Π Ο Λ Υ Τ Ε Χ Ν Ε Ι Ο Κ Ρ Η Τ Η Σ
T E C H N I C A L U N I V E R S I T Y O F C R E T E

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ Η/Υ

Διπλωματική Εργασία

ΘΕΜΑ:

DISTRIBUTED NUMERICAL COMPUTATION IN UNRELIABLE ENVIRONMENT

Μαργαρίτης Γιάννης
Α.Μ.:1999030053

Εξεταστική επιτροπή:

κ. Σαμολαδάς Βασίλης *(επιβλέπων)*
κ. Πετράκης Ευριπίδης
κ. Πνευματικάτος Διονύσης

Περιεχόμενα

Περιεχόμενα I
Πίνακας Εικόνων III
Εισαγωγή 1
Κεφάλαιο 1 Αναφορά σε κατανεμημένα project 2
1.1Distributed.net 2
1.1.1Γενικά 2
1.1.2Η ιστορία του οργανισμού 2
1.1.3Σημερινά Project 4
1.1.4Οι στόχοι του οργανισμού 4
1.1.5Σημερινά στοιχεία 4
1.2 Seti@home 5
1.2.1Γενικά 5
1.2.2Στατιστικά στοιχεία 6
1.2.3Συμπέρασμα 6
1.3Folding@home 7
1.3.1Γενικά 7
1.3.2Συμμετοχή-Στατιστικά 8
Κεφάλαιο 2 Τεχνικά χαρακτηριστικά-Πλατφόρμες 11
2.1 Technical reference of distributed.net 11
2.2 Technical reference of seti@home 12
2.3 Technical reference of folding@home 16
2.3.1Network Architecture 16
2.3.1.1Servers 17
2.3.1.2Proxies 17
2.3.1.3Clients 17
2.3.2Ασφάλεια 18
2.4 Πλατφόρμες 19
2.4.1 Boinc 19
2.4.1.1 Projects and applications 19
2.4.1.2Files and data servers 21
2.4.1.3Trickle messages 22
2.4.1.4Ασφάλεια 23
2.4.1.5Scheduling server: policy 25
2.4.1.6Host identification 26
2.4.1.7Επικοινωνία με τον Worker 26
2.4.2Parabon computation 27
2.4.2.1 Frontier Components 27
2.4.2.2The Pioneer Compute Engine 27
2.4.2.3The Frontier Server 28
2.4.2.4The Frontier SDK 28
2.4.2.5Data and Executable Elements 29
2.4.2.6Checkpoints 29
2.4.2.7Rigid Security Model 30
2.4.2.8Securing Providers 30
2.4.2.9Securing Client Intellectual Property 30
2.4.3Peer to peer - JXTA 30
2.4.3.1Distribution of tasks amongst workers 32
2.4.3.2Result retrieval 33
2.4.3.3Reliability 33
2.4.3.4Scalability 35
Κεφάλαιο 3 Η δικιά μας δουλειά 38
3.1 User 39
3.2 To interface για τον user 40
3.3 Worker 41
3.4 To interface για τον worker 42

3.5 Server..... 43
3.6 Η δομή των εργασιών..... 43
3.7 Η δομή των αποτελεσμάτων 45
3.8 Distribution of tasks amongst workers..... 45
3.9 Result retrieval..... 49
3.10 Comnunication..... 50
3.11 Garbage Collection..... 50
Κεφάλαιο 4 Τα πειράματα..... 51
4.1 Γινόμενο πινάκων..... 51
4.1.1Συμπέρασμα..... 60
Βιβλιογραφία – Αναφορές..... 62

Πίνακας Εικόνων

Εικόνα 1 Η εικόνα παρέχεται από το χώρο www.thinkquest.org..... 8
Εικόνα 2 αριθμός ενεργών μονάδων υπολογισμών του folding@home.....9
Εικόνα 3 στατιστικά Client όσον αφορά το λειτουργικό συστημα, τελευταια ενημέρωση Tue, 12 Oct 2004 02:20:54 Συνολικός αριθμός εθελοντών = 382333..... 9
Εικόνα 4 η διανομή των δεδομένων. 12
Εικόνα 5 Η συλλογή και η ανάλυση των αποτελεσμάτων.....14
Εικόνα 6 the proxy rings..... 17
Εικόνα 7 τα συστατικά ενός project..... 19
Εικόνα 8 αναλυτικά η επικοινωνία με το worker.....26
Εικόνα 9 το frontier client-server μοντέλο..... 28
Εικόνα 10 Παράδειγμα ενός Code Repository που περιέχει τρεις κώδικες.....31
Εικόνα 11 ένας Worker node υιοθετεί το ρόλο του task dispatcher όταν ο τελευταίος αποτυγχάνει.....33
Εικόνα 12 επικοινωνίες μεταξύ monitor και άλλων κόμβων.....35
Εικόνα 13 δίκτυο από work groups και associated monitor groups..... 36
Εικόνα 14 Το client server μοντέλο μας.....37
Εικόνα 15 η μορφή εγγραφής των δεδομένων.....42
Εικόνα 16 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -1.....44
Εικόνα 17 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -2.....45
Εικόνα 18 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -3.....45
Εικόνα 19 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -4.....46
Εικόνα 20 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -5.....47
Εικόνα 21 το μέσο speedup για υπολογισμό, με σπασίματα πινάκων 1000χ1000,750χ750 και 500χ500, με δοσμένες εργασίες από 10 διαφορετικούς users ως προς τους workers..... 50
Εικόνα 22 οι μέσοι χρόνοι που χρειάστηκαν για να ολοκληρωθούν 10 σταλμένες εργασίες, για σπασίματα πινάκων 1000χ1000,750χ750 και 500χ500 ,από users ως προς τους workers.....51
Εικόνα 23 το speedup για υπολογισμό, με σπασίματα πινάκων 1000χ1000,750χ750 και 500χ500, με δοσμένες εργασίες από 1 user ως προς τους workers..... 52
Εικόνα 24 οι χρόνοι που χρειάστηκαν για να ολοκληρωθεί 1 σταλμένη εργασία από user,για σπασίματα πινάκων 1000χ1000,750χ750 και 500χ500 , ως προς τους workers.....53
Εικόνα 25 το speedup για υπολογισμό, με σπάσιμο πινάκων 1000χ1000, με δοσμένες εργασίες από 1 user ως προς τους workers και πιθανότητα μη επιστροφής αποτελεσμάτων από τους workers 5% έως 20%..... 55
Εικόνα 26 ο χρόνος για υπολογισμό, με σπάσιμο πινάκων 1000χ1000, με δοσμένες εργασίες από 1 user ως προς τους workers και πιθανότητα μη επιστροφής αποτελεσμάτων από τους workers 5% έως 20%..... 55
Εικόνα 27 το speedup για υπολογισμό, με σπάσιμο πινάκων 500χ500, με δοσμένες εργασίες από 1 user ως προς τους workers και πιθανότητα μη επιστροφής αποτελεσμάτων από τους workers 5% έως 20%... 57
Εικόνα 28 ο χρόνος για υπολογισμό, με σπάσιμο πινάκων 500χ500, με δοσμένες εργασίες από 1 user ως προς τους workers και πιθανότητα μη επιστροφής αποτελεσμάτων από τους workers 5% έως 20%... 57

Εισαγωγή

Ο διανεμημένος υπολογισμός(distributed computing) είναι μια επιστήμη που λύνει ένα μεγάλο πρόβλημα με τη διάθεση μικρών μερών του προβλήματος σε πολλούς υπολογιστές για να λυθούν και έπειτα το συνδυασμό των λύσεων των μερών σε μια λύση για το πρόβλημα. Μερικά από τα πρόσφατα διανεμημένα προγράμματα υπολογισμού έχουν ως σκοπό να χρησιμοποιήσουν τους υπολογιστές των εκατοντάδων χιλιάδων εθελοντών σε όλο τον κόσμο, μέσω του διαδικτύου, να ψάξουν τα εξωγήινα ράδιο σήματα, να ψάξουν τους prime αριθμούς ,τόσο μεγάλους που έχουν περισσότερα από δέκα εκατομμύρια ψηφία, και να βρούνε τα αποτελεσματικότερα φάρμακα για να παλέψουν τον ιό του AIDS. Αυτά τα προγράμματα είναι τόσο μεγάλα, και απαιτούν τόση πολλή δύναμη υπολογισμού να λύσουν, που θα ήταν αδύνατο για οποιοδήποτε υπολογιστή ή πρόσωπο να τα λύσουν σε ένα λογικό χρονικό διάστημα.

Κεφάλαιο 1 Αναφορά σε κατανεμημένα project

1.1 Distributed.net

1.1.1 Γενικά

Το Distributed.net είναι ένας μη κερδοσκοπικός οργανισμός που έχει αναλάβει την υποχρέωση να λειτουργεί ως ένας τόπος συλλογής θεμάτων σχετικών με το distributed computing ή την διαδικασία με την οποία μεγάλος αριθμός από υπολογιστές μετέχουν όλοι μαζί στην λύση προβλημάτων που αποτελούν πρόκληση(challenge). Η ιστορία του οργανισμού, τα προβλήματα που μελετώνται από τον οργανισμό, τα μεμονωμένα άτομα που παίρνουν μέρος στον οργανισμό και οι βραχυπρόθεσμοι καθώς και οι μακροπρόθεσμοι στόχοι του οργανισμού, όλα σχετίζονται με το να βρεθούν καινούργιοι τρόποι για τους υπολογιστές που συνδέονται στο internet ώστε να χρησιμοποιούνται κατά την διάρκεια που μένουν αδρανείς. Αυτό είναι δυνατόν να επιτευχθεί με τη δημιουργία λογισμικού το οποίο επιτρέπει στους υπολογιστές, οι οποίοι δεν βρίσκονται σε χρήση, να επικοινωνούν μέσω του internet επιτρέποντας ένα απροσδιόριστο αριθμό από υπολογιστές να δουλεύουν για κάποιο κοινό σκοπό. Σήμερα το distributed.net έχει χρησιμοποιήσει την τεχνολογία του στη αμφισβήτηση αλγόριθμων κωδικοποίησης στο internet. Μέσω αυτής της διαδικασίας σκέψης κατάφερε να δημιουργήσει και να φιλτράρει όλες αυτές τις τεχνικές, βελτιώνοντας την εμβέλεια δράσης, το εύρος των γνώσεων και την ποικιλία των εργασιών που είναι κατάλληλες για αυτή την τεχνολογία.

1.1.2 Η ιστορία του οργανισμού

Σε απάντηση στο RC5-32/12/7 (56 bit) Secret Key Challenge, του RSA Lab's 56 bit τεχνολογία αλγορίθμου κωδικοποίησης, μια ομάδα μεμονωμένων ατόμων ξεκίνησε την ανάπτυξη εργαλείων λογισμικού τα οποία θα λειτουργούσαν για την επίλυση αυτής της πρόκλησης. Δημιουργήθηκε ένα πρόγραμμα – υπηρέτης, ο οποίος εγκαταστάθηκε σε πολλούς υπολογιστές και εκτελούσε τους περίπλοκους υπολογισμούς που ήταν απαραίτητοι για να λυθεί αυτή η πρόκληση. Επιπροσθέτως, σχεδιάστηκε και δημιουργήθηκε ένα δίκτυο από εξυπηρετητές, το οποίο μπορούσε να συντονίσει

όλους τους υπολογιστές – υπηρέτες. Το μεγάλο πρόβλημα του ελέγχου 72 επτάκις εκατομμυρίων κλειδιών έσπασε και διανεμήθηκε σε κάθε υπολογιστή - υπηρέτη. καθώς κάθε υπηρέτης ολοκλήρωνε το κομμάτι της εργασίας που του είχε ανατεθεί έπρεπε να αναφέρει στον εξυπηρετητή τα αποτελέσματα ώστε να του ανατεθεί καινούργια εργασία. Με αυτόν τον τρόπο της οργανωμένης συνεργασίας, πολλοί μικροί (σε υπολογιστική δύναμη) υπολογιστές μπορούσαν να φτάσουν, ακόμα και να ξεπεράσουν, την υπολογιστική δύναμη πολλών κεντρικών μηχανημάτων. Στις 8 Μαΐου 1997 αυτή η προσπάθεια ονομάστηκε distributed.net με τον Adam L. Beberg ως ιδρυτή και διευθυντή αυτού του μη κερδοσκοπικού οργανισμού .Στις 8 Ιουλίου 1997 μια καινούργια έκδοση του προγράμματος υπηρέτη έγινε διαθέσιμη. Σε αυτή τη δεύτερη έκδοση βελτιώθηκε και έγινε πιο γρήγορη και πιο απλή η διάθεση και η επεξεργασία των δεδομένων καθώς και πιο προσαρμόσιμη η λειτουργία.

Στις 22 Οκτωβρίου 1997 μετά από 212 ημέρες εργασίας η πρόκληση του RC5-56 λύθηκε. Στο τέλος του διαγωνισμού 4000 ενεργείς ομάδες εθελοντών (με συνολική επεξεργασία περισσότερο από 7 δισεκατομμύρια κλειδιά κάθε δευτερόλεπτο) με μια συνδυασμένη υπολογιστική δύναμη περίπου ίση και περισσότερη από 26 χιλιάδες προσωπικούς υπολογιστές ,κατάφερε να αξιολογήσει ποσοστό 46% των πιθανών λύσεων. ένας υπολογιστής, ο οποίος ανήκε στον Jo Hermans βρήκε την λύση. από το βραβείο των $10,000, $8,000 έγιναν δωρεά στο Project Gutenberg/CMU, $1,000 δόθηκαν στον Jo Hermans και $1,000 παρέμειναν στο distributed.net για να καλυφθούν τα έξοδα του.

Μετά από λίγο χρόνο επανακατασκευής και ανάπτυξης , ένα δεύτερο project ξεκίνησε στις 13 Ιανουαρίου του 1998.Ο δεύτερος διαγωνισμός κωδικοποίησης DES II-1 πήρε μόνο 40 μέρες για να ολοκληρωθεί. DES II-1 αποκωδικοποιήθηκε στις 23 Φεβρουαρίου 1998. Η επιτυχής ολοκλήρωση αυτής της πρόκλησης επέφερε ένα βραβείο $5,000 από τα οποία $3,000 δόθηκαν στο Free Software Foundation ένα ακόμα μη κερδοσκοπικό εγχείρημα.

Στις 18 Ιανουαρίου 1999 στις 9π.μ άρχισε το DES III.Το Distributed.net με τη βοήθεια του EFF's Deep Crack και των υπηρετών του distributed.net έλαβαν μέρος και ολοκλήρωσαν αυτή τη πρόκληση στις 19 Ιανουαρίου του 1999, δηλαδή σε λιγότερο από 24 ώρες από τη στιγμή που ξεκίνησε αυτή η πρόκληση.

Στις 17 Νοεμβρίου 1999 τα μεσάνυχτα το distributed.net ξεκίνησε να συμμετέχει στο CSC.Το CSC είναι πρόκληση κωδικοποίησης που διοργανώθηκε από το CS Communications and Systems για να αποδείξει πόσο ασθενές είναι ένα κλειδί 56-bit απέναντι σε κακόβουλες επιθέσεις. Το distributed.net ήταν επίσης επιτυχής και σε αυτή τη πρόκληση και στις 16 Ιανουαρίου 2000 στις 6.30 μ.μ. το κλειδί βρέθηκε.

Στις 14 Ιουλίου 2002 μετά από 1.757 μέρες και 58,747,597,657 κομμάτια εργασίας που εξετάστηκαν η RC5-64 πρόκληση λύθηκε όταν ένα P3-450 το οποίο έτρεχε Windows 2000 στο Τόκιο επέστρεψε το κλειδί σε έναν από τους εξυπηρετητές του distributed.net .Το εγχείρημα ολοκληρώθηκε με 331,252 συμμετέχοντες .Ο μεγαλύτερος ρυθμός των 270,147,024kkeys είναι παρόμοιο με 32,504 800MHz Apple PowerBook G4 laptops ή 45,998 2GHz AMD Athlon XP ή περίπου μισό εκατομμύριο Pentium Pro 200s.

1.1.3 Σημερινά Project

Από τις 3 Δεκεμβρίου 2002 το distributed.net εργάζεται στο RSA Labs' 72-bit secret-key, project RC5-72. Το μέγεθος τον υποτιθέμενων κλειδιών σε αυτό το project είναι 256 φορές το μέγεθος των υποτιθέμενων κλειδιών στο RC5-64, κάτι το οποίο κάνει αυτό να είναι το μεγαλύτερο εγχείρημα στην ιστορία του distributed.net.

Περισσότερες πληροφορίες για αυτό το project στο http://www.distributed.net/rc5/.
Το distributed.net επίσης εργάζεται στο OGR-24 και OGR-25 (Optimal 24-mark and 25-mark Golomb Rulers). Περισσότερες πληροφορίες για το project στο http://www.distributed.net/ogr/.

1.1.4 Οι στόχοι του οργανισμού

Οι βραχυπρόθεσμοι στόχοι του distributed.net περιστρέφονται γύρω από την υφιστάμενη κατάσταση των project που ήδη γίνονται αυτή τη στιγμή .Το Distributed.net το απασχολεί αρχικά η ολοκλήρωση των διαγωνισμών και η ολοκλήρωση μιας τρίτης έκδοσης του υπηρέτη λογισμικού που χρησιμοποιείται από τα αδρανή υπολογιστικά συστήματα.

Κάθε μακροπρόθεσμος στόχος είναι καθαρά ακαδημαϊκός εκ φύσεως .Το Distributed.net είναι αυτοδεσμευμένο στη έρευνα των δυνατοτήτων της κατανεμημένης υπολογιστικής και των εφαρμογών της .Όλες οι προσπάθειες είναι μη κερδοσκοπικές .Οι καινοτομίες και οι πρόοδοι θα γίνονται γνωστές στο κοινό οποτεδήποτε δυνατόν.

1.1.5 Σημερινά στοιχεία

Η συμμετοχή των εθελοντών αυτή τη στιγμή υπολογίζεται περίπου στους 60,000 ιδιώτες από σχεδόν κάθε έθνος και περιοχή ανά το κόσμο .Με συνδυαζόμενη υπολογιστική δύναμη περίπου 500,000 υπολογιστών ,το distributed.net θεωρείται σήμερα η πρώτη μεγάλης κλίμακας συνεργασία υπολογιστικής προσπάθειας που έγινε ποτέ.

1.2 Seti@home

1.2.1Γενικά

Το πρόγραμμα Seti@home είχε ως σκοπό την έρευνα σημάτων που προέρχονταν από το διάστημα για την εξεύρεση κάποιων σημάτων που θα υποδηλώνουν την ύπαρξη νοήμονος εξωγήινης ζωής.

