχείων που περιγράφουν τις εντολές της βιβλιοθήκης. Το περιβάλλον κατασκευάζει δυναμικά διάλογους με τους οποίους ο χρήστης ορίζει τις παραμέτρους κάθε λειτουργίας της βιβλιοθήκης. Εναλλακτικά ο χρήστης μπορεί να χρησιμοποιήσει ένα περιβάλλον οπτικού προγραμματισμού που βασίζεται στο παράδειγμα της ροής δεδομένων, το οποίο ενσωματώνεται μέσα στο υπόλοιπο σύστημα. Ο χρήστης μπορεί γραφικά να διαλέξει τις λειτουργίες που επιθυμεί, στη συνέχεια τις τοποθετεί σε ένα φύλλο εργασίας, όπου μπορεί να περιγράψει την ροή των δεδομένων ανάμεσα σε αυτές τις λειτουργίες, απλά ενώνοντας τις λειτουργίες με διάφορα βέλη, τα οποία δείχνουν την ροή των δεδομένων.

Khoros: [44,45,46] Αποτελεί μία συλλογή προγραμμάτων για την επεξεργασία δεδομένων, αναπαράσταση δεδομένων (data visualization), επεξεργασία εικόνων και ανάπτυξη λογισμικού. Περιλαμβάνει ένα περιβάλλον που επιτρέπει την οπτική προσομοίωση και τον οπτικό προγραμματισμό, ένα σύνολο από εργαλεία ανάπτυξης, ένα περιβάλλον για την κατασκευή επιφανειών χρήσης, ένα πρόγραμμα παρουσίασης εικόνων, και μία μεγάλη συλλογή από λειτουργίες για επεξεργασία εικόνων, διαχείριση δεδομένων, αναπαράσταση δεδομένων, γεωμετρία και πράξεις σε πίνακες. Το σύστημα μπορεί εύκολα να τροποποιηθεί και να χρησιμοποιηθεί σε διαφορετικούς χώρους εφαρμογών. Το όλο σύστημα, αποτελείται από διάφορα επίπεδα τα οποία

αλληλεπιδρούν. Ένα από τα πιο δυνατά χαρακτηριστικά του είναι η υψηλού επιπέδου, οπτική γλώσσα προγραμματισμού που χρησιμοποιεί (cantata) [44]. Είναι υλοποιημένο στη γλώσσα προγραμματισμού C και υποστηρίζει μία πληθώρα UNIX λειτουργικών συστημάτων. Επιτρέπει την εκτέλεση εξωτερικών αλγόριθμων οι οποίοι μπορεί να βρίσκονται είτε στο τοπικό είτε σε ένα απομακρυσμένο σύστημα, το οποίο όμως είναι προκαθορισμένο για κάθε αλγόριθμο.

ΠΑΡΑΡΤΗΜΑ Γ

Συναρτήσεις Κατακερματισμού Ενός Δρόμου

Μια συνάρτηση κατακερματισμού ενός δρόμου, H(M), δέχεται ένα μήνυμα οποιουδήποτε μεγέθους, M, και επιστρέφει ένα μήνυμα ορισμένου μεγέθους h:

h = H(M), όπου το h έχει μήκος m.

Πολλές συναρτήσεις μπορούν να δεχθούν ένα μήνυμα οποιουδήποτε μεγέθους και να επιστρέψουν ένα μήνυμα ορισμένου μεγέθους, αλλά οι συναρτήσεις κατακερματισμού ενός δρόμου έχουν μερικά πρόσθετα χαρακτηριστικά:

- δοσμένου του μηνύματος M, το h υπολογίζεται εύκολα.
- δοσμένου του h, είναι δύσκολο να υπολογιστεί ένα M τέτοιο ώστε H(M)=h.
- δοσμένου του μηνύματος M, είναι δύσκολο να βρεθεί ένα άλλο μήνυμα N, τέτοιο ώστε H(M) = H(N).

Σε μερικές εφαρμογές η ιδιότητα της συνάρτησης να είναι ενός δρόμου, δεν αρκεί. Χρειαζόμαστε η συνάρτηση να αντιστέκεται στις συγκρούσεις, δηλαδή υπάρχει η επιπλέον απαίτηση να ισχύει το παρακάτω:

- είναι δύσκολο να βρεθούν δύο τυχαία μηνύματα M και N, τέτοια ώστε H(M)=H(N).