Το μεγαλύτερο πρόβλημα ήταν το μεγάλο πόσο υπολογιστικής δύναμης .Εάν ο υπολογιστής έψαχνε εξονυχιστικά για ασθενή σήματα ή για κάποιο τύπο σημάτων τα οποία μπορούσαν να καταχωρηθούν σε μια κοινή ομάδα ,η εργασία αυτή θα απαιτούσε μεγάλο αριθμό δεδομένων για ανάλυση .Το πρόγραμμα SETI δεν θα μπορούσε ποτέ να παράγει ή να αγοράσει αυτή την υπολογιστική δύναμη αντίθετα όμως εκείνο που μπόρεσε να κάνει είναι, αντί να χρησιμοποιήσει ένα υπολογιστικό σύστημα με τεράστια υπολογιστική δύναμη, να χρησιμοποιήσει ένα κανονικό υπολογιστή, ο οποίος ναι μεν θα χρειαζόταν πολύ περισσότερο χρόνο για να επεξεργαστεί τα δεδομένα, όμως εάν αυτό γινόταν με πολλούς "κανονικούς" υπολογιστές οι οποίοι θα δούλευαν ταυτόχρονα αλλά σε διαφορετικό κομμάτι των υπολογισμών, τότε τελικά θα είχε λυθεί το πρόβλημα .Μάλιστα η εργασία αυτή της επεξεργασίας των σημάτων μπορούσε εύκολα να διασπαστεί σε μικρά κομμάτια ,τα οποία θα δούλευαν ξεχωριστά και παράλληλα καθώς κανένα από αυτά τα κομμάτια δεν εξαρτάται από το άλλο.

Πως όμως θα μπορούσαν να βρεθούν τόσοι πολλοί υπολογιστές; Η ομάδα UC Berkeley SETI σκέφτηκε πως αυτός ο μεγάλος αριθμός υπολογιστών μπορούσε να βρεθεί αρκεί να μπορούσαν να "δανειστούν" όλους εκείνους τους υπολογιστές των απλών χρηστών την στιγμή που οι χρηστές δεν θα τους χρησιμοποιούσαν .Έτσι γεννήθηκε και το SETI@home project που η ιδέα λειτουργίας του ήταν ότι δεδομένα από αστρονομικές μετρήσεις θα διανέμονται σε πολλούς υπολογιστές (workers) για επεξεργασία και αφού ολοκληρωθεί η επεξεργασία τα αποτελέσματα που παρήχθησαν επιστρέφονται πίσω στον υπηρέτη (server) από τον οποίο δόθηκαν τα αρχικά δεδομένα για την επεξεργασία .Όλη η επεξεργασία των δεδομένων γίνεται από ένα πρόγραμμα (agent) το οποίο όμως έχει τη μορφή screen saver και "τρέχει" μόνο όταν ο χρηστής δεν χρειάζεται τον υπολογιστή του και στη περίπτωση που ο χρηστής θελήσει να χρησιμοποιήσει τον υπολογιστή του, τότε η λειτουργία του αναστέλλεται αυτόματα και επανενεργοποιείται μόνο όταν ο χρηστής τελειώσει την εργασία του .Αυτήν την περίοδο χρησιμοποιούν ένα όργανο καταγραφής στοιχείων που καταγράφει μόνο 2.5MHz από το 100MHz που παραλαμβάνεται από το SERENDIP. Ένα δεύτερο όργανο καταγραφής στοιχείων θα διπλασίαζε το καταγεγραμμένο εύρος ζώνης και το ποσό στοιχείων που αναλύουν.

1.2.2Στατιστικά στοιχεία

Από τις 23 Οκτωβρίου 2000, 2.438.045 εθελοντές είχαν τρέξει το πρόγραμμα SETI@home. Από εκείνους, 519.725 έτρεχαν ενεργά το πρόγραμμα και είχαν επιστρέψει ένα αποτέλεσμα στις προηγούμενες δύο εβδομάδες. Αυτοί οι εθελοντές είχαν δώσει συνολικά 437.000 έτη χρόνων ΚΜΕ. Αυτήν την περίοδο, το μέσο ποσοστό επεξεργασίας υπολογιστών που τρέχουν SETI@home είναι 15,7 Tflops -που υπολογίζεται κατά μέσο όρο – ενώ από την έναρξη του προγράμματος το ποσοστό επεξεργασίας είναι 9,5 Tflops. από ό,τι ξέρουμε, το SETI@home είναι το μεγαλύτερο διανεμημένο πρόγραμμα υπολογισμού. Θα μπορούσε επίσης να θεωρηθεί ο μεγαλύτερος υπολογισμός ανέκαθεν διενεργηθείς.

Τα 1,1 δισεκατομμύριο σήματα στη βάση δεδομένων SETI@home εξετάζονται με τις τεχνικές που έχουμε περιγράψει .Το ποσοστό στο οποίο εξετάζονται αυτήν την περίοδο τα σήματα είναι χαμηλότερο από το ποσοστό στο οποίο τα νέα σήματα προστίθενται στη βάση δεδομένων, και έτσι έχουν ερευνήσει λεπτομερώς μόνο ένα αρκετά μικρό μέρος των πιθανών σημάτων. Θα προσθέσουν σύντομα ένα άλλο συγκρότημα ηλεκτρονικών υπολογιστών στην οργάνωση κεντρικών υπολογιστών τους για να επιταχύνουν αυτήν την επεξεργασία κατά μήκος .Ελπίζουν να εξετάσουν τα σήματα στον πραγματικό χρόνο πριν από την έλευση μεγάλου χρονικού διαστήματος. Μέχρι τώρα, κανένα από τα σήματα που εξετάζονται δεν έχει παρουσιάσει στοιχεία εξωγήινης νοημοσύνης.

1.2.3 Συμπέρασμα

Το SETI@home ήταν αρχικά εγκεκριμένο να επεξεργαστεί για δύο έτη τα στοιχεία από το τηλεσκόπιο Arecibo. Η ισχυρή δημόσια απάντηση και οι νέες βελτιώσεις στο λογισμικό πελατών έχουν προτρέψει να επεκταθεί η έρευνα.

Το SETI@home αυτήν την περίοδο ερευνάει δείγματα μόνο μιας μικρής μερίδας του ράδιο φάσματος και μιας μικρής μερίδας του ουρανού. γίνεται προσπάθεια για επέκταση της κάλυψης του ουρανού και διεύρυνσης του εύρος ζώνης συχνότητας .Το SETI@home II, αυτήν την περίοδο, ελπίζει να κάνει και τα δύο.

Για την καλύτερη κάλυψη του ουρανού επρόκειτο να προσθέσουν ένα σύστημα οργάνων καταγραφής SETI@home σε ένα ράδιο τηλεσκόπιο νοτίου ημισφαιρίου. Αυτό θα αύξανε την κάλυψη του ουρανού από περίπου 25% σε 75%. Συζητούν αυτήν την περίοδο τη δυνατότητα αυτή με ένα νότιο παρατηρητήριο.

Το σύστημα καταγραφής περιορίζει αυτήν την περίοδο το εύρος ζώνης συχνότητάς τους. Με την αναπαραγωγή του συστήματος καταγραφής, θα μπορούσαν να διπλασιάσουν την κάλυψη εύρους ζώνης SETI@home's (και, φυσικά, το ποσοστό στοιχείων του).

Όπως σε οποιαδήποτε εθελοντική οργάνωση, είναι σημαντικό ότι το SETI@home ανταποκρίνεται στις επιθυμίες των εθελοντών του, επειδή η επιτυχία του προγράμματος εξαρτάται εξ ολοκλήρου από τους εθελοντές που παρέχουν τους πόρους υπολογισμού .Γι' αυτό κρατούν τους εθελοντές ενήμερους για την κάθε πρόοδό και μοιράζονται με αυτούς την επιστήμη πίσω από το SETI.Εργάζονται επίσης για να παρουσιάσουν στους εθελοντές πώς έχουν συμβάλει χωριστά στο πρόγραμμα με την παροχή των πληροφοριών για τα πιθανά σήματα που έχουν ανιχνεύσει και τις περιοχές του ουρανού που έχουν ανιχνεύσει.

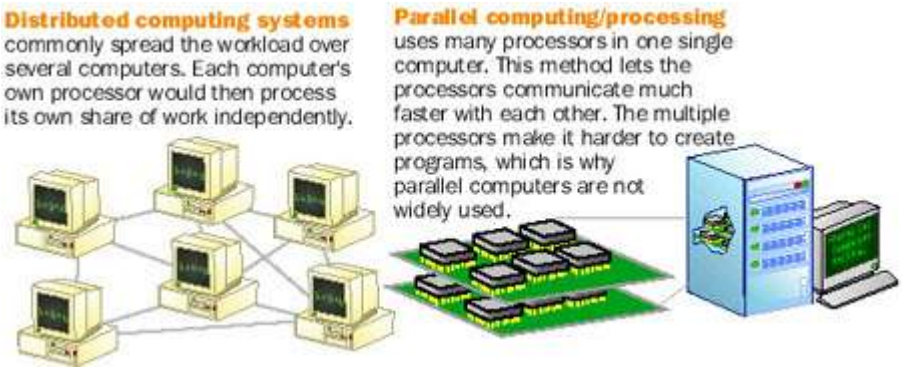
1.3 Folding@home

1.3.1 Γενικά

Σήμερα έχουμε τις υπολογιστικές μεθόδους για να μιμηθούμε το πρωτεϊνικό δίπλωμα. Εντούτοις, οι τρέχουσες ταχύτητες υπολογιστών (ακόμη και οι υπερυπολογιστές) επιτρέπουν σε μας να μιμηθούμε μόνο ένα νανοδευτερόλεπτο (10-9) αυτής της διαδικασίας, όπου η διαδικασία του πρωτεϊνικού διπλώματος μπορεί να πάρει περισσότερο από μικροδευτερόλεπτο (10-6). Κατά συνέπεια υπάρχει χάσμα χιλίων πτυχών μεταξύ της τρέχουσας ταχύτητας υπολογισμού και της δύναμης υπολογισμού που απαιτούνται για να μιμηθούμε το πρωτεϊνικό δίπλωμα.

To Folding@home έχει αναπτύξει έναν νέο τρόπο να μιμηθεί το πρωτεϊνικό δίπλωμα που μπορεί να σπάσει το εμπόδιο μικροδευτερολέπτου με τη διαίρεση της εργασίας μεταξύ των πολλαπλάσιων επεξεργαστών. Με τη χρησιμοποίηση μιας νέας προσέγγισης οι πελάτες και ο κεντρικός υπολογιστής του Folding@home επιτυγχάνουν μια κοντινή γραμμική ταχύτητα. Το γεγονός ότι ο αριθμός των επεξεργαστών αυξάνεται κάνει την ταχύτητα με την οποία διπλώνουμε τις πρωτεΐνες να αυξάνεται και εκείνη. Κατά συνέπεια, με χιλιάδες επεξεργαστές, μπορούμε να σπάσουμε το εμπόδιο μικροδευτερολέπτου και να ξεκλειδώσουμε το μυστήριο για το πώς οι πρωτεΐνες διπλώνουν.

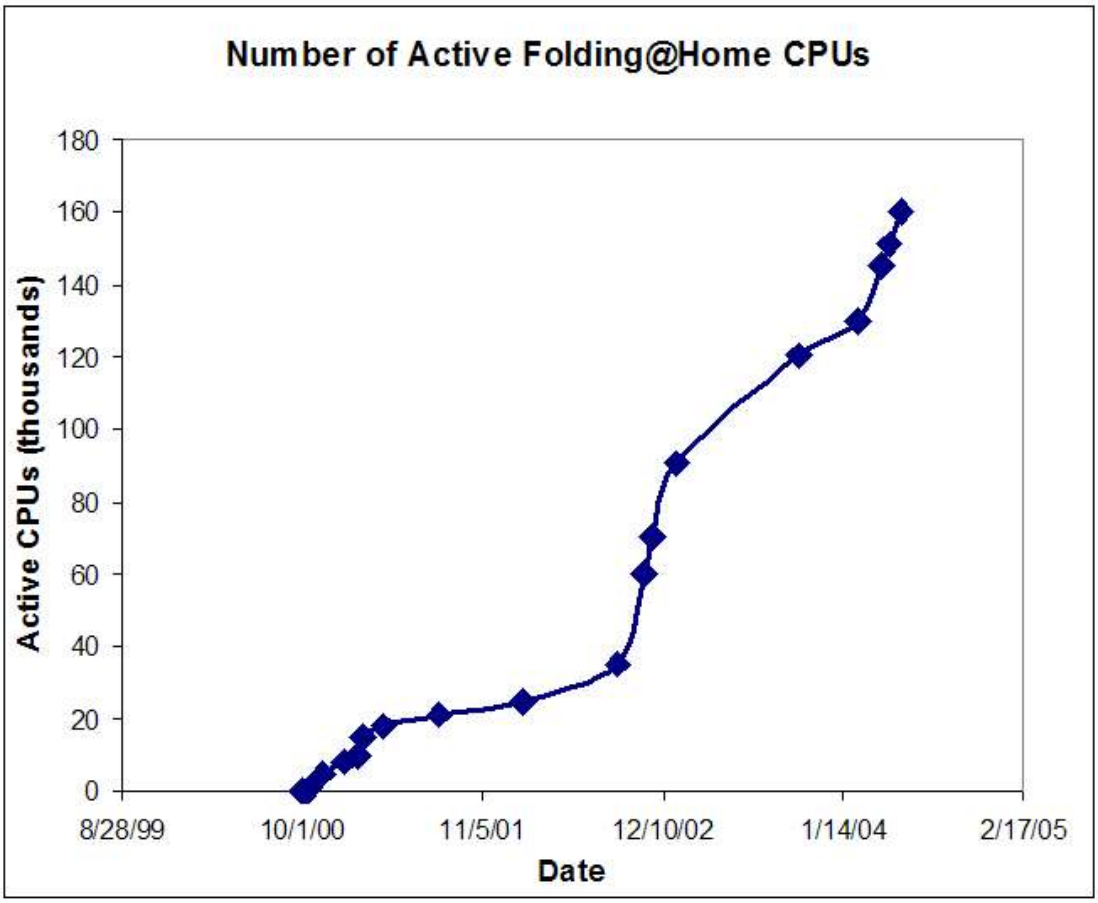
Η ομάδα Pande σπάζει τους συνολικούς υπολογισμούς στα μικρά «χοντρά» κομμάτια αποκαλούμενα μονάδες εργασίας .Όταν ο υπολογιστής δεν απασχολείται κάνει τους υπολογισμούς στο υπόβαθρο . Όταν η μονάδα εργασίας ολοκληρώνεται και η σύνδεση με το διαδίκτυο καθιερώνεται τα αποτελέσματα στέλνονται σε μια βάση δεδομένων στους κεντρικούς υπολογιστές του Στάνφορντ. Οι ερευνητές μπορούν να έχουν πρόσβαση σε αυτές τις βάσεις δεδομένων και να εξαγάγουν τις πολύτιμες πληροφορίες. Δείτε επίσης research paper summary.



Εικόνα 1 Η εικόνα παρέχεται από το χώρο **www.thinkquest.org**

1.3.2 Συμμετοχή-Στατιστικά

Η συμμετοχή φαίνεται μέσα από αυτό το διάγραμμα.



Εικόνα 2 αριθμός ενεργών μονάδων υπολογισμών του folding@home

Τα σημεία είναι κύρια σημεία που καταγράφονται στο τμήμα ειδήσεων. Η γραμμή προορίζεται να καθοδηγήσει το μάτι -- έχουν υπάρξει διακυμάνσεις στον αριθμό ενεργών CPUs σε κύρια σημεία και δεν παρουσιάζονται.

OS Type	Current TFLOPS	Active CPUs	Total CPUs
Windows	176.756	147297	898293
Mac OS X	7.515	9394	63404
Linux	13.151	15555	97595
Other	0.0	0	0
Total	197.422	172246	1059292

Εικόνα 3 στατιστικά Client όσον αφορά το λειτουργικό συστημα, τελευταια ενημέρωση Tue, 12 Oct 2004 02:20:54 **Συνολικός αριθμός εθελοντών = 382333**

Σημαντικό είναι να αναφέρουμε πως τα αποτελέσματα από το project,θα διατεθούν στο κοινό και δεν είναι εμπορεύσιμα με κανένα τρόπο καθώς το Folding@home τρέχει σε ένα ακαδημαϊκό ίδρυμα (ειδικότερα το Pande Group, at Stanford University's Chemistry Department), το οποίο είναι ένα μη κερδοσκοπικό ίδρυμα αφιερωμένο στην επιστημονική έρευνα και την εκπαίδευση.

Κεφάλαιο 2 Τεχνικά χαρακτηριστικά-Πλατφόρμες

2.1 Technical reference of distributed.net

Το distributed.net βασίζεται στους εθελοντές για να εξασφαλίσει την λειτουργία του .Η φιλοσοφία της λειτουργίας του βασίζεται σε client-server αρχιτεκτονική .Το μεγαλύτερο μέρος του κώδικα που χρησιμοποιεί είναι ανοιχτό στον κόσμο και μπορεί ακόμα και να το κατεβάσει Έτσι ώστε να το μελετήσει .Εκούσια παραμένουν κάποια κομμάτια του κώδικα κλειστά (όπως το πως χειρίζεται τα αρχεία των αποτελεσμάτων αλλά και των δεδομένων και ο κώδικας που αφορά το δίκτυο)Έτσι ώστε να εξασφαλιστεί η σωστή λειτουργία των project καθώς είναι αρκετές οι επιθέσεις διαφόρων μορφών που έχουν γίνει κατά καιρούς και έχουν ως στόχο να πλήξουν την αξιοπιστία των project αλλά και του οργανισμού γενικότερα.

Αυτή τη στιγμή είναι διαθέσιμοι στο κοινό οι κώδικες για το client(το κομμάτι εκείνο που υλοποιεί τον αλγόριθμο για τους υπολογισμούς) γραμμένο σε visual c++ , sql κώδικας αλλά και κώδικας επεξεργασίας που χρησιμοποιείται από τον server που είναι υπεύθυνος για τα στατιστικά και τέλος ο κώδικας PHP που παράγει το μεγαλύτερο μέρος των ιστοσελίδων .Οι κώδικες είναι διαθέσιμοι ακόμα και on line στην ιστοσελίδα του distributed.net.

Οι εθελοντές εκτός από τον τομέα των υπολογισμών είναι στην κρίση τους να βοηθήσουν των οργανισμό ακόμα και παρέχοντας βελτιώσεις στον υπάρχον κώδικα. Οι χρηστές εθελοντές μπορούν να κατεβάσουν ένα πρόγραμμα από την ιστοσελίδα του distributed.net το οποίο ουσιαστικά θα είναι εκείνο το οποίο θα τους καταστήσει εθελοντές για το project στο οποίο θα επιλέξουν να συμβάλουν . αφού δώσουν το e-mail τους ,έτσι ώστε να μπορέσουν να λάβουν διάφορες σημαντικές πληροφορίες όπως για το κωδικό τους και ακόμα και να ειδοποιηθούν σε περίπτωση που ο υπολογιστής τους καταφέρει να βρει την λύση στο πρόβλημα στο οποίο είναι εθελοντές ,είναι έτοιμοι να ξεκινήσουν τους υπολογισμούς .αφού ολοκληρωθεί ο υπολογισμός ο client με το server όπου δίνει το αποτέλεσμα αλλά και του αποδίδεται το αντίστοιχο credit για τους υπολογισμούς τους οποίους έκανε. Εκτός από τον κανονικό τρόπο επικοινωνίας μεταξύ worker και server μπορεί κάποιος να στείλει το αποτέλεσμα του υπολογισμού στον server με την αποστολή email απλά επισυνάπτοντας στο email το αρχείο που περιέχει το αποτέλεσμα του υπολογισμού .εκτός από το πρόγραμμα του client ο εθελοντής μπορεί να κατεβάσει ακόμα και το προσωπικό του proxy ο οποίος θα του παρέχει μεγαλύτερο κομμάτι εργασιών σε περίπτωση που δεν είναι δυνατή η συχνή του επικοινωνία με τους κεντρικούς proxies του distributed.net.

Κάθε φορά ο χρηστής μπορεί να δει την προσωπική του σελίδα με τα στατιστικά έτσι ώστε να γνωρίζει ανά πάσα στιγμή πόσο έχει συμβάλει σε υπολογιστική ισχύ. Για την ασφάλεια το distributed.net

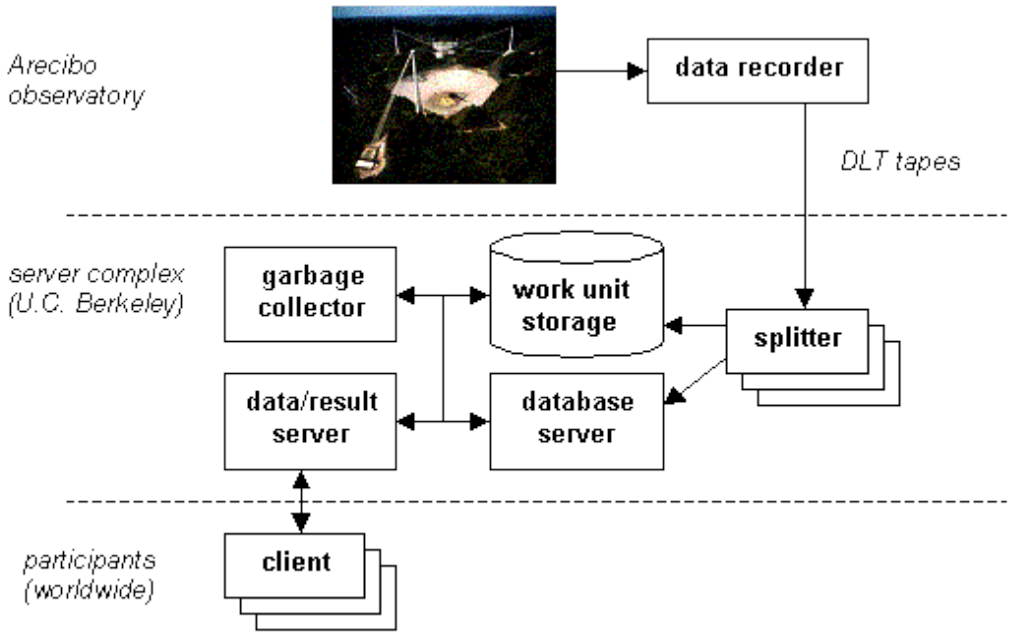
χρησιμοποιεί κάποιες δικλίδες ασφαλείας όπως τον κατά καιρούς επανυπολογισμό αποτελεσμάτων ο οποίος σε περίπτωση που αποδείξει ότι κάποιος χρηστής έχει παραδώσει παραποιημένα αποτελέσματα αυτόματα ακυρώνονται και όλα τα υπόλοιπα αποτελέσματα τα οποία έχει παραδώσει εκείνος ο χρηστής στο παρελθόν .Τέλος χρησιμοποιούνται ακόμα και κρυπτογραφικές μέθοδοι υπογραφής των δεδομένων που στέλνονται Έτσι ώστε να μπορέσει να ανιχνευτεί ότι χρησιμοποιήθηκαν τα δεδομένα που στάλθηκαν και ότι τα αποτελέσματα που στέλνονται είναι αυθεντικά.

2.2 Technical reference of seti@home

Το SETI@home's υπολογιστικό πρότυπο είναι απλό. Τα δεδομένα του σήματος διαιρούνται σε προκαθορισμένου μεγέθους **work units** που μπορούν να διανεμηθούν, μέσω του διαδικτύου, σε ένα πρόγραμμα πελατών που τρέχει σε πολυάριθμους υπολογιστές. Το πρόγραμμα πελατών υπολογίζει ένα αποτέλεσμα (ένα σύνολο υποψηφίων σημάτων), το επιστρέφει στον κεντρικό υπολογιστή, και παίρνει άλλο work unit. Δεν υπάρχει καμία επικοινωνία μεταξύ των πελατών.

Κάθε work unit επεξεργάζεται πολλές φορές από διαφορετικούς πελάτες(clients). Αυτό μας επιτρέπει να ανιχνεύσουμε και να απορρίψουμε τα αποτελέσματα από ελαττωματικούς επεξεργαστές και από κακόβουλους χρήστες. Μια επανάληψη υπολογισμού δύο έως τρεις φορές αρκεί για αυτόν το λόγο. Παράγουμε work units με ένα καθορισμένο ρυθμό και ποτέ δεν διώχνουμε έναν πελάτη που ζητά εργασία, Έτσι οι επαναλήψεις των ήδη υπολογισμένων work units αυξάνονται ανάλογα με τον αριθμό των πελατών. Αυτές οι ποσότητες έχουν αυξηθεί πολύ κατά τη διάρκεια της ζωής του project. Έχουμε κρατήσει τον αριθμό των επαναλαμβανόμενων υπολογισμών μέσα σε ένα επιθυμητό όριο με την ανάθεση στο πελάτη να κάνει περισσότερο υπολογισμό ανά work unit.

Η εργασία της δημιουργίας και τη διανομής των work units γίνεται από έναν κεντρικό υπολογιστή τοποθετημένο στο εργαστήριό τους .



Εικόνα 4 η διανομή των δεδομένων**.**

Χρησιμοποιούν μια σχεσιακή βάση δεδομένων για να αποθηκεύσουν τις πληροφορίες για τα workunits, results, users και άλλες πτυχές του προγράμματος. Ανέπτυξαν ένα multithreaded **data/result server** για να διανείμει work units στους πελάτες. Χρησιμοποιεί ένα HTTP- βασισμένο πρωτόκολλο έτσι ώστε πελάτες μέσα από firewalls να μπορούν να επικοινωνούν μαζί του Μέσω sockets. Οι Work units στέλνονται με την least-recent-send σειρά.

Ένα **garbage collector** πρόγραμμα αφαιρεί work units από το δίσκο, απενεργοποιώντας ένα on-disk flag στα αρχεία βάσεων δεδομένων τους. Έχουν πειραματιστεί με δύο πολιτικές:

- Διαγραφή work units για τις όποιες *N* αποτελέσματα έχουν παραληφθεί, όπου *N* είναι ο αριθμός των επανυπολογισμών. Εάν συμπληρωθεί ο χώρος αποθήκευσης work unit, η παραγωγή work unit εμποδίζεται και το throughput του συστήματος πέφτει.
- Διαγραφή work units που έχουν σταλεί Μ φορές, όπου *το M* είναι ελαφρώς μεγαλύτερο από *το N*. Αυτό αποβάλλει τα ανωτέρω bottleneck, αλλά μπορεί να οδηγήσει ορισμένα work units να μην έχουν ποτέ αποτελέσματα. Αυτό το ποσοστό μπορεί να γίνει αυθαίρετα μικρό με την αύξηση του *M*. Χρησιμοποιούμε αυτήν την περίοδο αυτήν την πολιτική.

Η διατήρηση της συνεχούς λειτουργίας των κεντρικών υπολογιστών είναι το δυσκολότερο και δαπανηρότερο μέρος του SETI@home. Οι πηγές αποτυχίας τόσο στο hardware όσο και στο software έχουν φανεί απεριόριστες. Έχουν συγκλίνει σε μια αρχιτεκτονική που ελαχιστοποιεί τις εξαρτήσεις

μεταξύ των υποσυστημάτων του server. Παραδείγματος χάριν, ο data/result server μπορεί να λειτουργεί με τη μορφή όπου, αντί να χρησιμοποιήσει τη βάση δεδομένων για να απαριθμήσει τα work units που θα στείλει, παίρνει αυτές τις πληροφορίες από ένα αρχείο του δίσκου. Αυτό μας επιτρέπει να στείλουμε τα στοιχεία ακόμα και όταν έχει πέσει η βάση δεδομένων.

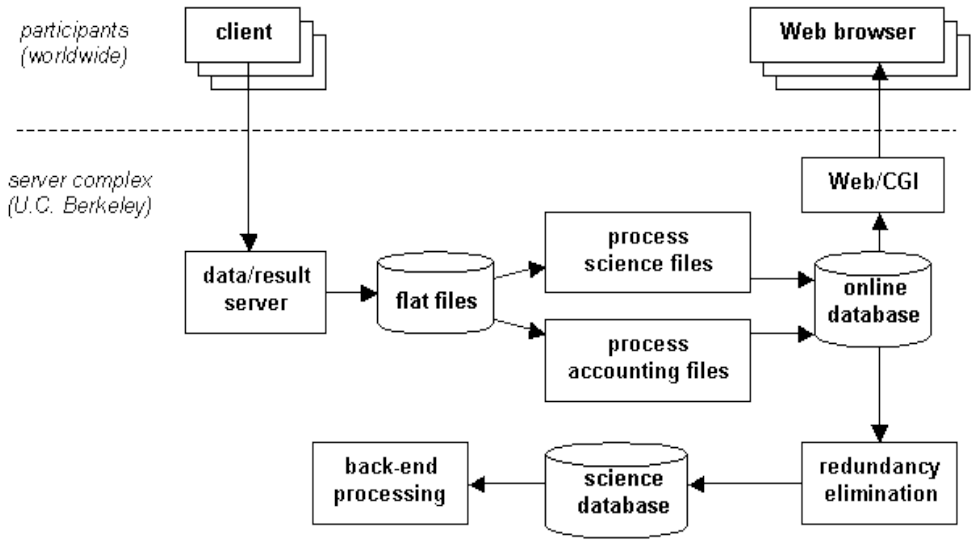
Το πρόγραμμα πελατών παίρνει επανειλημμένα work unit από το data/result server, τα αναλύει, και επιστρέφει το αποτέλεσμα (ένα κατάλογο υποψηφίων σημάτων) στον server. Χρειάζεται σύνδεση με το διαδίκτυο μόνο κατά την επικοινωνία με το server. Ο πελάτης μπορεί να διαμορφωθεί για να υπολογίζει μόνο όταν ο οικοδεσπότης του είναι ανενεργός, ή για να υπολογίζει συνεχώς σε μια χαμηλή προτεραιότητα. Το πρόγραμμα γράφει περιοδικά τα αποτελέσματα του σε ένα αρχείο στο δίσκο, και διαβάζει αυτό το αρχείο στο ξεκίνημα του, ως εκ τούτου σημειώνει πρόοδο ακόμα κι αν ο οικοδεσπότης κλείνεται συχνά.

Το SETI@home client program είναι γραμμένο σε C++. Ο κώδικας αποτελείται από ένα platform-independent πλαίσιο για το διανεμημένο υπολογισμό (6,423 γραμμές), components με platform-specific εφαρμογές, όπως η graphics library (2,058 γραμμές στην έκδοση Unix), SETI- συγκεκριμένος κώδικας ανάλυσης δεδομένων(6,572 γραμμές) και SETI- συγκεκριμένος κώδικας γραφικών(2,247 γραμμές).

Ο πελάτης έχει γίνει port σε 175 διαφορετικές πλατφόρμες. Τα GNU εργαλεία, συμπεριλαμβανομένων των gcc και autoconf, έχουν διευκολύνει πολύ αυτόν τον στόχο. Διατηρούν μόνοι τους τις εκδόσεις των Windows, Macintosh, και SPARC/Solaris ενώ όλο το άλλο porting γίνεται από τους εθελοντές.

Ο πελάτης μπορεί να τρέξει ως background process, ως GUI εφαρμογή, ή ως screensaver. Για να υποστηρίξουν αυτούς τους διαφορετικούς τρόπους στις πολλαπλάσιες πλατφόρμες, χρησιμοποιούν μια αρχιτεκτονική στην οποία ένα thread κάνει την επικοινωνία και την επεξεργασία δεδομένων, ένα δεύτερο thread χειρίζεται τις GUI αλληλεπιδράσεις, και ένα τρίτο thread (ίσως σε ένα χωριστό διάστημα διευθύνσεων) δίνει τη γραφική παράσταση που βασίζεται on a shared-memory δομή δεδομένων.

Τα αποτελέσματα επιστρέφονται στο SETI@home server, όπου καταγράφονται και αναλύονται (δείτε Εικόνα 5).



Εικόνα 5 Η συλλογή και η ανάλυση των αποτελεσμάτων.

Ο χειρισμός ενός αποτελέσματος αποτελείται από δύο στόχους:

- **Επιστημονικός**: The data/server γράφει το αποτέλεσμα σε ένα αρχείο του δίσκου. Ένα πρόγραμμα διαβάζει αυτά τα αρχεία, δημιουργώντας τα αρχεία αποτελέσματος και σημάτων στη βάση δεδομένων. Για να βελτιστοποιηθεί το throughput, διάφορα αντίγραφα αυτού του προγράμματος λειτουργούν ταυτόχρονα.
- **Λογιστικός**: Για κάθε αποτέλεσμα, ο κεντρικός υπολογιστής γράφει ένα log entry που περιγράφει το αποτέλεσμα του χρηστή, το χρόνο της CPU, και τα λοιπά. Ένα πρόγραμμα διαβάζει αυτά τα log files, συσσωρεύοντας σε μια κρύπτη μνήμης τις αναπροσαρμογές σε όλα τα σχετικά αρχεία βάσεων δεδομένων (χρήστης, ομάδα, χώρα, τύπος CPU, και τα λοιπά). Ανά ένα συγκεκριμένο χρονικό διάστημα περνάει αυτά τα στοιχεία στη βάση δεδομένων.

Με την αποθήκευση logs σε αρχεία του δίσκου, το σύστημα κεντρικών υπολογιστών μπορεί να χειριστεί τις περιόδους διακοπής λειτουργίας και υπερφόρτωσης βάσεων δεδομένων.

Τελικά, κάθε ένα work unit έχει ένα αριθμό αποτελεσμάτων στη βάση δεδομένων. Το **redundancy elimination** πρόγραμμα εξετάζει κάθε ομάδα περιττών αποτελεσμάτων – τα όποια μπορεί να διαφέρουν σε αριθμό σημάτων και παραμέτρων σημάτων - και χρησιμοποιεί μια κατά προσέγγιση πολιτική συναίνεσης που επιλέγει ένα "κανονικό" αποτέλεσμα για αυτό το work unit. Τα κανονικά αποτελέσματα αντιγράφονται σε μια χωριστή βάση δεδομένων. Η τελική φάση, **back-end processing**, αποτελείται από διάφορα βήματα. Για να ελέγξει το σύστημα, ελέγχουν για τα σήματα δοκιμής που έρχονται από το τηλεσκόπιο. Προκαλούμενα από τον άνθρωπο σήματα (RFI) προσδιορίζονται και

αποβάλλονται. Ψάχνουν τα σήματα με τις παρόμοιες συντεταγμένες συχνότητας και ουρανού που ανιχνεύονται σε διαφορετικούς χρόνους. αυτά τα "επαναλαμβανόμενα σήματα", καθώς επίσης και τα one-time σήματα ικανοποιητικής αξίας, ερευνώνται περαιτέρω, ενδεχομένως οδηγώντας σε μια τελική διασταύρωση από άλλα SETI projects σύμφωνα με ένα αποδεκτό πρωτόκολλο [DOP90].

2.3 Technical reference of folding@home

Το project αυτό βασίζεται στην πλατφόρμα του Cosm,ακολουθεί μια αναφορά στην αρχιτεκτονική της πλατφόρμας αυτής.

Η σχεδίαση του COSM είναι ιδιαίτερα redundant σε όλα τα επίπεδα έτσι ώστε εάν ένας κόμβος αποτυγχάνει, κανένας άλλος κόμβος δεν πρέπει να επηρεαστεί. Οι Proxies και servers είναι σε κυκλική μορφή ,όπου κάθε κόμβος μέσα στο κύκλο μπορεί να προσφέρει τις ίδιες λειτουργίες με οποιοδήποτε άλλο κόμβο στην κυκλική διάταξη.