Χρησιμοποιώντας μηνύματα διαφορετικών μεγεθών, πιθανά διαφορετικά μηνύματα σε μέγεθος να έχουν την ίδια τιμή κατακερματισμού. Το πρόβλημα τις ίδιας τιμής για διαφορετικά σε μέγεθος μηνύματα μπορούμε να το εξαλείψουμε χρησιμοποιώντας μία τεχνική στην οποία σπάμε το μήνυμα σε κομμάτια, και για κάθε κομμάτι παράγουμε την τιμή κατακερματισμού, χρησιμοποιώντας τη σχέση h(i)=f[M(i),h(i-1)]. Η τιμή κατακερματισμού του μηνύματος είναι η τιμή κατακερματισμού της συνάρτησης για το τελευταίο κομμάτι. Η τεχνική αυτή ονομάζεται MD-strengthening.

Το μέγεθος της τιμής που παράγεται από την hash συνάρτηση, είναι συνήθως 128-bit. Αυτό έχει σαν αποτέλεσμα κάποιος που προσπαθεί να βρει δύο μηνύματα τα οποία παράγουν την ίδια τιμή, να πρέπει να χρησιμοποιήσει 2^{64} τυχαία μηνύματα. Προφανώς κάτι τέτοιο δεν παρέχει απόλυτη ασφάλεια, αλλά είναι αρκετά δύσκολο να γίνει. Μια αναλυτική περιγραφή γνωστών συναρτήσεων κατακερματισμού ενός δρόμου δίνεται στο [116].

Χρήση συναρτήσεων κατακερματισμού για ταυτοποίηση εικόνων

Η ανάγκη για την ελαχιστοποίηση της μεταφοράς δεδομένων στο δίκτυο κατά τη διάρκεια πολλαπλών εκτελέσεων αλγόριθμων, οδήγησε στη χρήση συναρτήσεων κατακερματισμού

ενός δρόμου, για την ταυτοποίηση της κάθε εικόνας. Η διαδικασία της ταυτοποίησης κάθε εικόνας και η χρήση της ταυτότητας στη διαδικασία της εκτέλεσης ενός αλγόριθμου, γίνεται ως εξής:

- Για κάθε εικόνα που εισάγεται στο σύστημα, υπολογίζεται η τιμή της συνάρτησης κατακερματισμού. Η τιμή αυτή χρησιμοποιείται συνοδευόμενη από στοιχεία που κρίνονται κατάλληλα σε κάθε περίπτωση (όνομα χρήστη, τύπος εικόνας), και αποστέλλεται όπου χρειάζεται η ταυτότητα της εικόνας.
- Εφόσον ο χρήστης έχει επιλέξει τη χρήση ψηφιακής ταυτοποίησης εικόνας, κάθε αίτηση που κάνει για εκτέλεση ενός αλγόριθμου, συνοδεύεται από τον αριθμό ταυτότητας των δεδομένων που χρησιμοποιεί ο αλγόριθμος για είσοδο.
- Ο κάθε πράκτορας εκτέλεσης ξέρει τους αριθμούς ταυτότητας των δεδομένων που έχει στην τοπική κρυφή μνήμη. Αν ο αριθμός ταυτότητας ενός δεδομένου που χρειάζεται για την εκτέλεση του αλγόριθμου υπάρχει στην τοπική κρυφή μνήμη, το δεδομένο αυτό δεν μεταφέρεται από το δίκτυο.

Για την παραγωγή των αριθμών ταυτότητας στο DIPE χρησιμοποιήθηκε ο αλγόριθμος MD5 (Message Digest-5) [80]. Για τον MD5 ισχύουν τα εξής:

- Είναι υπολογιστικά ανέφικτο να βρεθούν δύο μηνύματα που η συνάρτηση κατακερματισμού να παράγει την ίδια τιμή. Ο μόνος τρόπος είναι η εξάντληση των δυνατών περιπτώσεων (2^{64}).
- Είναι αρκετά γρήγορος στον υπολογισμό της τιμής (χρησιμοποιεί 32-bit πράξεις).
- Είναι αρκετά απλός χωρίς χρήση πολύπλοκων δομών δεδομένων.
- Παράγει τιμή μήκους 128-bit.

Η χρήση του αλγόριθμου MD5 μας δίνει μία ικανοποιητική ασφάλεια στην ταυτοποίηση των εικόνων. Έχει μικρή πιθανότητα δύο διαφορετικές εικόνες να έχουν τον ίδιο αριθμό ταυτότητας. Σε περίπτωση που κριθεί σκόπιμο, μπορεί να αυξηθεί η ασφάλεια της ταυτοποίησης των εικόνων (δεδομένων) χρησιμοποιώντας επιπλέον στοιχεία για την παραγωγή της ταυτότητας κάθε εικόνας, (π.χ. το όνομα του χρήστη, το μηχάνημα από το οποίο παράχθηκε η αίτηση για εκτέλεση, το όνομα του αρχείου της εικόνας, κλπ).

ΠΑΡΑΡΤΗΜΑ Δ

Σχετικές Δημοσιεύσεις

Η παρούσα εργασία έχει βασιστεί σε ιδέες σχεδιασμού και υλοποίησης από το CoMed, ένα σύστημα για την σύγχρονη συνεργατική τηλεδιάσκεψη στην ιατρική:

> Μ. Ζήκος, "CoMed: Συνεργασία στην Ιατρική", Διπλωματική Εργασία, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, Σεπτέμβρης 1995.
>
> M. Zikos, C. Stephanidis, and S.C. Orphanoudakis, "CoMed: Cooperation in Medicine", Proceedings of EuroPACS'96, pp. 88-92, Heraklion, Crete, Greece, October 3-5, 1996.

Δημοσιεύσεις που προέκυψαν με βάση την παρούσα εργασία είναι οι παρακάτω:

> M. Zikos, E. Kaldoudi, and S.C. Orphanoudakis, "Image Processing within an Integrated Teleradiology Services Network", In: H.U. Lemke, M.W. Vannier, K. Inamura (eds.), Proceedings of CAR'97, p. 1027, Berlin, Germany, June 25-28, 1997.
>
> M. Zikos, E. Kaldoudi, and S.C. Orphanoudakis, "DIPE: A Distributed Environment for Medical Image Processing", Proceedings of MIE'97, p. 465-469, Porto Carras, Greece, May 25-29, 1997.
>
> M. Zikos, E. Kaldoudi, and S.C. Orphanoudakis, "Image Processing Services in Health Telematics Networks", ERCIM News, Vol.29, 1997.
>
> E. Kaldoudi, M. Zikos, and S.C. Orphanoudakis, "An Environment Supporting Visual Information Processing Services", Technical Report FORTH-ICS/TR-205, Institute of Computer Science, Foundation for Research and Technology – Hellas, August 1997.

Τέλος, η αρχιτεκτονική που αναπτύχθηκε στην παρούσα εργασία αποτέλεσε τη βάση για το σχεδιασμό περιβάλλοντος για την επεξεργασία ροών εργασίας:

> E. Kaldoudi, M. Zikos, E. Leisch, S.C. Orphanoudakis, "Agent-Based Workflow Processing for Functional Integration and Process Re-engineering in the Health Care Domain", Proceedings of EuroPACS'97, Pisa, Italy, September 25-27, 1997.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. J. Birnbaum, "Pervasive Information Systems", Comm. ACM, vol. 40(2), 40-41, 1997.
2. J.B. Quinn, J.J. Baruch, P.C. Paquette, "Technology in Services", Scientific American, vol. 257(6), 50-58, 1987.
3. J. M. Tien, D. Berg, "Systems Engineering in the Growing Service Economy", IEEE Trans. On Systems, Man, and Cybernetics, vol. 25(5), 721-726, May 1995.
4. Museum of London, "Exhibition on the History of London", London, 1997.
5. Negroponte, "Being Digital", Coronet Books, 1995.
6. M. Gray , "Internet Growth Summary",