2.3.1 Network Architecture

Υπάρχουν τρία στρώματα στην αρχιτεκτονική του Cosm δικτύου:

2.3.1.1 Servers

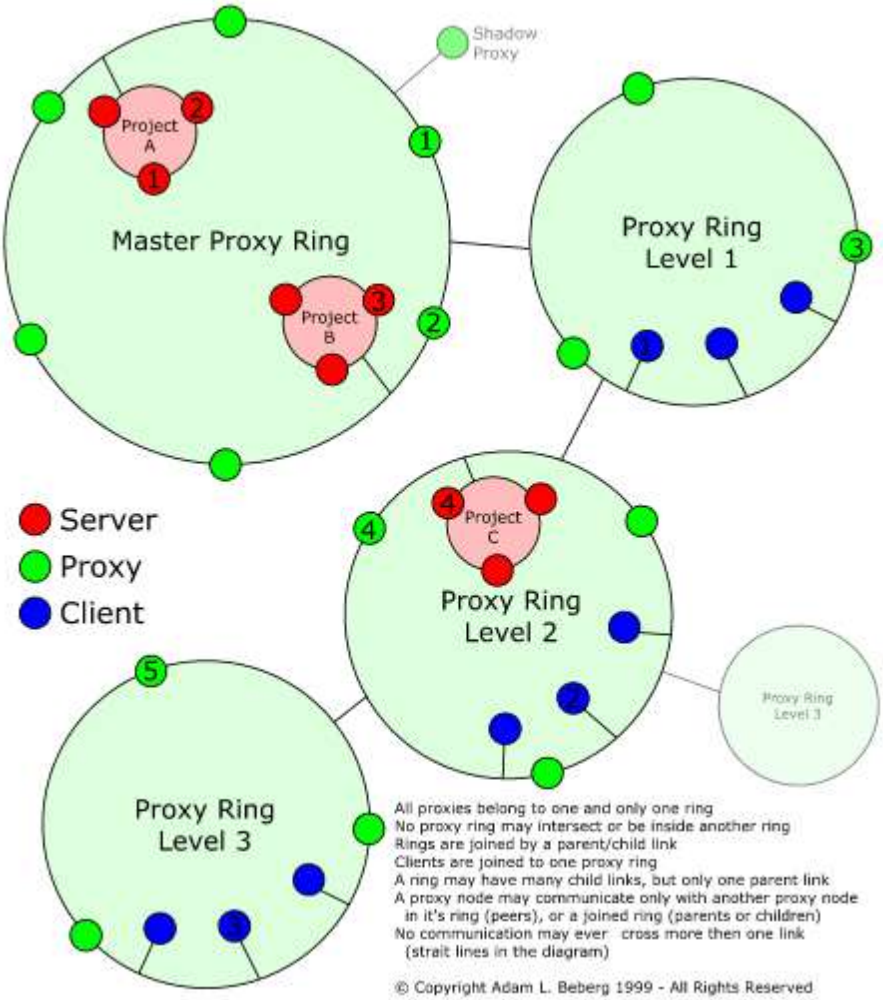
Οι Servers λειτουργούν στο να δίνουν εργασίες στους workers, να δέχονται τα αποτελέσματα από τις ήδη ολοκληρωμένες εργασίες, και να ενημερώνουν αρχεία πληροφοριών σχετικά με το project. Οι Servers επικοινωνούν με κάθε ένα proxy στο layer που βρίσκεται κοντύτερα σε αυτούς, ή σε άλλους servers, αλλά δεν επικοινωνούν με clients.

2.3.1.2 Proxies

Οι Proxies είναι οι ενδιάμεσοι του δικτύου και κάνουν σχεδόν όλες τις επικοινωνίες. ένας proxy μπορεί κάποιες φορές να επικοινωνήσει με κάποιο άλλο proxie στο layer του για να διανείμει μηνύματα, για να ισορροπήσει τα φορτία, ή να επιλύσει δικτυακά προβλήματα.

2.3.1.3 Clients

Οι Clients είναι οι εργαζόμενοι του δικτύου. Τους δίνονται εργασίες από τους proxies και επιστρέφουν τα αποτελέσματα σε αυτούς. κάθε φορά που έχουν εργασία τρέχουν τον κώδικα τους και αναφέρουν στην συνέχεια, αφού τελειώσει η εργασία, τα αποτελέσματα πίσω στους proxies. Οι Clients επίσης λειτουργούν ως DFS κόμβοι Εάν αυτό τους ζητηθεί. Οι clients είναι Έτσι σχεδιασμένοι ώστε να τους εγκαταστήσεις και στην συνέχεια να μην ξανασχοληθείς μαζί τους εκτός και Εάν χρειαστεί να αλλαχτούν κάποιες ρυθμίσεις ασφαλείας. όταν εμφανιστεί κάποιο πρόβλημα το οποίο δεν μπορεί να λυθεί αυτόματα τότε ο CLIENT μπορεί να στείλει ένα email αναφέροντας ότι παρουσιάστηκε κάποιο πρόβλημα.



Εικόνα 6 the proxy rings

2.3.2 Ασφάλεια

Το λογισμικό θα φορτώσει και θα μεταφορτώσει τα στοιχεία μόνο από τον κεντρικό υπολογιστή στοιχείων στο Στάνφορντ. Ο κεντρικός υπολογιστής στοιχείων(server) δεν μεταφορτώνει οποιοδήποτε εκτελέσιμο κώδικα στον υπολογιστή των εθελοντών. Λαμβάνουν εκτενή μέτρα για να ελεγχθούν όλα τα στοιχεία που εισάγονται στον υπολογιστή και τα αποτελέσματα που στέλνονται πίσω στο Στάνφορντ με τις ψηφιακές υπογραφές μπιτ 2048. Εάν οι υπογραφές δεν ταιριάζουν (on either the input out the output) ο πελάτης θα απορρίψει τα στοιχεία και θα αρχίσει πάλι. Αυτό εξασφαλίζει, χρησιμοποιώντας τα καλύτερα μέτρα ασφάλειας λογισμικού που αναπτύσσονται μέχρι σήμερα (digital signatures and PKI in version 3.0), ότι κρατούν τη καλύτερη δυνατή ασφάλεια.

Τέλος, ο screen server είναι διαθέσιμος προς μεταφόρτωση μόνο τον επίσημο ιστοχώρο του project, έτσι ώστε να μπορούν να εγγυηθούν την ακεραιότητα του λογισμικού. Δεν υποστηρίζουν το λογισμικό Folding@home αποκτηθέν αλλού και απαγορεύουν τη διανομή του λογισμικού, εξασφαλίζοντας έτσι ότι παρήχθη πραγματικά από το client software και δεν αλλάχτηκε προτού επιστραφεί στον κεντρικό υπολογιστή. Οι ψηφιακές υπογραφές χρησιμεύουν ως ένας τύπος checksum, keyed by some long (e.g. 256 bit) string of data. Εάν υπήρξε κάποια αλλαγή στα στοιχεία, ο ψηφιακός έλεγχος υπογραφών θα αποτύχει και ο κεντρικός υπολογιστής μπορεί να καθορίσει ότι η ακεραιότητα των δεδομένων παραβιάστηκε. Εναλλακτικά, άλλες κρυπτογραφικές μέθοδοι μπορούν να χρησιμοποιηθούν, όπως η κρυπτογράφηση.

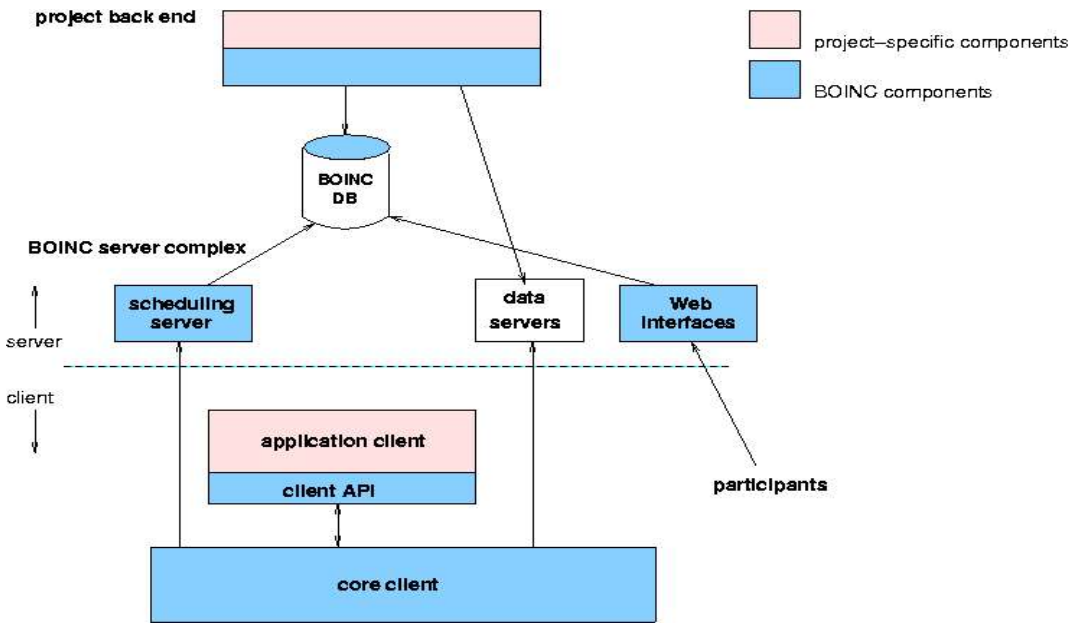
2.4 Πλατφόρμες

2.4.1 Boinc

2.4.1.1 Projects and applications

Ένα **project** είναι ένα σύνολο μιας ή περισσότερων διανεμημένων εφαρμογών, το οποίο χρησιμοποιεί την πλατφόρμα BOINC. Τα project είναι ανεξάρτητα. Καθένα έχει τις εφαρμογές, τις βάσεις δεδομένων και τους κεντρικούς υπολογιστές του, και δεν επηρεάζεται από τη θέση άλλων projects.

Τα συστατικά ενός project παρουσιάζονται κατωτέρω.



Εικόνα 7 τα συστατικά ενός project

Η πλευρά κεντρικών υπολογιστών ενός προγράμματος αποτελείται από δύο μέρη:

- ένα **project back end** που παρέχει εφαρμογές και work units και χειρίζεται τα υπολογιστικά αποτελέσματα. Input και output αρχεία διανέμονται από **data servers**, που είναι HTTP server's ικανοί να χειριστούν CGI προγράμματα με POST εντολές. Αυτοί οι servers δεν χρειάζεται να ανήκουν η να χειρίζονται από το project. ένα project μπορεί, παραδείγματος χάριν, να στρατολογήσει άλλους οργανισμούς για να δώσουν bandwidth με τη φιλοξενία data servers. Τα δεδομένα θα μπορούσαν να μετακινηθούν με αποθηκευτικά μέσα μεταξύ project back end και data servers.

- ένας **BOINC server complex** διαχειρίζεται τη διανομή και τη συλλογή δεδομένων.

Ο BOINC server complex περιλαμβάνει τα ακόλουθα συστατικά:

- Ένας ή περισσότεροι **scheduling servers.** Αυτός επικοινωνεί με τους οικοδεσπότες συμμετεχόντων.

- Μια σχεσιακή βάση δεδομένων που αποθηκεύει τις πληροφορίες για την εργασία, τα αποτελέσματα, και τους συμμετέχοντες.

- Προγράμματα χρήσης και βιβλιοθήκες που επιτρέπουν στο project back end να αλληλεπιδράσει με το server complex.

- Web interfaces για τους συμμετέχοντες και τους υπεύθυνους για την ανάπτυξη.

2.4.1.2 Files and data servers

Το πρότυπο αποθήκευσης BOINC είναι βασισμένο στα αρχεία. Τα inputs and outputs από τις εφαρμογές, και τα executables των εφαρμογών, είναι αρχεία. Το BOINC core client μεταφέρει αρχεία από και προς τους **data servers** που χειρίζονται το project, χρησιμοποιώντας HTTP. Μόλις δημιουργηθεί ένα αρχείο (στο data server ή σε ένα οικοδεσπότη συμμετεχόντων) είναι αμετάβλητο. Χαρακτηριστικά, ο BOINC server στέλνει "δεδομένα" στους πελάτες, και οι πελάτες εκτελούν τον υπολογισμό και δίνουν τα αποτελέσματα στο server.

Το BOINC παρέχει μια μορφή redundant υπολογισμού στην οποία κάθε υπολογισμός εκτελείται από πολλαπλάσιους πελάτες, τα αποτελέσματα συγκρίνονται, και γίνεται αποδεκτός μόνο όταν επιτυγχάνεται μια «συμφωνία» στα αποτελέσματα.

Οι περισσότερες αριθμητικές εφαρμογές παράγουν διαφορετικά αποτελέσματα για work unit ανάλογα με την αρχιτεκτονική μηχανών, το λειτουργικό σύστημα, το μεταγλωττιστή, και τα flag των μεταγλωττιστών. Σε τέτοιες περιπτώσεις μπορεί να είναι δύσκολο να διακρίνει κανείς μεταξύ των αποτελεσμάτων που είναι σωστά, αλλά διαφέρουν λόγω της αριθμητικής παραλλαγής, και των αποτελεσμάτων που είναι λανθασμένα.

Το BOINC παρέχει την δυνατότητα ενός **homogeneous redundancy** για τέτοιου είδους εφαρμογές. Μπορεί να επιτραπεί για ένα πρόγραμμα με τη συμπερίληψη της γραμμής

<homogeneous_redundancy/>

μέσα στο config.xml αρχείο.

Όταν αυτό το χαρακτηριστικό γνώρισμα επιτρέπεται, ο BOINC scheduler θα στείλει τα αποτελέσματα για workunit μόνο στους οικοδεσπότες με το ίδιο όνομα συστημάτων λειτουργίας και CPU vendor (π.χ. το λειτουργικό σύστημα και την ταυτότητα του επεξεργαστή από την περιγραφή του οικοδεσπότη). Παραδείγματος χάριν: εάν ένα αποτέλεσμα έχει σταλεί σε ένα πλήθος του τύπου Windows XP, Intel, έπειτα άλλα αποτελέσματα αυτού του work unit θα σταλούν μόνο στους οικοδεσπότες του τύπου Windows XP, Intel.

Το BOINC διαχειρίζεται τις περισσότερες από τις λεπτομέρειες, εντούτοις υπάρχουν δύο θέσεις όπου ο υπεύθυνος για την ανάπτυξη εφαρμογής αναμιγνύεται:

Validation: Αυτό εκτελεί δύο λειτουργίες. Κατ' αρχάς, όταν ένας ικανοποιητικός αριθμός από τα επιτυχή αποτελέσματα έχει επιστραφεί, τους συγκρίνει και βλέπει εάν υπάρχει «συμφωνία». Η μέθοδος σύγκρισης των αποτελεσμάτων (η όποια μπορεί να πρέπει να λάβει υπόψη platform- διαφορετική floating point αριθμητική) και η πολιτική για τον καθορισμό «συμφωνίας» (π.χ. καλύτερα δύο από τα τρία) παρέχεται από την εφαρμογή. Εάν επιτυγχάνεται «συμφωνία», ένα ιδιαίτερο αποτέλεσμα υποδεικνύεται ως «κανονικό» αποτέλεσμα. Δεύτερον, εάν ένα αποτέλεσμα φθάνει μετά από την επίτευξη «συμφωνίας», το νέο αποτέλεσμα συγκρίνεται με το «κανονικό» αποτέλεσμα και αυτή η σύγκριση καθορίζει εάν ο χρήστης παίρνει την πίστωση.

Assimilation: Αυτό είναι ο μηχανισμός από τον οποίο το πρόγραμμα ειδοποιείται για την ολοκλήρωση (επιτυχής ή ανεπιτυχής) μιας work unit. Εκτελείται ακριβώς μία φορά ανά work unit. Εάν το work unit ολοκληρώθηκε επιτυχώς (π.χ. εάν υπάρχει ένα «κανονικό» αποτέλεσμα) μια παρεχόμενη λειτουργία από το project διαβάζει το ή τα αρχεία παραγωγής και διαχειρίζεται τις πληροφορίες, όπως π.χ. με την καταγραφή τους σε μια βάση δεδομένων. Εάν το work unit απέτυχε, η λειτουργία μπορεί να γράψει μια είσοδο στο log,να στείλει ένα email, κ.λπ.

2.4.1.3 Trickle messages

Τα **Trickle messages** αφήνουν τις εφαρμογές να επικοινωνήσουν με τον κεντρικό υπολογιστή κατά τη διάρκεια της εκτέλεσης ενός workunit. Προορίζονται για τις εφαρμογές που έχουν τις μακροχρόνιες workunit. Τα Trickle messages μπορούν να ακολουθούν καθεμία κατεύθυνση: 'trickle up' messages πηγαίνουν από την εφαρμογή στο κεντρικό υπολογιστή, 'trickle down' messages πηγαίνουν από τον κεντρικό υπολογιστή στην εφαρμογή. Χαρακτηριστικές χρήσεις αυτού του μηχανισμού:

- Η εφαρμογή στέλνει a trickle-up message περιέχοντας το χρόνο χρήσης του CPU, έτσι ώστε στους χρήστες μπορεί να χορηγηθεί πίστωση (παρά την αναμονή μέχρι το τέλος της work unit).

- Η εφαρμογή στέλνει το trickle-up message περιέχοντας μια αναφορά για την κατάσταση του υπολογισμού, έτσι ώστε η λογική των κεντρικών υπολογιστών μπορεί να αποφασίσει εάν θα σταματήσει ο υπολογισμός.

- Ο κεντρικός υπολογιστής στέλνει a trickle-down message λέγοντας στην εφαρμογή να σταματήσει τον υπολογισμό.

- Ο κεντρικός υπολογιστής στέλνει a trickle-down message αναφέροντας την τρέχουσα συνολική πίστωση του χρήστη.

Τα Trickle messages είναι ασύγχρονα και αξιόπιστα. Τα Trickle messages μεταβιβάζεται μέσα από scheduler RPC messages

2.4.1.4 Ασφάλεια

Πολλοί τύποι επιθέσεων είναι δυνατοί.

- **Result falsification**. Οι επιτιθέμενοι επιστρέφουν ανακριβή αποτελέσματα.

- **Credit falsification**. Οι επιτιθέμενοι επιστρέφουν τα αποτελέσματα και απαιτούν περισσότερο CPU time από ότι χρησιμοποιήθηκε πραγματικά.

- **Malicious executable distribution**. Οι επιτιθέμενοι μπαίνουν σε ένα BOINC server και, με την τροποποίηση της βάσης δεδομένων και των αρχείων, προσπαθούν να διανείμουν εκτελέσιμό τους (π.χ. ένα πρόγραμμα ιών) μεταμφιεσμένο ως BOINC εφαρμογή.

- **Overrun of data server**. Οι επιτιθέμενοι στέλνουν επανειλημμένα μεγάλα αρχεία στους BOINC data servers, γεμίζοντας τους δίσκους τους και καθιστώντας τους ακατάλληλους προς χρήση.

- **Theft of participant account information by server attack**. Οι επιτιθέμενοι μπαίνουν σε ένα BOINC server και κλέβουν τις διευθύνσεις ηλεκτρονικού ταχυδρομείου και άλλες πληροφορίες απολογισμού.

Το BOINC παρέχει τους μηχανισμούς για να μειώσει την πιθανότητα για μερικές από αυτές τις επιθέσεις.

Result falsification

Αυτό μπορεί να ανιχνευθεί βάσει πιθανοτήτων χρησιμοποιώντας επανειλημμένους υπολογισμούς για την επαλήθευση αποτελέσματος: εάν η πλειοψηφία των αποτελεσμάτων συμφωνεί (σύμφωνα με μια οριζόμενη από εφαρμογή σύγκριση) αυτά ταξινομούνται ως σωστά.

Credit falsification

Αυτό μπορεί να ανιχνευθεί βάσει πιθανοτήτων χρησιμοποιώντας περιττούς υπολογισμούς για την πιστωτική επαλήθευση: σε κάθε συμμετέχοντα δίνεται η ελάχιστη πίστωση μεταξύ των σωστών αποτελεσμάτων (ή κάποιου άλλου αλγορίθμου, όπως ο μέσος των πιστώσεων που απαιτούνται).

Malicious executable distribution

Το BOINC χρησιμοποιεί code signing για να αποτρέψει αυτό. Κάθε ένα project έχει ένα ζευγάρι κλειδιά για code signing. Το private key πρέπει να φυλάσσεται σε απομονωμένο από το δίκτυο μηχάνημα που θα χρησιμοποιείται για την παραγωγή digital signatures για τα εκτελέσιμα .Το public key διανέμεται, και αποθηκεύεται στους clients. Τα αρχεία που συνδέονται όλα με τις εκδόσεις εφαρμογής στέλνονται με ψηφιακή υπογραφή χρησιμοποιώντας αυτό το ζευγάρι κλειδιών.

Το BOINC παρέχει έναν μηχανισμό από τον οποίο projects μπορούν περιοδικά να αλλάξουν το code-signing ζευγάρι κλειδιών. Το project παράγει ένα νέο ζευγάρι κλειδιών, κατόπιν (χρησιμοποιώντας το

code-signing machine) παράγει μια ψηφιακή υπογραφή για το νέο public key, υπογεγραμμένη με το παλαιό private key. Ο core client θα δεχτεί έναν νέο key μόνο εάν υπογράφεται με το παλαιό κλειδί. Αυτός ο μηχανισμός σχεδιάζεται για να αποτρέψει τους επιτιθεμένους από το να μπουν στο BOINC server και να διανείμουν ένα ψεύτικο κλειδί.

Denial of server attacks on data servers

Κάθε αρχείο αποτελέσματος έχει ένα σχετικό μέγιστο μέγεθος. Κάθε ένα project έχει upload authentication ζευγάρι κλειδιών. Το public key αποθηκεύεται σε project's data servers. Οι περιγραφές αρχείων αποτελέσματος στέλνονται στους πελάτες με digital signature, το οποίο διαβιβάζεται στον data server όταν το αρχείο γίνεται uploaded. Ο data server ελέγχει την περιγραφή αρχείων, και εξασφαλίζει ότι το ποσό στοιχείων των uploaded δεν υπερβαίνει το μέγιστο μέγεθος.

Theft of participant account information by server attack

Όλοι οι server machines πρέπει να προστατεύονται από firewall, και πρέπει να έχουν όλα τα αχρησιμοποίητα network services απενεργοποιημένα. Η πρόσβαση σε αυτές τις μηχανές πρέπει να γίνει μόνο με τα κρυπτογραφημένα πρωτόκολλα όπως SSH.

2.4.1.5 Scheduling server: policy

Εάν το πόσο της εργασίας που έχει δώσει ο scheduling server ξεπερνά ένα συγκεκριμένο όριο (η προεπιλογή είναι τέσσερις εβδομάδες),ο scheduling server δεν θα δώσει περισσότερη εργασία με τη scheduler reply. Αυτό δεν αποτρέπει το scheduling server από την αποστολή ενός μεγάλου workunit, αλλά μάλλον από την αποστολή πολλαπλάσιων μεγάλων workunits.

Εάν μια work unit χρησιμοποιήσει περισσότερους πόρους δίσκων από ότι ένας worker έχει διαθέσιμο, ο κεντρικός υπολογιστής σχεδιασμού δεν θα διαθέσει εκείνη την work unit.

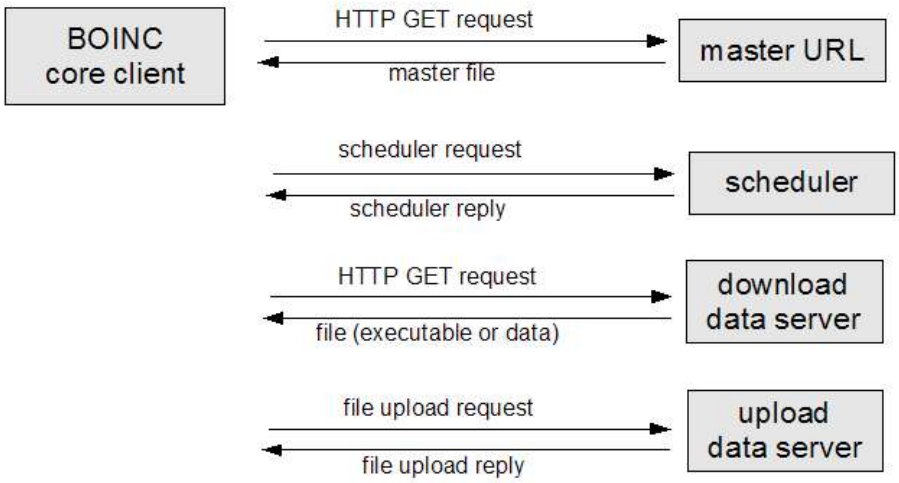
Ο scheduling server υπολογίζει το χρονικό διάστημα που μια work unit θα πάρει για να ολοκληρώσει με τον τύπο (number of flops)/(flops per second)+(number of iops)/(iops per second). Ο αριθμός floating point και integer operations παρέχεται από το project κατά τη δημιουργία της work unit, και οι ταχύτητες υπολογισμού των worker συμπεριλαμβάνονται στο scheduler request.

Εάν καμία εργασία δεν είναι διαθέσιμη, ή εάν ο οικοδεσπότης δεν μπορεί να την δεχτεί για οτιδήποτε λόγο (πάρα πολύ αργός, όχι αρκετό διάστημα, κ.λπ...), scheduling server στέλνει στο worker ένα μήνυμα της μορφής no work available, και ζητά από το worker να περιμένει πριν στέλνει ένα άλλο αίτημα.

2.4.1.6 Host identification

Όταν ένας host αρχικά έρχεται σε επαφή με το scheduling server του αναθέτεται ένα host ID. Ο server επίσης διατηρεί ένα RPC sequence αριθμό για κάθε έναν host. Και τα δύο το host Id και ο RPC sequence αριθμός αποθηκεύεται στους client's ,στο client_state.xml αρχείο.

2.4.1.7 Επικοινωνία με τον Worker



Εικόνα 8 αναλυτικά η επικοινωνία με το worker

- Ο client μεταφορτώνει τη σελίδα από το project's master URL. Από XML tags ενσωματωμένο σε αυτήν την σελίδα, λαμβάνει έναν κατάλογο domain names από **schedulers**.
- Ο client ανταλλάσσει request and reply messages με το scheduling server. Το μήνυμα απάντησης περιέχει, μεταξύ άλλων, τις περιγραφές της εργασίας που εκτελούνται, και των καταλόγων URLS για input και output files για εκείνη την εργασία.
- Ο πελάτης μεταφορτώνει τα αρχεία (προγράμματα εφαρμογής και αρχεία δεδομένων) από έναν ή περισσότερους **download data servers**. Για να γίνει αυτό χρησιμοποιεί τα πρότυπα HTTP GET requests, ίσως με Range commands για να επαναληφθούν ελλιπείς μεταφορές.
- Αφότου ο υπολογισμός τελειώσει, ο client φορτώνει τα αρχεία αποτελέσματος. Για να γίνει αυτό χρησιμοποιεί ένα BOINC-specific protocol που προστατεύει από DOS attacks σε data servers.
- Ο client κατόπιν έρχεται σε επικοινωνία με το scheduling server πάλι, αναφέρει την ολοκληρωμένη εργασία και ζητά άλλη εργασία.

2.4.2 Parabon computation

2.4.2.1 Frontier Components

Η Frontier's πλατφόρμα υπολογισμού διαδικτύου υποστηρίζει ότι τα Tasks δεν μπορούν να επικοινωνούν με άλλα tasks που τρέχουν και αποτελείται από τρία κύρια συστατικά:

- The Pioneer Compute Engine
- The Frontier Server
- The Frontier Software Development Kit (SDK)

2.4.2.2The Pioneer Compute Engine

Το Pioneer είναι μια desktop εφαρμογή που διαχειρίζεται αυτόματα και χρησιμοποιεί την αχρησιμοποίητη δύναμη επεξεργασίας του υπολογιστή ενός εθελοντή. Η μηχανή επιτρέπει διακριτές υπολογιστικές εργασίες (work units) να γίνουν download από το Frontier server και επεξεργάζονται κατά τη διάρκεια που είναι μη ενεργός ο υπολογιστής. Όταν η επεξεργασία τελειώσει, τα αποτελέσματα φορτώνονται την επόμενη φορά που μια σύνδεση επιτυγχάνεται με το Frontier server.

Το Pioneer που τρέχει στον υπολογιστή ενός εθελοντή είναι γραμμένο σε Java™ γλώσσα προγραμματισμού και εκτελείται μέσα σε ένα περιορισμένο, self-enclosed περιβάλλον αποκαλούμενο "sandbox." Η επεξεργασία των work units μέσα στο sandbox είναι παρόμοια με το πώς τα Java applets αντιμετωπίζονται στις ιστοσελίδες και εκτελούνται από σύγχρονους browsers. Αντίθετα από τη γλώσσα προγραμματισμού C, η Java's sandbox τεχνολογία εξασφαλίζει ότι ένας υπολογιστής εθελοντή παραμένει ασφαλής. Η χρήση της Java επίσης εξασφαλίζει ότι το διαφορετικό hardware από τους εθελοντές εμφανίζεται ομοιογενής και συμπεριφέρεται όμοια για την αποδοτικότερη επεξεργασία.

2.4.2.3The Frontier Server

Ο Frontier server είναι ο αρμόδιος για την ενορχήστρωση της εκτέλεσης από τις εργασίες των πελατών. Ελέγχει τη κατάσταση όλων των εργασιών που τρέχουν στο σύστημα, όπως και τη κατάσταση κάθε Pioneer compute engine, έτσι ώστε να μπορεί αποτελεσματικά να δώσει εργασίες σε αυτές τις μηχανές. Ο Frontier server λαμβάνει υπόψη το γεγονός ότι κάποιοι πόροι των εθελοντών μπορούν να μην είναι συνεχώς διαθέσιμοι, δεδομένου ότι οι εθελοντές μπορούν να αποσυνδέσουν τους υπολογιστές τους

οποιαδήποτε στιγμή. Εάν, μετά από έναν ορισμένο χρόνο, η εργασία στον υπολογιστή του εθελοντή δεν ανταποκρίνεται τότε ο server στέλνει την εργασία σε άλλο διαθέσιμο εθελοντή.

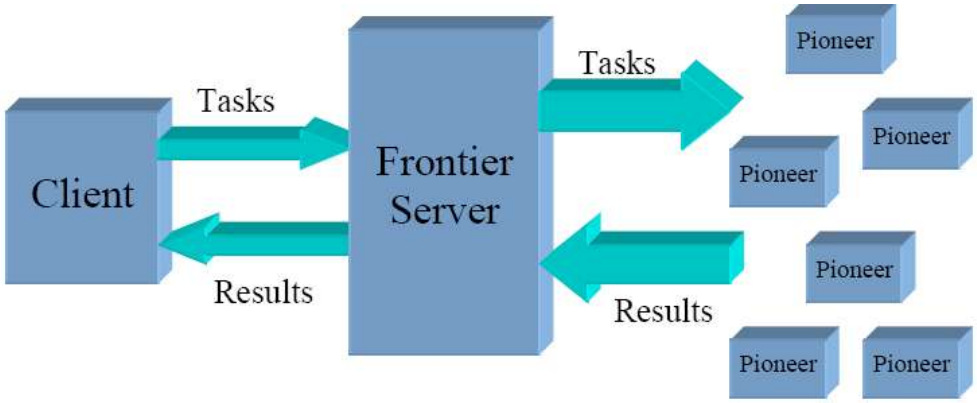
Ο Frontier server είναι βασισμένος σε μια αποδεδειγμένη τρεις-τοποθετημένη στη σειρά αρχιτεκτονική. Λόγω της αρχιτεκτονικής των server's, οποιοδήποτε συστατικό μπορεί να αποτύχει χωρίς την απώλεια μιας συναλλαγής. αυτή η αρχιτεκτονική του server του επιτρέπει να ικανοποιήσει μια γρήγορα αυξανόμενη ζήτηση.

2.4.2.4The Frontier SDK

Το Frontier SDK είναι μια συλλογή εργαλείων λογισμικού που επιτρέπει στους πελάτες να γράψουν και να ελέγχουν την εκτέλεση μεγάλων υπολογιστικά - εντατικών εργασιών που μπορούν να αποτελούνται από εκατοντάδες χιλιάδες μικρότερες εργασίες. Στον πυρήνα του Frontier SDK είναι η διεπαφή προγραμματισμού εφαρμογής (API), η όποια παρέχει μια εύκολη σε χρήση διεπαφή προγραμματισμού για την ανάπτυξη εφαρμογών που θα τρέξουν επάνω στο Frontier.

Το Frontier SDK παρέχει διάφορες βιβλιοθήκες που εφαρμόζουν αποδεδειγμένα παράλληλα παραδείγματα για να βοηθήσουν τους πελάτες να αναπτύξουν και να κάνουν porting τις εφαρμογές τους που θα τρέξουν επάνω στο Frontier. χρησιμοποιώντας το SDK's πίνακα γραφικών εργαλείων, ο πελάτης μπορεί να προωθήσει τις εργασίες, να ελέγξει τις εργασίες και να συλλέξει τα αποτελέσματα των εργασιών, όλα από οποιοδήποτε υπολογιστή με πρόσβαση διαδικτύου.

Κάθε εργασία που προωθείται παρατίθεται μαζί με τη κατάσταση της. Εάν ένας πελάτης καθορίσει ότι η εργασία περιέχει ένα λάθος ή θέλει να ρυθμίσει τον κώδικά του ,να τροποποιήσει ή διορθώσει τον κώδικά του ,μπορεί να το κάνει και να στείλει εκ νέου την εργασία του.



Εικόνα 9 το frontier client-server μοντέλο.

2.4.2.5Data and Executable Elements

ένα *element* είναι ο μηχανισμός που χρησιμοποιείται για να μεταφέρει αποτελεσματικά τα σχετικά μεγάλα κομμάτια των δυαδικών δεδομένων που απαιτούνται από το worker για να εκτελέσει την εργασία. Τα δεδομένα και τα εκτελέσιμα στοιχεία μπορούν να είναι συνδεμένα με μια ενιαία εργασία. Στέλνονται από το client application στο server και κατευθύνονται σε υπολογιστικούς κόμβους όπως απαιτείται πριν από την εκτέλεση μιας εργασίας

Τα ***Executable elements*** παρέχουν Java bytecode instructions απαραίτητες για να τρέξει η εργασία . κάθε executable element είναι σχηματοποιημένο ως Java jar αρχείο και γίνεται διαθέσιμο από την engine στο Java Virtual Machine (JVM) στο οποίο η εργασία εκτελείται

Τα ***Data elements*** είναι μια σειρά από δεδομένα που αναφέρονται σε task's parameter list, και ως εκ τούτου απαιτούνται για το τρέξιμο ενός task. Όλα τα data elements μόλις λαμβάνονται στέλνονται στο compute engine πριν από το τρέξιμο της εργασίας .Η εργασία μπορεί να χειρίζεται αυτά τη ροή από δεδομένα όταν το κρίνει κατάλληλο.

2.4.2.6Checkpoints

Οι εργασίες που τρέχουν στον εθελοντή μπορεί να μην τρέξουν και να τελειώσουν απευθείας αλλά να διακοπούν στο ενδιάμεσο .Εάν συμβεί κάτι τέτοιο τότε η εργασία την επόμενη φορά που θα ξεκινήσει δεν είναι απαραίτητο να επανεκινήσει από την αρχή αλλά μπορεί ,και είναι πιο αποδοτικό ,να ξεκινήσει από το σημείο που διεκόπη .Για να γίνει αυτό το Frontier χρησιμοποιεί ένα μηχανισμό που τον ονομάζει *checkpoints* για να μπορεί να καταγράψει την κατάσταση του υπολογισμού .Η εργασία καταγράφει ένα στιγμιότυπο της ίδιας και το χρησιμοποιεί την επόμενη φορά που θα επανεκινησει

Οποιαδήποτε γλώσσα που μπορεί να γίνει compile δίνοντας Java byte code μπορεί να χρησιμοποιηθεί, εφ' όσον μπορεί να διασυνδεθεί με το Frontier Runtime API. Πέρα από αυτό, η εφαρμογή πελατών δεν χρειάζεται να γραφτεί σε Java. Στην πραγματικότητα, εκτός από τον κώδικα της εργασίας, το μόνο κομμάτι της εφαρμογής που πρέπει να χρησιμοποιεί Java είναι το κομμάτι που επικοινωνεί με το Frontier server μέσω του Client API

2.4.2.7Rigid Security Model

Το Frontier's rigid security model συντηρητικά υποθέτει ότι ούτε providers ούτε
οι clients είναι εμπιστευμένες οντότητες.

2.4.2.8Securing Providers

Το Java sandbox λειτουργεί ως ένα αδιαπέραστο εμπόδιο .Ο sandbox security
μηχανισμός ελέγχει την ακεραιότητα του κώδικα μιας εργασίας προτού να μπορέσει να εκτελεσθεί στον υπολογιστής του εθελοντή και απαγορεύει στις εργασίες την δημιουργία οποιονδήποτε συνδέσεων με το διαδικτύο εκτός από εκείνη με τον Frontier server.

2.4.2.9Securing Client Intellectual Property

Το Parabon προστατεύει όλα τα δεδομένα των πελατών χρησιμοποιώντας SSL (Secure Sockets Layer). SSL's authentication mechanism που επιτρέπει στο Frontier server να επιβεβαιώσει την ταυτότητα ενός εθελοντή πριν από την αποστολή των εργασιών ή τη λήψη των αποτελεσμάτων από το Pioneer compute engine.Με το SSL
επίσης ελέγχει ότι τα στοιχεία που διαβιβάζονται πέρα από τη σύνδεση δεν είναι παραποιημένα κατά τη διάρκεια της διέλευσης από το διαδίκτυο Για να εξασφαλίσει την ακεραιότητα των αποτελεσμάτων, το Parabon περιοδικά στέλνει *canary* tasks υπολογισμούς με γνωστά αποτελέσματα στους υπολογιστές των εθελοντών. Εάν ένας εθελοντής επιστρέψει ανακριβή αποτελέσματα, τότε ο εθελοντής δεν χρησιμοποιείται πλέον.