   http://www.mit.edu/people/mkgray/net/internet-growth-summary.html
7. G. Flammia, M. McCandless, "From Software to Service: The Transformation of Shrink-Wrapped Software on the Internet", IEEE Expert, vol. 12(2), 4-6, 1997.
8. R. Comerford, "The Battle for the Desktop", IEEE Spectrum, 21-28, May 1997.
9. Network Computer Reference Profile 1, http://www.nc.ihost.com
10. M. Zikos, E. Kaldoudi, and S.C. Orphanoudakis, "Image Processing within an Integrated Teleradiology Services Network", In: H.U. Lemke, M.W. Vannier, K. Inamura (eds.), Proceedings of CAR'97, p. 1027, Berlin, Germany, June 25-28, 1997.
11. M. Zikos, E. Kaldoudi, and S.C. Orphanoudakis, "DIPE: A Distributed Environment for Medical Image Processing", Proceedings of MIE'97, p. 465-469, Porto Carras, Greece, May 25-29, 1997.
12. S.C. Orphanoudakis, M. Tsiknakis, C. Chronaki, S. Kostomanolakis, M. Zikos, and Y. Tsamardinos, "Development of an Integrated Image Management and Communication System on Crete", In: H.U. Lemke, K. Inamura, C.C. Jaffe, M.W. Vanier (eds.), Proceedings of CAR'95, Berlin, p. 481-487, 1995.
13. S.C. Orphanoudakis, E. Kaldoudi, and M. Tsiknakis, "Technological Advances in Teleradiology", European Journal of Radiology, vol. 22, 205-217, 1996.
14. S.C. Orphanoudakis, C. Chronaki, and S. Kostomanolakis, "*I²C*: A System for the Indexing, Storage, and Retrieval of Medical Images by Content", Med. Inform., vol. 19, 109-122, 1994.
15. S.C. Orphanoudakis, C.E. Chronaki, and D. Vamvaka, "*I²Cnet*: Content-Based Similarity Search in Geographically Distributed Repositories of Medical Images", Computerized Medical Imaging and Graphics, vol. 20(4), 193-207, 1996.
16. M. Zikos, C. Stephanidis, and S.C. Orphanoudakis, "CoMed: Cooperation in Medicine", Proceedings of EuroPACS'96, pp. 88-92, Heraklion, Crete, Greece, October 3-5, 1996.
17. G. Lawton, "Extranets: Next Step for the Internet", IEEE Computer, vol. 30(5), 17, 1997.

18. New Mexico Software Inc., Albuquerque, NM, "Secure Photo Search and Edit Capabilities – Via the Web", page 48 in: B. Mazor (ed), "Imaging Solutions Issue 1997: The Contest. The Directory. The Issue.", Advanced Imaging, vol. 12(6), 24-65, 1997.

19. E. Kaldoudi, M. Zikos, E. Leisch, S.C. Orphanoudakis, "Agent-Based Workflow Processing for Functional Integration and Process Re-engineering in the Health Care Domain", Proceedings of EuroPACS'97, Pisa, Italy, September 25-27, 1997. (accepted for publication)

20. B. Mazor (ed), "Imaging Solutions Issue 1997: The Contest. The Directory. The Issue.", Advanced Imaging, vol. 12(6), 24-65, 1997.

21. "PBMPlus", ftp://wuarchive.wustl.edu/graphics/graphics/packages/pbmplus

22. "NetPBM" ftp://ftp.cs.ubc.ca/ftp/archive/netpbm

23. "Utah Raster", ftp://princeton.edu/pub/Graphics/URT

24. K. Martin, W. Schroeder, B. Lorensen "The Visualization Toolkit: An Object-Oriented Approach To 3D Graphics", Prentice Hall, NJ, 1996

25. "Image Gear", Accusoft Corporation, Westborough, Massachusetts, USA.

26. "Image Vision", Silicon Graphics, Inc., Mountain View, California, USA.

27. "Leadtools", LEAD Technologies, Inc, Charlotte, USA.

28. J.L. Mundy, and the IUE Committee, "The Image Understanding Environment Program", IEEE Expert, vol. 10(6), 64-73, 1995.

29. W.K. Pratt, "PIKS Foundation C Programmer's Guide", Manning Publications, 1995.

30. "Adobe Phtoshop", Abode Systems Inc., San Jose, CA, USA.

31. "Corel Photoshop", Corel Corporation, Ottawa, Canada.

32. "Paint Shop Pro", Jasc Inc., Eden Prairie, MN, USA.

33. "XV", Computer and Information Science Dept., University of Pennsylvania, USA.

34. "GIMP", University of California at Berkeley, USA.

35. "Imagemagick", E. I. du Pont de Nemours and Company, Delaware, USA.

36. "Visilog", Noesis S.A., Orsay, France.

37. Y. Ligier, O. Ratib, M. Logean, C. Girard, "OSIRIS : A Medical Image Manipulation System", M.D. Computing Journal, vol. 11(4), 212-218, 1994.