2.4.3Peer to peer - JXTA

Το Project JXTA άρχισε από την Sun Microsystems το 2001.Το JXTA ορίζει ένα σύνολο από πρωτόκολλα που μπορούν να εφαρμοστούν από peers για να επικοινωνήσουν και να συνεργαστούν με άλλους peers που εφαρμόζουν τα JXTA πρωτόκολλα. Προσπαθεί να τυποποιήσει την ανταλλαγή μηνυμάτων, μεταξύ των peer-to-peer συστημάτων, τυποποιώντας πρωτόκολλα παρά εφαρμογές. Αυτήν την περίοδο Java και C υλοποιήσεις των JXTA πρωτοκόλλων είναι διαθέσιμες .Στο JXTA, κάθε peer αναγνωρίζεται ως ID, μοναδικό κατά τη διάρκεια του χρόνου ύπαρξης του.

Τα Peer groups είναι σύνολα από peers τα οποία έχουν κοινό ενδιαφέρον για κάποια εργασία .Τα Peer groups επίσης αναγνωρίζονται από μοναδικά IDs. κάθε Peer μπορεί να ανήκει σε περισσότερα από ένα peer groups, μπορούν να ανακαλύψουν άλλους peers και αλλά peer groups δυναμικά και μπορούν επίσης να δημοσιοποιήσουν τον εαυτό τους για να μπορέσουν να τους ανακαλύψουν άλλοι peers. Τρία είδη επικοινωνίας υποστηρίζονται από το JXTA.

- Το πρώτο είδος ονομάζεται unicast pipe και είναι παρόμοιο με το UDP καθώς είναι αναξιόπιστο.

- Ο δεύτερος τύπος ονομάζεται secure pipe. Το secure pipe δημιουργεί ένα secure tunnel μεταξύ εκείνου που στέλνει και εκείνου που λαμβάνει ,Έτσι δημιουργεί μια ασφαλής και αξιόπιστη μετάδοση.

- Ο τρίτος τύπος ονομάζεται broadcast pipe.όταν χρησιμοποιείται αυτό το πρωτόκολλο τότε το μήνυμα στέλνεται σε όλους τους peers στο συγκεκριμένο peer group.

Το distributed computing framework περιέχει τα ακόλουθα peer groups:το monitor group, το worker group, το task dispatcher group, και το repository group.

- Το monitor group είναι η κορυφαία ομάδα που συντονίζει τη γενική δραστηριότητα του framework, συμπεριλαμβανομένου του χειρισμού των αιτήσεων από peers που θέλουν να λάβουν μέρος στο framework αλλά και την αμέσως επόμενη ανάθεση τους σε κάποιο peer group,και διάφορες άλλες υψηλού επιπέδου πτυχές της διαδικασίας του να σταλθεί κάποια δουλειά.

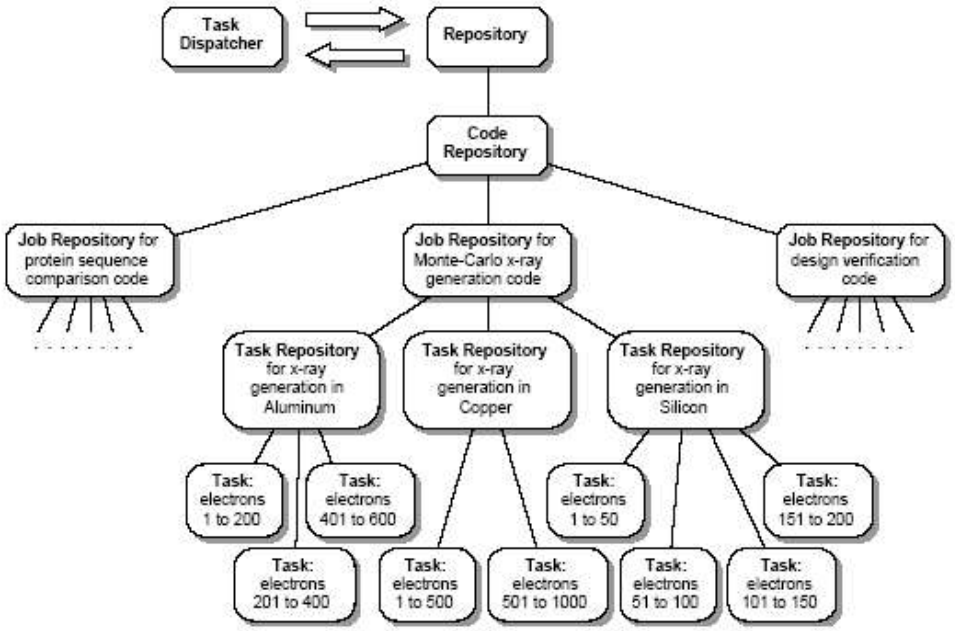
- Το worker group είναι το peer group που είναι υπεύθυνο στο να φέρνει εις πέρας τους υπολογισμούς κάθε δουλειάς.

- Το task dispatcher group διανέμει τις εργασίες στους workers.

- Το repository group έχει το ρόλο της αποθήκευσης δεδομένων και των προγραμμάτων.

Ένας κόμβος μπορεί να ανήκει σε παραπάνω από ένα peer group στο framework,και αντίστοιχα μπορεί να υπάρχουν παρά πάνω από ένα από peer group στο framework.

Υπάρχουν δύο μέρη στην υποβολή μιας εργασίας: ο κώδικας που χρησιμοποιείται από τον εργαζόμενο κόμβο που είναι κοινός για όλες τις εργασίες, και τα δεδομένα που χρησιμοποιούνται από τον κώδικα που ποικίλλουν γενικά για κάθε εργασία. Τα δεδομένα της υποβολής εργασίας μπορεί να κυμαίνονται από την ύπαρξη απλών παραμέτρων που ποικίλουν από εργασία σε εργασία ως και μεγάλα σύνολα δεδομένων που απαιτούνται για τους υπολογισμούς. Όπως με άλλες πτυχές αυτού του πλαισίου, η αποθήκευση των δεδομένων για μια εργασία είναι διανεμημένη σε όλο το δίκτυο σε ένα αποκεντρωμένο τρόπο. Η διαχείριση αυτών ανήκει στο repository peer group, και παράδειγμα του οποίου δίνεται στην εικόνα 10.



Εικόνα 10 Παράδειγμα ενός Code Repository που περιέχει τρεις κώδικες

Παράδειγμα ενός Code Repository που περιέχει τρεις κώδικες ,όπου ο καθένας έχει τη δικιά του job repository. Το Monte-Carlo job repository έχει τρεις διαφορετικές εργασίες ,η κάθε μια αποτελείται από ένα διαφορετικό αριθμό εργασιών .Το interaction με το code repository group και το υπόλοιπο framework γίνεται διαμέσου του task dispatcher group. όταν δεχθεί μια εργασία, ο task dispatcher επικοινωνεί με το repository για να καθορίσει τη κατάσταση του κώδικα μέσα στη code repository. Για κάθε εργασία μια job repository δημιουργείται ,η οποία είναι ένα δέντρο το οποίο περιέχει ένα repository για δοσμένες εργασίες της κύριας εργασίας ,που έχουν σταλεί από τον τελικό χρηστή.

2.4.3.1Distribution of tasks amongst workers

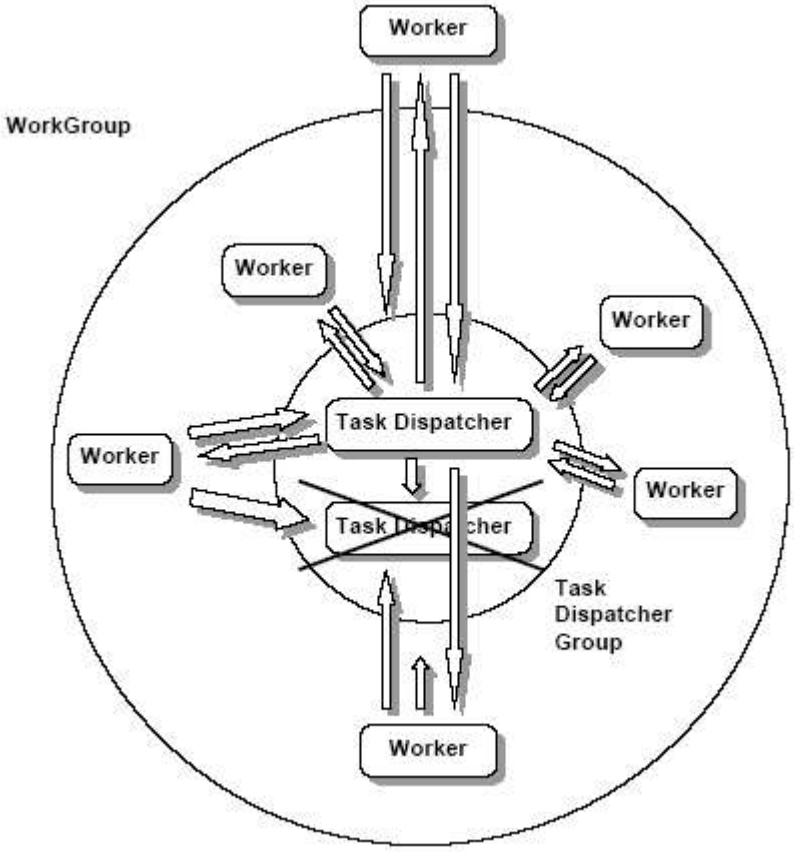
Μέσα σε κάθε ένα worker group υπάρχει ένας task dispatcher. Αχρησιμοποίητοι workers τακτικά δίνουν στο task dispatcher πληροφορίες σχετικά με τους διαθέσιμους πόρους, συμπεριλαμβανομένων των κωδίκων που ο worker έχει εναποθηκεύσει. Με βάση αυτές τις πληροφορίες, ο task dispatcher δίνει στη repository εργασίες που εκτελούνται από τους διαθέσιμους κώδικες, ή τους κώδικες που μεταφορτώνονται στους εργαζομένους .Ο worker εκτελεί την εργασία και επιστρέφει το αποτέλεσμα στον task dispatcher. Είναι σημαντικό να σημειωθεί ότι ο task dispatcher δεν κρατά ποιοι worker εκτελούν ποιες εργασίες .ένας worker θα μπορούσε να γίνει απρόσιτος κατά τη διάρκεια της εκτέλεσης της εργασίας, κάτι το οποίο δεν επιδρά στην συνολική ολοκλήρωση της εργασίας. Ο task dispatcher ενημερώνει τη repository με πληροφορίες για την ολοκλήρωση μιας εργασίας ,και γίνονται επαναυπολογισμοί εργασιών Έτσι ώστε να καλυφθεί η περίπτωση απώλειας ενός worker.

2.4.3.2Result retrieval

Μόλις ολοκληρωθεί μια εργασία, δηλαδή όλες οι επιμέρους εργασίες που βρίσκονται στην task repository ολοκληρωθούν ,τα αποτελέσματα είναι έτοιμα να σταλούν πάλι πίσω σε εκείνο που αρχικά έδωσε τις εργασίες για υπολογισμό. Εντούτοις ,ο task dispatcher δεν παρακολουθεί εκείνους που δίνουν την αρχική «εργασία». Είναι επομένως μέλημα εκείνου που παρείχε την εργασία να κινήσει τη διαδικασία ανάκτησης αποτελέσματος .Εκείνος που έδωσε την εργασία έχει μια μέθοδο που μπορεί να καλέσει και εκείνη θα αναζητήσει από τον task dispatcher να βρει εάν η εργασία έχει ολοκληρωθεί . κάθε εργασία ,της έχει ανατεθεί ένα task repository,το οποίο έχει μοναδικό ID. αυτό το ID στέλνεται στον job submitter όταν η task repository δημιουργηθεί ,και χρησιμοποιείται για να ζητηθεί το αποτέλεσμα .Ο task dispatcher δέχεται αυτή τη ζήτηση για το αποτέλεσμα και ανατρέχει στην repository και επιστρέφει τα αποτελέσματα εφόσον έχει ολοκληρωθεί η εργασία

2.4.3.3Reliability

Εάν υπήρχε μόνο ένας task dispatcher και διακόπηκε, όλα τα αποτελέσματα που έχουν επιστραφεί από τους workers σε εκείνο τον task dispatcher θα
χάνονταν. Επομένως, είναι απαραίτητο να υπάρχουν παραπάνω από ένας ενημερωμένοι task dispatchers σε κάθε task dispatcher peer groups. Με δύο task dispatchers που κρατούν ο ένας τον άλλον ενήμερο με τα πιο πρόσφατα αποτελέσματα που έχουν λάβει, οι πληροφορίες δεν χάνονται εάν ένας από αυτούς υφίστανται μια διακοπή λειτουργίας.



Εικόνα 11 ένας Worker node υιοθετεί το ρόλο του task dispatcher όταν ο τελευταίος αποτυγχάνει.

Ένας νέος worker που προσχωρεί σε μια ομάδα εργασίας δεν έρχεται σε επαφή με έναν ιδιαίτερο task dispatcher,αλλά με το task dispatcher peer group. Ο task dispatcher απαντά στο εισερχόμενο μήνυμα. Ο worker τότε καθιερώνει επικοινωνία
με το task dispatcher. Αυτό το πρωτόκολλο καθιέρωσης επικοινωνίας με το worker διευκρινίζεται στην κορυφή της εικόνας 11. Σε αυτό το πρότυπο, εάν ένας task dispatcher αποτυγχάνει να αποκριθεί σε έναν worker, ο worker επαναλαμβάνει πάλι το αρχικό βήμα επικοινωνώντας πάλι με το task dispatcher peer group. Αυτή τη φορά, ένας διαφορετικός task dispatcher αποκρίνεται στο αίτημά του. Αυτό το πρωτόκολλο σε περίπτωση αποτυχίας task dispatcher με το worker φαίνεται στο κατώτατο σημείο της εικόνας 11.

Οι Task dispatchers σε ένα peer group επικοινωνούν με την αποστολή μεταξύ τους μηνυμάτων σε τακτά χρονικά διαστήματα. Όταν οι task dispatchers λαμβάνουν τα νέα αποτελέσματα από έναν worker, τα στέλνουν στον άλλο task dispatcher για να κρατήσουν ένα περιττό αντίγραφο από αυτά τα αποτελέσματα.

Μόλις ανακαλύψει ο άλλος αποστολέας στην ίδια όμοια ομάδα ότι ο συνάδελφος του λείπει, θα προσκαλέσει έναν εργαζόμενο που ζητά μια εργασία να εκτελέσει το κώδικα του task dispatcher στην ομάδα του, που μετασχηματίζει έναν κανονικό εργαζόμενο σε task dispatcher. Αυτή η ανταλλαγή

ρόλου είναι απλή να εφαρμοστεί, επειδή και τα δύο, οι κώδικες εργαζομένων και task dispatcher , εφαρμόζουν μια κοινή διεπαφή, κάνοντας τους εξίσου schedulable σε αυτό το πρότυπο. Αυτή η ανταλλαγή ρόλου φαίνεται στην εικόνα 11 από τον εργαζόμενο στην αριστερή πλευρά της εικόνας

Ο αριθμός task dispatcher στην ομάδα task dispatchers δεν πρέπει απαραιτήτως να περιοριστεί σε δύο.

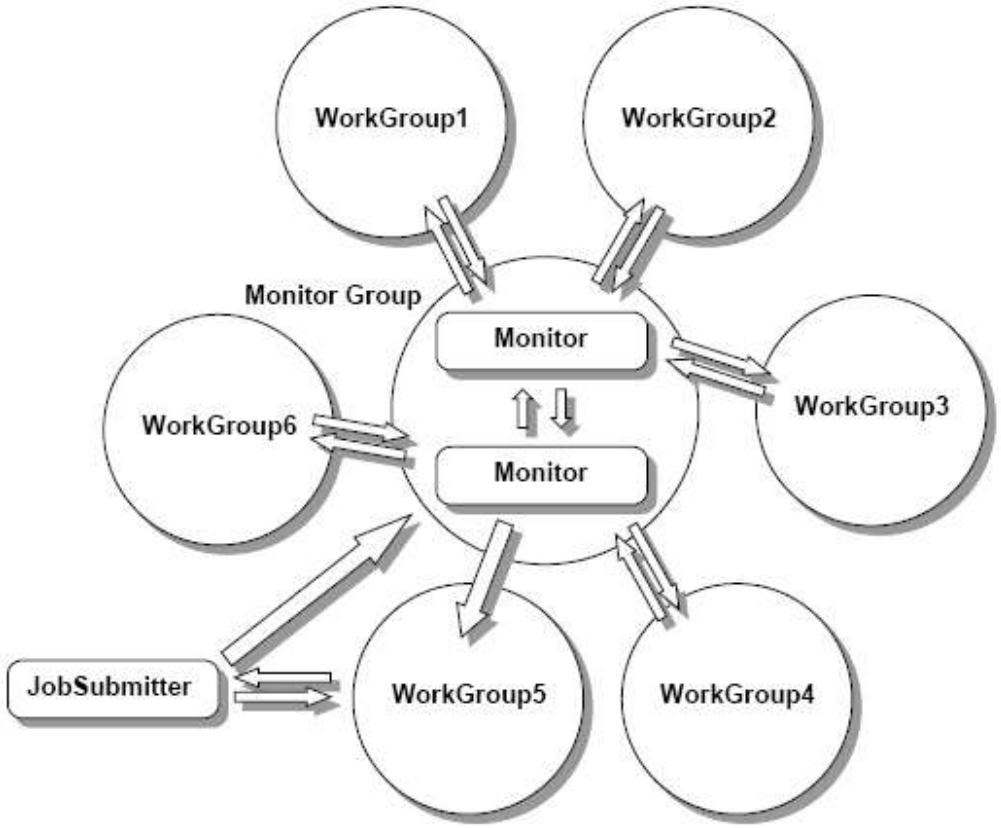
2.4.3.4Scalability

Για να αποτραπεί η δημιουργία ενός Bottleneck στην επικοινωνία των worker και του task dispatcher μια καινούργια έννοια δημιουργείται αυτή του Monitor.Η κύρια λειτουργία του μόνιτορ είναι να χειρίζεται την επικοινωνία των καινούργιων peer που δεν ανήκουν ακόμα σε κάποιο group.λειτουργούν δηλαδή ως ενδιάμεσοι μεταξύ των καινούργιων peer και των work group.Οι Job submitters οι ποιοι θέλουν να δώσουν μια εργασία και οι workers που θέλουν να ενταχθούν σε κάποιο work group πρέπει να επικοινωνήσουν με ένα monitor. Οι Monitors απελευθερώνουν τους task dispatchers από την απευθείας επικοινωνία με τον υπόλοιπο κόσμο. ένας monitor μπορεί να έχει πολλά διαφορετικά work groups και μπορεί να στέλνει αιτήματα από καινούργιους peers σε οποιοδήποτε από αυτά τα work groups που έχει .Η επιλογή του κατάλληλου work group γίνεται λαμβάνοντας υπ' όψιν το φορτίο του κάθε work group. όπως και με τα task dispatcher peer groups, υπάρχουν επίσης monitor peer groups, με πολλούς monitors που ενημερώνουν ο ένας τον άλλο .Οι Job submitters κάνουν αίτηση στο monitor peer group.Οι Monitors σε αυτό το peer group προωθούν αυτές τις αιτήσεις σε κάποιο work group. Η επιλογή του group εξαρτάται από τι κώδικα τρέχουν ήδη αυτά τα work groups από το φορτίο τους κ.τ.λ .Το work group απαντά απευθείας στο job submitter,ο οποίος δημιουργεί μια σχέση εργασίας με αυτό το work group.

Η προώθηση από το κορυφαίο monitor group συμβαίνει μόνο μια φορά κατά την αρχική ζήτηση από το job submitter για να δώσει μια εργασία. Κατόπιν, τα μηνύματα στέλνονται απευθείας από το job submitter στο σωστό work group.ένα παρόμοιο πρωτόκολλο ακολουθείτε και για την περίπτωση που ένας καινούργιος worker ζητά να ενταχθεί στο framework.

Ο ρόλος του monitor δεν είναι μόνο να κατευθύνει τους καινούργιους workers στο σωστό work group,αλλά επίσης να διευθύνει και τα work groups,επειδή η απόφαση της επιλογής του work group για κάποια εργασία είναι δική του υποχρέωση .Γι΄ αυτό κρατά πληροφορίες για το φορτίο των work group,τους κώδικες ,και την αστοχία κάποιου task dispatcher σε κάποιο work group.

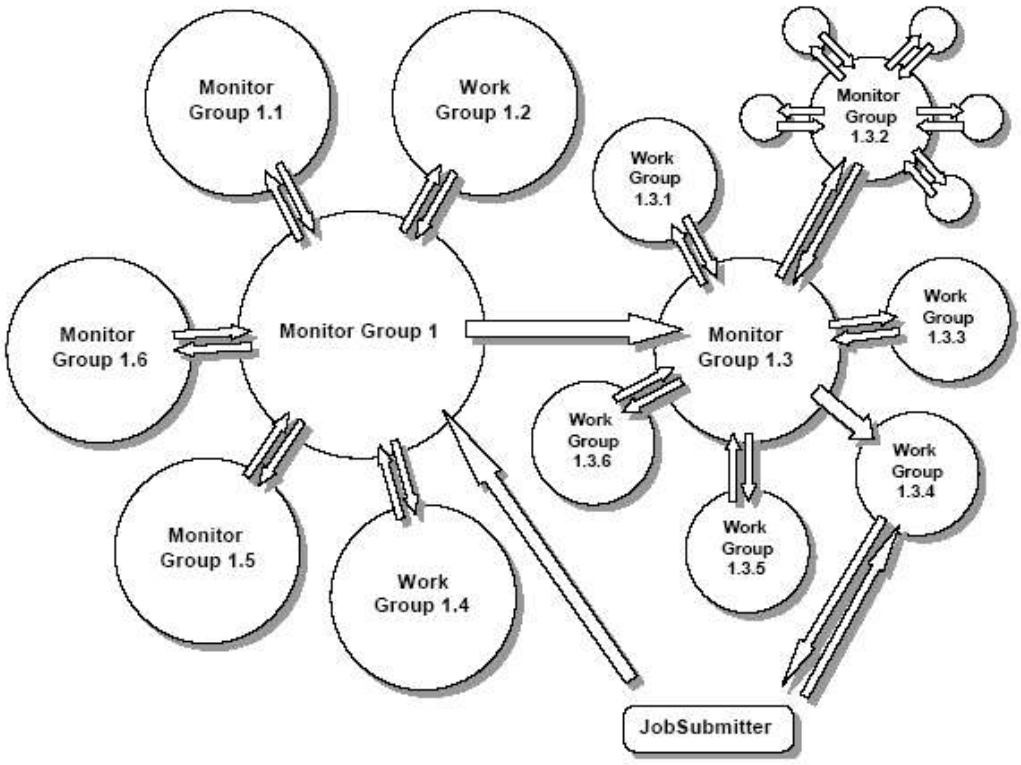
Οι Monitors επίσης μπορούν να ζητήσουν από κάποιο worker να γίνει monitor εάν υπάρχει ανάγκη.



Εικόνα 12 επικοινωνίες μεταξύ monitor και άλλων κόμβων.

Για να αποφύγουμε την περίπτωση ύπαρξης κάποιου bottleneck από την ύπαρξη μεγάλου αριθμού peers σε κάποιο group υπάρχει κάποια ιεραρχία στο μοντέλο των monitor peer group.επίσης δίνεται η δυνατότητα στο monitor group κάθε φορά που έχουμε υπερβολικά μεγάλο αριθμό από peers να διασπαστεί δημιουργώντας κάποιο καινούργιο monitor group.η διαδικασία της επικοινωνίας των job submitter γίνεται ως εξής:επικοινωνεί με το πρώτου επιπέδου monitor group, το monitor group αποφασίζει που θα προωθήσει το αίτημα και το προωθεί εάν αυτό το group στο οποίο προωθήθηκε δεν είναι work group αλλά monitor group τότε η αίτηση επαναπροωθείτε έως ότου εντοπίσει ένα work group ,τότε αυτό επικοινωνεί με το job submitter δίνοντας του το id του task dispatcher καθώς και τα ενδιάμεσα id τα οποία χρειάζονται για να φτάσει το μήνυμα σε αυτόν και τελικά επικοινωνεί με αυτόν.

Επειδή όλοι οι καινούργιοι peers που εντάσσονται στο δίκτυο είναι αναγκασμένοι να επικοινωνήσουν με το πρώτου επιπέδου monitor group, η επικοινωνία μπορεί να γίνει
bottleneck . Ένας απλός τρόπος για να αποφευχθεί αυτό είναι να υπάρχει ένας συγκεκριμένος αριθμός αιτήσεων που θα διεκπεραιώνει κάθε ένας monitor.



Εικόνα 13 δίκτυο από work groups και associated monitor groups.

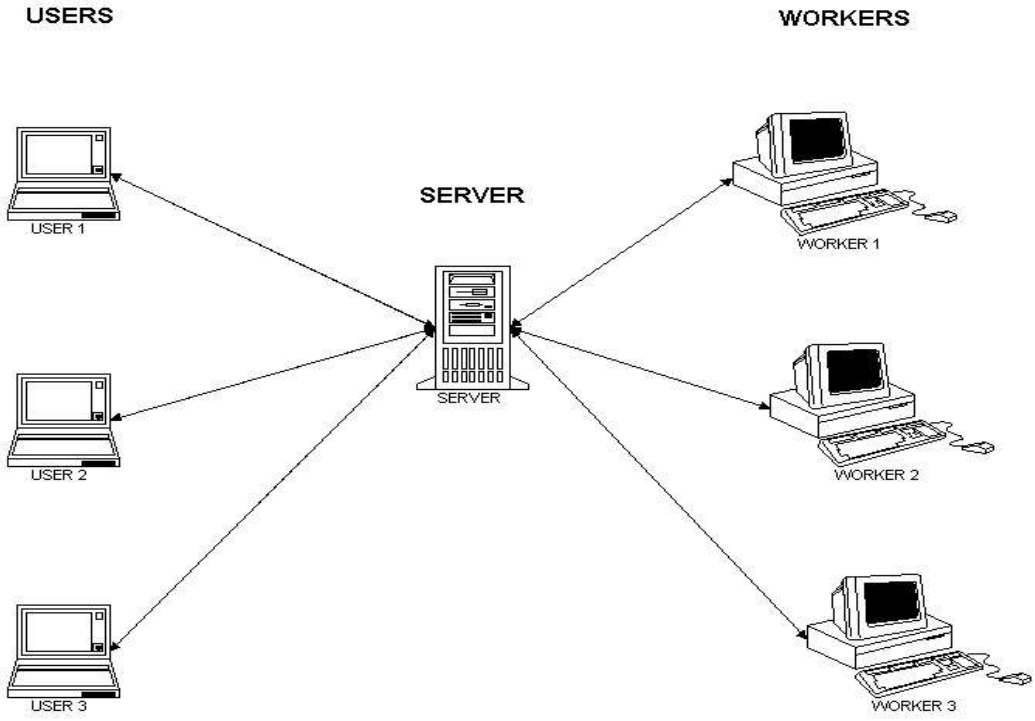
Ο τρόπος με τον οποίο θα ανατεθεί σε κάποιο monitor να απαντήσει στο αίτημα κάποιου καινούργιου peer ποικίλη ανάλογα με το σχεδιασμό του κάθε ενός .Για παράδειγμα μπορεί να γίνεται εντελώς αυτόματα ώστε να αποφύγουμε μια περαιτέρω επικοινωνία μεταξύ των monitors (εάν έχουμε δυο monitors μπορούμε να αναθέσουμε τα αιτήματα από peers με μόνο id στον έναν και στον άλλο τα ζυγά) ή μπορεί να γίνεται λαμβάνοντας υπ' όψιν και την γεωγραφική θέση του κάθε καινούργιου peer.

Κεφάλαιο 3 Η δικιά μας δουλειά

Το project το δικό μας δίνει την δυνατότητα σε κάποιο άτομο να χρησιμοποιήσει τον κώδικα μας έτσι ώστε να επιτύχει να κάνει υπολογισμούς για μια μεγάλη σε υπολογιστική δύναμη εργασία που θέλει λαμβάνοντας όμως υπ' όψιν πως η εργασία του θα μπορεί να σπάσει σε μικρότερα κομμάτια τα οποία θα διανεμηθούν σε workers οι οποίοι θα φέρουν εις πέρας τους επί μέρους υπολογισμούς και τελικά συνδέοντας όλα τα αποτελέσματα μεταξύ τους θα έχουμε το αποτέλεσμα της αρχικής εργασίας, όμως εντωμεταξύ θα έχουμε εξοικονομήσει πολύ χρόνο από το παράλληλο υπολογισμό των επί μέρους εργασιών .Με αυτό το τρόπο μπορούμε να υπολογίσουμε προβλήματα ,που κι αν ακόμα έτρεχαν σε ένα υπερυπολογιστή θα χρειαζόταν υπερβολικά μεγάλο χρόνο για να τελειώσουν ,μέσα σε ένα λογικό χρονικό διάστημα.

Ο κώδικας του project είναι γραμμένος σε python και για την υλοποίηση των επικοινωνιών και των interface έχουμε χρησιμοποιήσει την CORBA .Παρόλα' αυτά η όλη υλοποίηση δεν περιορίζει τους χρηστές να υλοποιήσουν τις εφαρμογές τους σε python καθώς με την χρήση της CORBA μπορούμε να επιτύχουμε την επικοινωνία και με εφαρμογές οι οποίες δεν είναι γραμμένες σε python.

Για να αντιληφθούμε όμως καλύτερα την λειτουργία αλλά και την αρχιτεκτονική του project θα πρέπει να αποσαφηνίσουμε κάποιες έννοιες πριν ακόμα προχωρήσουμε και αυτές οι έννοιες είναι του user,worker και server.



Εικόνα 14 Το client server μοντέλο μας

3.1 User

Είναι η ονομασία του χρηστή εκείνου ο οποίος τρέχει το κώδικα του user και θέλει να δώσει μια μεγάλη εργασία για υπολογισμό .Ο χρηστής αυτός δεν επικοινωνεί με το worker αλλά παρά μόνο με το server.παρέχει στο server το κώδικα ο οποίος θα τρέξει στους workers για να γίνει ο υπολογισμός των επιμέρους εργασιών καθώς και ότι αρχικά δεδομένα χρειάζονται για να αρχίσει ο υπολογισμός των επί μέρους εργασιών .Τα δεδομένα αυτά μπορούν να είναι πολύ μικρά σε μέγεθος ή ακόμα και ολόκληρα blocks από δεδομένα και μπορούν να σταλούν στον server είτε απευθείας μέσω της CORBA είτε μέσω αρχείων.

Ο user για να μπορέσει να δώσει κάποιο υπολογισμό στο server θα πρέπει να έχει υλοποιήσει δυο προγράμματα.

- Στο πρώτο πρόγραμμα θα πρέπει να έχει υλοποιήσει μια συνάρτηση την οποία θα πρέπει να έχει ονομάσει workflow και η οποία δεν θα παίρνει κάποιο όρισμα αλλά θα επιστρέφει μια λίστα με το όνομα των αρχείων εκείνων που περιέχουν τα δεδομένα για τις επί μέρους εργασίες και που είναι αναγκαία να σταλούν με http στον server , σε περίπτωση που δεν υπάρχουν τέτοια αρχεία απλά επιστρέφει μια κενή λίστα . Μέσα στην συνάρτηση αυτή θα πρέπει να έχει υλοποιηθεί ο κώδικας ο οποίος θα χρησιμοποιηθεί για να δημιουργήσει όλες τις εργασίες που θα σταλούν στον server για να δοθούν στην συνέχεια στους workers.

- Στο δεύτερο πρόγραμμα θα πρέπει να έχει υλοποιήσει μια συνάρτηση την οποία θα πρέπει να έχει ονομάσει Task και η οποία δεν θα παίρνει κάποιο όρισμα αλλά θα επιστρέφει μια λίστα με τα αποτελέσματα του υπολογισμού που έκανε ,Έτσι κάθε στοιχείο της λίστας θα είναι της μορφής arg.Στην συνάρτηση αυτή θα πρέπει να έχει υλοποιηθεί ο κώδικας ο οποίος θα στέλνεται στους workers για να μπορέσουν να κάνουν τον εκάθωστε υπολογισμό.

Στην συνέχεια εκτελώντας το πρόγραμμα user.py και δίνοντας τα ονόματα των δυο αρχείων με τις πιο πάνω συναρτήσεις το πρόγραμμα θα κάνει δυο http κλήσεις για να μπορέσει να πάρει το αρχείο provider_fileior.txt το οποίο περιέχει το IOR για να μπορέσει να επικοινωνήσει ο user με το server και το αρχείο comunication_idl το οποίο είναι απαραίτητο για να μπορεί να χρησιμοποιήσει το interface ο user και να κάνει τις απαραίτητες κλήσεις στο server.

Το πρόγραμμα user.py είναι εκείνο το οποίο θα κάνει τις κλήσεις στον server και με το οποίο τελικά θα αναζητήσει και θα λάβει από τον server το αποτέλεσμα των υπολογισμών.

Σε περίπτωση που ο user θέλει να υλοποιήσει τον κώδικα του σε κάποια άλλη γλώσσα προγραμματισμού εκτός της python και για την οποία υπάρχει mapping με την CORBA θα πρέπει να υλοποιήσει και το αντίστοιχο πρόγραμμα user.py το οποίο βάση του Interface για τον user θα κάνει τις αντίστοιχες κλήσεις στον server και με το οποίο τελικά θα λάβει τα αποτελέσματα του υπολογισμού που έχει δώσει στον server.Η CORBA παρέχει mapping για μια πληθώρα γλωσσών προγραμματισμού και μερικές από αυτές είναι c,java,c++,….κ.α.

Σημαντικό είναι να αναφέρουμε πως κατά την εκκίνηση του προγράμματος του κάθε user ο χρηστής είναι απαραίτητο να γνωρίζει το ip του server έτσι ώστε να το περάσει ως όρισμα στο πρόγραμμα, για να μπορέσει να κάνει http κλήσεις προς αυτόν και να στείλει αλλά και να λάβει όσα αρχεία είναι απαραίτητα

3.2 Το interface για τον user

Οι διαθέσιμες κλήσεις για τον user είναι οι εξής:

repository	Κλήση για την ανάθεση ενός url από το server στο οποίο θα αποθηκεύονται τα δεδομένα και τα εκτελέσιμα του user.
put_dict	Κλήση για την παράδοση ενός dictionary, με όλες τις επιμέρους εργασίες ,από τον user στον server.
get_result	Κλήση με την οποία ζητά από το server ένα σύνολο αποτελεσμάτων για κάποιες επιμέρους εργασίες.
quit	Κλήση με την οποία αποσυνδέεται ο user από τον server.

Ο αντίστοιχος κώδικας:

```
interface computation{
        url repository();
        void put_dict(in dict s3);
        args get_result(in jid_t s4);
        void quit();
        };
```

Κάνοντας κλήση στην συνάρτηση repository ο user ζητά από τον server να του δοθεί ένα μοναδικό url στο οποίο στη συνέχεια θα τοποθετήσει τον κώδικα του προγράμματος που είναι απαραίτητος για να γίνει ο υπολογισμός στη μεριά του worker καθώς και τα δεδομένα για τον υπολογισμό εάν έχουν την μορφή αρχείων και δεν αποστέλλονται απευθείας από την CORBA.

Κάνοντας κλήση στην συνάρτηση put_dict ο user παρέχει στον server ένα dictionary στο οποίο υπάρχουν όλες οι εργασίες τις οποίες πρέπει να παραδώσει ο server στους worker για υπολογισμό.

Κάνοντας κλήση στην συνάρτηση get_result και δίνοντας ένα string της μορφής: (ονομα εργασιας : 0 ή 1,....,όνομα εργασίας : 0 ή 1) όπου 0 ή 1 αντίστοιχα συμβολίζει το πως αναμένει να του δοθούν τα αποτελέσματα των αντιστοίχων εργασιών από τον server (0 Μέσω της CORBA και 1 μέσω αρχείων και http).Αυτές είναι οι εργασίες που ενδιαφέρουν τον user να λάβει τα αποτελέσματα τους και είναι λογικό να είναι λιγότερες από τον συνολικό αριθμό των εργασιών που δίνει στον server αφού τα ενδιάμεσα αποτελέσματα εργασιών είναι απαραίτητα για την συγκρότηση των δεδομένων των εξαρτημένων εργασιών που δίνονται στους workers και μπορεί να μην τον ενδιαφέρουν άμεσα.

Τέλος κάνοντας κλήση στη συνάρτηση quit ο user αποσυνδέεται από τον server αφού έχει λάβει όλα τα αποτελέσματα των εργασιών που τον ενδιαφέρουν.

3.3 Worker

Είναι η ονομασία του χρηστή εκείνου ο οποίος τρέχει το κώδικα του worker και θέλει να λάβει μέρος στον υπολογισμό ενός μεγάλου προβλήματος υπολογίζοντας μια επί μέρους εργασία του όλου υπολογισμού. Ο χρηστής αυτός δεν επικοινωνεί με το user αλλά παρά μόνο με το server.Παίρνει από το server το κώδικα εφαρμογή για να γίνει ο υπολογισμός της εργασίας και όλα εκείνα τα δεδομένα τα οποία χρειάζεται για να ξεκινήσει τους υπολογισμούς .αφού τελειώσει τον υπολογισμό της εργασίας που του έχει δοθεί τότε αμέσως επιστρέφει το αποτέλεσμα της εργασίας στον server.