38. U. Engelmann, H.P. Meinzer, A. Schröter, U. Günnel, A.M. Demiris, M. Schäfer, H. Evers, F.C. Jean, P. Degoulet, "The Image Related Services of the HELIOS Software Engineering Environment", Computer Methods and Programs in Biomedicine, vol. 46, 1-12, 1995.

39. A.H. Bridle, E.W. Greisen, "The NRAO AIPS Project – A Summary", AIPS memo 87, National Radio Astronomy Observatory, USA, 1994.

40. B.A. Myers, "Taxonomies of Visual Programming and Program Visualisation", Journal of Visual Languages and Computing, vol. 1, 97-123, 1990.

41. P. Eggleston, "General Support Tools for Algorithm Development and Scientific Research in Computer Vision", Technical Report, Amerinex Advanced Imaging ftp://ftp.aai.com/pub/tech_reports/GenSupportTools.ps.gz

42. "Aphelion", Amerinex Applied Imaging, Inc., Amherst, USA.

43. R. van Balen, T. ten Kate, D. Koelma, B. Mosterd, and A.W.M. Smeulders. "ScilImage: A Multi-Layered Environment for Use and Development of Image Processing Software", in: H.I. Christensen and J.L. Crowley (eds), "Experimental Environments for Computer Vision and Image Processing", p. 107-126, World Scientific Press, Singapore, 1994.

44. M. Young, D. Argiro, S. Kubica, "Cantata: Visual Programming Environment for the Khoros System", Computer Graphics, vol. 29(2), 22-24, 1995.

45. J.R. Rasure, S. Kubica, "The Khoros Application Development Environment", in H.I Christensen and J.L Crowley(eds.), "Experimental Environments for Computer Vision and Image Processing", World Scientific 1994.

46. K. Konstantinides, J.R. Rasure, "The Khoros Software Development Environment for Image and Signal Processing", IEEE Trans. Image Processing, vol. 3(3), 243-252, 1994.

47. C. Hinton, "TRACEE: Target Recognition Algorithm Control and Evaluation Environment. User Guide, v 3.0, Aug. 29, 1996", King's College London, 1996

48. ΕΛΟΤ-ΟΤΕ, "Ενιαίο Λεξιλόγιο Τηλεπικοινωνιακής Ορολογίας", ΕΛΟΤ, Ιούνιος 1992.

49. R. A. Brooks, "Inteligence Without Reason", Computers and Thought, Proc. of IJCAI-91, Sidney, Australia, also in A.I. Memo No. 1293, AI Lab, MIT, 1991.

50. S. Russell, P. Norvig, "Artificial Intelligence: A Modern Approach", Prentice Hall, Inc., 1995.

51. P. Maes, "Modeling Adaptive Autonomous Agents", Artificial Life Journal, vol. 1(1&2), 1994.

52. M. Wooldridge, N.R. Jennings, "Intelligent Agents: Theory and Practice", Knowledge Engineering Review, vol. 10(2), 115-152, 1995.

53. S. Franklin, A. Graesser, "Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents", Proc. of the 3rd International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag, 1996.

54. N.L. Foner, "What's An Agent, Anyway ? A sociological Case Study", Agents Memo 93-01, Agents Group, MIT Media Lab, 1993.

55. Y. Shoham, "Agent-Oriented Programming", Artificial Intelligence, vol. 60, 51-92, 1993.

56. M.R. Genesereth, "Software Agents", Comm. ACM, vol. 37(7), 48-53, July 1994.

57. N. R. Jennings, M. Wooldridge, "Applying Agent Technology", Applied Artificial Intelligence, vol. 9(4), 351-361, 1995.

58. N. R. Jennings, M. Wooldridge, "Software Agents", IEEE Review, vol. Jan 96, 17-20, 1996.

59. P. Maes, "Agents that Reduce Work and Information Overload", Comm. ACM, vol. 37(7), 31-40, July 1994.

60. P. Maes, "Artificial Life Meets Entertainment: Lifelike Autonomous Agents", Comm. ACM, vol. 38(11), 108-114, 1995.

61. O. Etzioni, D.S. Weld, "Intelligent Agents on the Internet: Fact, Fiction, and Forecast", IEEE Expert, vol. 10(4), 44-49, 1995.

62. A. Chavez, A. Moukas, P. Maes, "Challenger: A Multi-agent System for Distributed Allocation", Proc. of the International Conference on Autonomous Agents, Marina Del Ray, California, 1997.

63. A. Goscinski, "Distributed Operating Systems, The Logical Design", Addison Wesley, 1991.

64. D.L. Eager, E.D. Lazowska, and J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed Systems", IEEE Trans. Software Eng., vol. SE-12(5), 662-675, 1986.

65. J.M. Litzkow, M. Livny, and M.W. Mutka, "Condor: A Hunter for Idle Workstations", Proc. IEEE 8th Int'l Conf. on Distributed Computing Systems, IEEE CS Press, pp. 104-111, 1988.

66. P. Krueger, R. Chawla, "The Stealth Distributed Scheduler", in Proc. IEEE 11th Int'l Conf. On Distributed Computing Systems, IEEE CS Press, pp. 336-343, 1991.

67. N.G. Shivaratri, P. Krueger, M. Singhal, "Load Distributing for Locally Distributed Systems", IEEE Computer, vol. 25(12), 33-44, 1992.

68. D.F. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini, "Economic Models for Allocating Resources in Computing Systems", in Clearwater S. (editor), "Market Based Control of Distributed Systems", World Scientific Press, 1995.

69. P. Mehra, "Automated Learning of Load Balancing Strategies for a Distributed Computer System", PhD Thesis, Computer Science Department, University of Illinois at Urbana-Champaign, 1993.

70. M.P. Wellman, "Market-Oriented Programming: Some Early Lessons", in S. Clearwater (ed.), "Market-Based Control: A Paradigm for Distributed Resource Allocation", World Scientific Press, 1996.

71. H.R. Varian, "Microeconomic Analysis", 3rd Ed., W.W. Norton & Company, New York, 1992.

72. J.C. Pasquale, "Intelligent Decentralized Control in Large Distributed Computer Systems", PhD Thesis, University of California, Berkeley, April 1988.

73. J.F. Kurose J.F., R. Simha, "A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems", IEEE Trans. Computers, vol. 38(5), 705-717, 1989.

74. R. Davis, R. Smith, "Negotiation as a Metaphor for Distributed Problem Solving", Artificial Intelligence, vol. 20, 1983.

75. T. Sandholm, "An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations", 11th National Conference on Artificial Intelligence (AAAI-93), Washington DC, 256-252, 1993.

76. T. Sandholm, V. Lesser, "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework", 1st International Conference on Multiagent Systems (ICMAS-95), San Francisco, pp. 328-335, 1995.

77. T.W. Malone, R.E. Fikes, K.R. Grant, and M.T. Howard, "Enterprise: A Market-Like Task Scheduler for Distributed Computing Environments", in B. Huberman (ed.), "The Ecology of Computation", pp. 177-205, North Holland, Amsterdam, 1988.

78. D.F. Ferguson, "The Application of Microeconomics to the Design of Resource Allocation and Control Algorithms", PhD Thesis, Columbia University, 1989.

79. T. Berners-Lee, L. Masinter, M. McCahill (eds), "Universal Resource Locator, or URL", Network Working Group Request for Comments: 1738 (RFC1738), 1994.

80. R. Rivest, "The MD5 Message Digest Algorithm", Network Working Group Request for Comments: 1321, April 1992.

81. Rational Rose, Rational Software Co., Santa Clara, USA.

82. P. Giladi, N. Ahituv, "SPEC as a Performance Evaluation Measure", IEEE Computer, vol. 28(8), 33-42, 1995.

83. SPEC Benchamarks Organization, http://www.specbench.org

84. G. Booch, "Object Oriented Analysis and Design with Applications", Benjamic - Cummings Publishing, 1994.

85. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Elements of Reusable Object Oriented Software", Addison Wesley, 1994.

86. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, "Pattern Oriented Software Architecture: A System of Patterns", John Wiley & Sons Ltd., 1996

87. J. Lakos, "Large Scale C++ Software Design", Addison Wesley, 1996.

88. R.W. Stevens, "UNIX Network Programming", Prentice Hall, Englewood Cliffs, New Jersey, 1990.

89. B.W. Kerninghan, and R. Pike, "The UNIX Programming Environmnent", Prentice Hall, Englewood Cliffs, New Jersey, 1984.

90. B.W. Kerninghan, and D.M. Ritchie, "The C Programming Language", 2nd Ed. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

91. K. Arnold, J. Gosling, "The Java Programming Language", Addison Wesley, 1996.

92. B. Stroustrup, "The C++ Programming Language", Addison Wesley, 1986.

93. D.C. Schmidt, "The ADAPTIVE Communication Environment: An Object-Oriented Network Programming Toolkit for Developing Communication Software", 11th and 12th Sun Users Group Conference, December 1993 and June 1994.

94. D.C. Schmidt, "Using Design Patterns to Develop Reuseable Object-Oriented Software", Computing Surveys ACM, vol. 28A(4), December 1996.

95. D.C. Schmidt, "An OO Encapsulation of Lightweight OS Concurrency Mechanisms in the ACE Toolkit", Technical Report WUCS-95-31, Washington University, 1995.

96. D.C. Schmidt, "Reactor: An Object Behavioral Pattern for Event Demultiplexing and Event Handler Dispatching", Proceedings of the First Pattern Languages of Programs conference in Monticello, Illinois, August 1994. Also as a chapter in the book "Pattern

Languages of Program Design", editors Coplien J. and Schmidt C. D., Addison Wesley, 1995.

97. D.C. Schmidt, "Acceptor and Connector, Design Patterns for Initializing Communication Services", in proceedings of the EuroPLoP '96 conference, Kloster Irsee, Germany, July 10-14, 1996.

98. T.J. Mowbray, R. Zahavi, "The Essential CORBA: Systems Integration Using Distributed Objects ", John Wiley & Sons, New York, 1995.

99. Object Management Group, Inc., "Common Object Request Broker Architecture and Specification, CORBA", Revision 2, Framingham, MA, USA, 1995.

100. OMG, Object Management Group Inc., Framingham, USA, http://www.omg.com

101. I. Pyarali, T.H. Harrison and D.C. Schmidt, "Design and Performance of an Object-Oriented Framework for High-Speed Electronic Medical Imaging", Computing Systems, USENIX, vol. 9(4), 1996.

102. zApp, Rogue Wave, http://www.roguewave.com

103. Microsoft Windows, Microsoft Corporation, http://www.microsoft.com

104. OpenWindows, Sun Microsystems, http://www.sun.com

105. Oracle DBMS, Oracle Corporation, http://www.oracle.com

106. Sybase, Sybase Corporation, http://www.sybase.com

107. A. Etemadi, J-P. Schmidt, G. Matas, J. Illingworth, and J. Kittler, "Low-Level Grouping of Straight-Line Segments", Proceedings of the British Machine Vision Conference, 1991.

108. S.M. Smith, "Flexible Filter Neighborhood Designation", Proceedings of 13th Int. Conf. on Pattern Recognition, vol. 1, 206-212, 1996.

109. Κ. Γεωργιάδης, "Τμηματοποίηση του Αγγειακού Δέντρου του Αμφιβληστροειδούς Χιτώνα σε Ιατρικές Εικόνες", Διπλωματική Εργασία, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, Ηράκλειο, 1995.

110. K.J. Ousterhout, "Scripting: Higher Level Programming for the 21st Century", White Paper, May 1997, Sun Microsystems Laboratories
http://www.sunlabs.com/~ouster/scripting.html

111. K.J. Ousterhout, "Tcl and the Tk Toolkit", Addison Wesley, 1987.

112. A.J. Rees, W. Clinger (eds), "The Revised^3 Report on the Algorithmic Language Scheme", In ACM SIGPLAN Notices 21(12), p. 37-79, December 1986.

113. J.M. Tenenbaum, T.S. Chowdhry, K. Hughes, "Eco System: An Internet Commerce Architecture", IEEE Computer, vol. 30(5), 48-55, 1997.

114. A. Kambil, "Doing Business in the Wired World", IEEE Computer, vol. 30(5), 56-61, 1997.

115. Taligent Inc., "Building Object-Oriented Frameworks", Taligent White Paper, http://www.taligent.com/Technology/WhitePapers/BuildingFwks

116. B. Schneier, "Applied Cryptography 2nd Edition: Protocols, Algorithms and Source Code in C", JohnWiley and Sons, New York, 1996.

117. M.C. Little, D. L. McCue, “Construction and Use of Simulation Package in C++”, ftp://arjuna.ncl.ac.uk.

118. A.S. Tanenbaum, “Modern Operating Systems”, Prentice Hall, Englewood Cliffs, New Jersey, 1992