Στο κάθε worker θα πρέπει να υπάρχει ένα πρόγραμμα worker.py το οποίο είναι εκείνο το οποίο θα λάβει τα δεδομένα από το server για κάθε εργασία και κάνοντας τις αντίστοιχες κλήσεις.

Αρχικά θα λάβει δυο αρχεία το πρώτο είναι το fileior.txt το οποίο είναι απαραίτητο για να υπάρξει η επικοινωνία με το server καθώς παρέχει το IOR .Στη συνέχεια το αρχείο comnunication_idl το οποίο παρέχει τις υλοποιήσεις των interface έτσι ώστε να μπορέσει να κάνει τις απαραίτητες κλήσεις.
Το πρόγραμμα αυτό, αφού θα έχει λάβει τα δεδομένα καθώς και των κώδικα για τον υπολογισμό, θα εκτελέσει την συνάρτηση του υπολογισμού και θα στείλει στον server το αποτέλεσμα.

Επειδή θέλουμε ο κάθε worker να εκτελεί μια φορά το κώδικα και στην συνέχεια να τρέχει στο backround ως deamon κάνοντας όσο το δυνατόν περισσότερους υπολογισμούς έχουμε υλοποιήσει ένα πρόγραμμα το deamon.py το οποίο αναλαμβάνει να κάνει αυτό ακριβώς το πράγμα.

Σημαντικό είναι να αναφέρουμε πως κατά την εκκίνηση του προγράμματος του κάθε worker ο χρηστής είναι απαραίτητο να γνωρίζει το ip του server έτσι ώστε να το περάσει ως όρισμα στο πρόγραμμα, για να μπορέσει να κάνει http κλήσεις προς αυτόν και να λάβει όσα αρχεία του είναι απαραίτητα

Επίσης κάθε φορά που τελειώνει μια εργασία και αφού έχει λάβει κάποια αρχεία από τον server ως δεδομένα αυτά τα αρχεία διαγράφονται από το χώρο του worker.

3.4 Το interface για τον worker

Οι διαθέσιμες κλήσεις για τον worker είναι οι εξής:

code	Κλήση με την οποία αποδίδεται στον worker ένα url για την αναζήτηση των δεδομένων και των εκτελέσιμων.
data	Κλήση με την οποία του δίδονται τα δεδομένα τα οποία θα χρειαστεί για τον υπολογισμό.
put_result	Κλήση με την οποία παραδίδει το αποτέλεσμα της επιμέρους εργασίας στον server.
quit	Κλήση με την οποία αποσυνδέεται από τον server.

Ο αντίστοιχος κώδικας:

```
interface session{
                url code();
                args data(in string code);
                void put_result(in args s1);
                void quit();
                };
```

Κάνοντας κλήση στη συνάρτηση code επιστρέφετε στον Worker ένα url βάση του οποίου στη συνέχεια θα αναζητήσει μέσω http τον κώδικα για να μπορέσει να κάνει τον υπολογισμό .Ουσιαστικά με αυτό τον τρόπο ο συγκεκριμένος worker εντάσσεται στο δυναμικό του user που του έχει αποδοθεί αυτό το url που έλαβε ως απάντηση ο worker.

Κάνοντας κλήση στη συνάρτηση data και δίνοντας ως όρισμα το url που έχει λάβει από την κλήση της συνάρτησης code λαμβάνει ως απάντηση τα δεδομένα για τον υπολογισμό .Τα δεδομένα αυτά μπορεί να περιέχουν το όνομα κάποιον αρχείων που περιέχουν τα δεδομένα τα οποία τα παραλαμβάνει μέσω http έχοντας το url και το όνομα των αρχείων

Κάνοντας κλήση στη συνάρτηση put_result και δίνοντας ως όρισμα το αποτέλεσμα του υπολογισμού που πραγματοποιήσει αποστέλλει στον server το αποτέλεσμα του υπολογισμού .Και πάλι το αποτέλεσμα μπορεί να είναι γραμμένο σε κάποιο αρχείο οπότε σε αυτή τη περίπτωση αποστέλλει μέσω http το αρχείο στον Server.

Τέλος κάνοντας κλήση στη συνάρτηση quit ο worker αποσυνδέεται από τον server αφού πρώτα έχει επιστρέψει το αποτέλεσμα στον server.

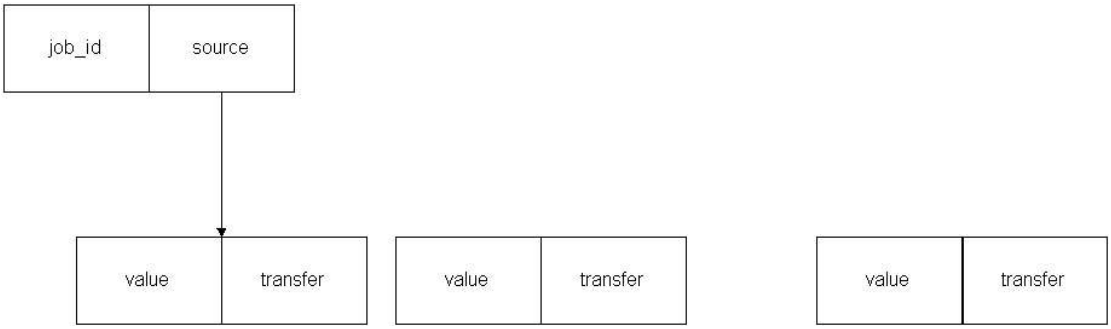
3.5 Server

Είναι η ονομασία του χρηστή εκείνου ο οποίος τρέχει το κώδικα του server και έχει ως υποχρέωση να δέχεται τις εργασίες και τα δεδομένα από τους users και να διαθέτει την εφαρμογή του υπολογισμού και τα δεδομένα σε κάθε worker ο οποίος θέλει να λάβει μέρος σε αυτό τον υπολογισμό .

Ο server χρησιμοποιεί ένα αποθηκευτικό χώρο όπου εκεί αποθηκεύει τα αποτελέσματα που λαμβάνει από τους workers αλλά και τα δεδομένα και τα εκτελέσιμα που λαμβάνει από τους users.Σε κάθε user αναθέτεται ένας φάκελος μέσα στον οποίο αποθηκεύονται όλα τα δεδομένα και τα αποτελέσματα που αφορούν τον υπολογισμό .επικοινωνεί απευθείας και με τους users και με τους workers.

3.6 Η δομή των εργασιών

Οι εργασίες που δημιουργεί ο κάθε user έχουν τη μορφή μιας δομής δυο μεταβλητών των job_id τύπου string ,η οποία αντιστοιχεί στην ονομασία της εργασίας(π.χ. j1),και της source τύπου args, το οποίο args είναι μια σειρά (λίστα) από μια καινούργια δομή δυο μεταβλητών που αντιστοιχούν στα δεδομένα της εκάθωστε εργασίας .Η πρώτη μεταβλητή είναι η value τύπου string και αντιστοιχεί ανάλογα, είτε απευθείας στα δεδομένα της εργασίας είτε στο όνομα του αρχείου που περιέχει τα δεδομένα .Ο τρόπος που πρέπει να γραφτεί το όνομα του αρχείου που περιέχει τα δεδομένα είναι συγκεκριμένος και έχει την μορφή #name_of_file .Η μεταβλητή transfer είναι τύπου boolean και είναι αυτή που καθορίζει εάν η μεταβλητή value περιέχει απευθείας τα δεδομένα(transfer=0) ή το όνομα του αρχείου που βρίσκονται τα δεδομένα (transfer=1).



Εικόνα 15 η μορφή εγγραφής των δεδομένων

Οι μεταβλητές που αναπαριστούν τα δεδομένα είναι:

Job_id	Το όνομα της επιμέρους εργασίας ,μια μεταβλητή τύπου string.
source	Μια σειρά δεδομένων τα οποία χρειάζεται αυτή η εργασία ώστε να υπολογιστεί, μια μεταβλητή τύπου arg[1].
value	Μια τιμή ή αντίστοιχα ένα αρχείο στο οποίο βρίσκεται η τιμή αυτή, μια μεταβλητή τύπου string.
transfer	Μια τιμή (0 ή 1) η οποία αντιστοιχεί στη μορφή[2] που πρέπει να σταλεί η τιμή της value μεταβλητης. Η μεταβλητή είναι τύπου Boolean.

Ο αντίστοιχος κώδικας σε IDL παρουσιάζεται πιο κάτω.

```
typedef string jid_t;
typedef sequence <arg> args;
struct job{
                jid_t job_id;
                args source;
        };
struct arg{
                string value;
                boolean transfer;
        };
```

3.7 Η δομή των αποτελεσμάτων

[1] Μια δομη η οποια περιεχει τις μεταβλητες value και transfer.
[2] Εάν πρεπει να σταλει μεσω http πρωτοκολλου ή μεσω της corba.

Ο user λαμβάνει από το server τα αποτελέσματα για τις εργασίες που ζητά με τη μορφή μιας λίστας όπου κάθε στοιχείο της είναι μια δομή job και αντιστοιχεί στην αντίστοιχη εργασία που έχει αναζητήσει το αποτέλεσμα της ο user.

Ο worker στέλνει στον server το αποτέλεσμα της εργασίας που έχει υπολογίσει με τη μορφή μιας λίστας τόσων στοιχείων όσα και τα αποτελέσματα που πρέπει να επιστρέψει για την συγκεκριμένη εργασία που του έχει δοθεί .Σημαντικό είναι να σημειωθεί πως ο worker δεν γνωρίζει το όνομα της εργασίας που του έχει δοθεί καθώς από τον server δεν του στέλνεται κανένα τέτοιο στοιχείο και γι' αυτό και το κάθε στοιχείο στην λίστα των αποτελεσμάτων που επιστρέφει στο server δεν είναι της μορφής job αλλά της μορφής arg.

3.8 Distribution of tasks amongst workers

Η διαδικασία της διανομής των work units είναι αποκλειστική υποχρέωση του server και αυτή η διανομή βασίζεται στην υλοποίηση ενός αλγόριθμου στο κώδικα του server και αυτός ο αλγόριθμος είναι ο round robin.

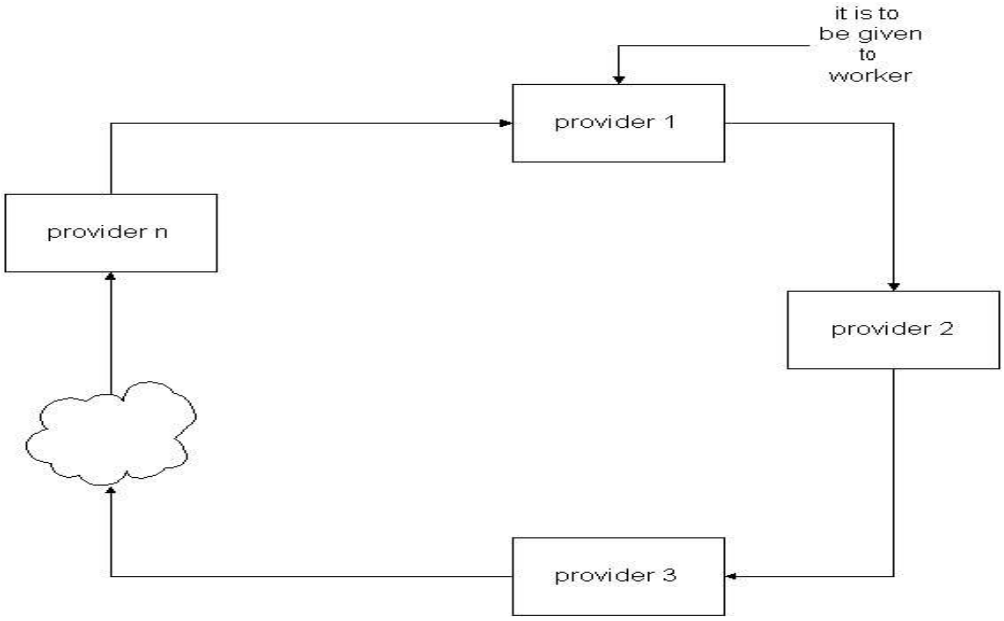
Υπάρχει μια κυκλική απλά συνδεδεμένη λίστα στην οποία κάθε φορά αφού έχουμε μια κλήση από ένα user και αφού εκείνος στείλει το dictionary με όλες τις εργασίες που θέλει να υπολογιστούν καταχωρούμε σ' αυτήν το url το οποίο έχουμε δώσει σ' αυτόν τον user.

Οι εσωτερικές συναρτήσεις που παρέχει αυτή η κυκλική απλά συνδεδεμένη λίστα είναι οι εξής:

add_node	Καλείται εσωτερικά από το πρόγραμμα για να προσθέσει ένα κόμβο[3] στην λίστα.
rm_node	Καλείται εσωτερικά από το πρόγραμμα για να διαγράψει ένα κόμβο από την λίστα.
give_node	Καλείτε εσωτερικά από το πρόγραμμα για να δώσει τον τρέχον κόμβο.

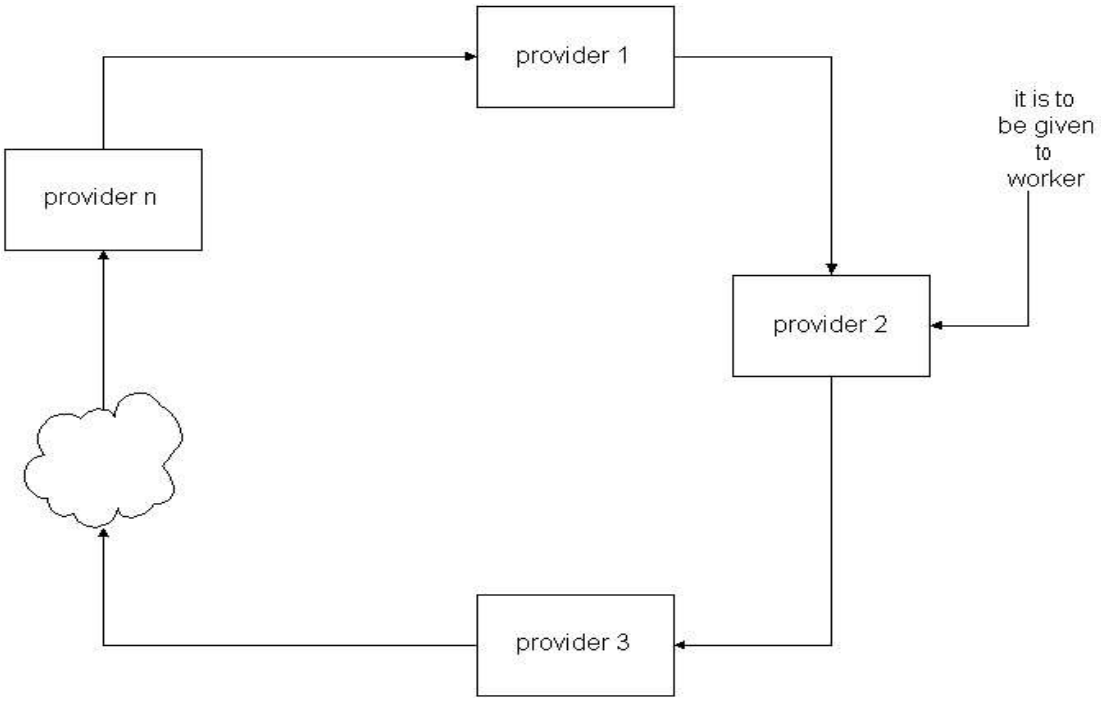
Αρχικά στο πρώτο worker που θα προσφερθεί να εκτελέσει έναν υπολογισμό αποδίδεται σ' αυτόν το πρώτο στοιχείο της λίστας δηλαδή του δίνεται το url που έχει ανατεθεί στο πρώτο user έτσι ώστε να ανατρέξει και να αναζητήσει το κώδικα του υπολογισμού αλλά και τα απαραίτητα δεδομένα.

[3] Ο κομβος αυτος ουσιαστικα παριστα το url που δωθηκε από το προγραμμα στον user.



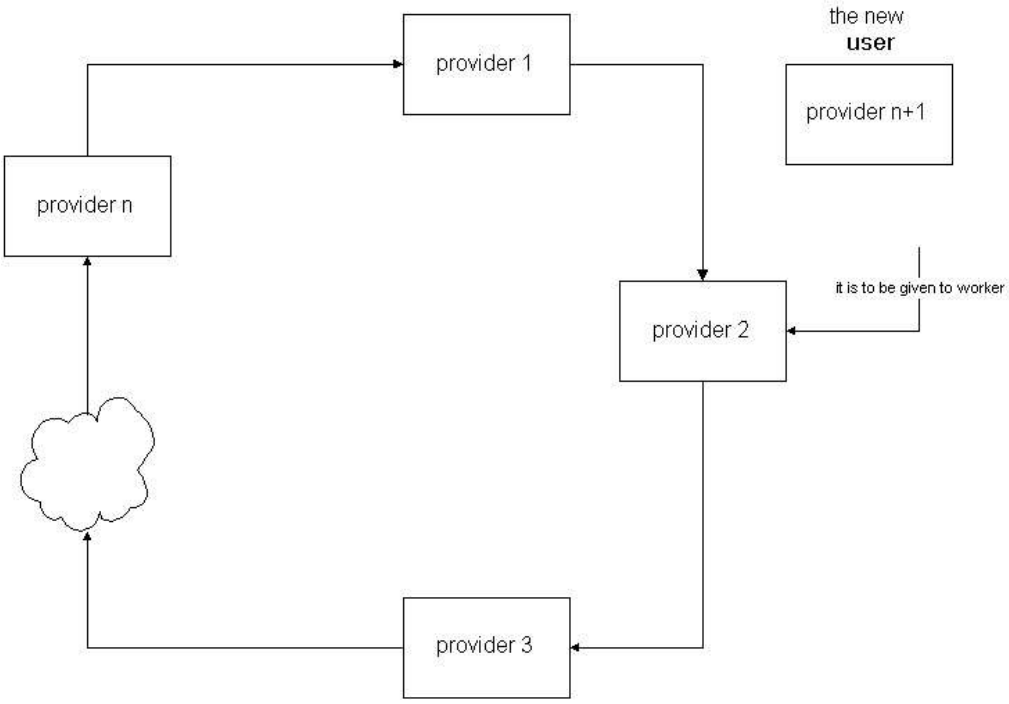
Εικόνα 16 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -1

Στον επόμενο worker θα αποδοθεί το αμέσως επόμενο στοιχείο από εκείνο που έχει ήδη δοθεί από την λίστα και ούτω κάθε εξής.



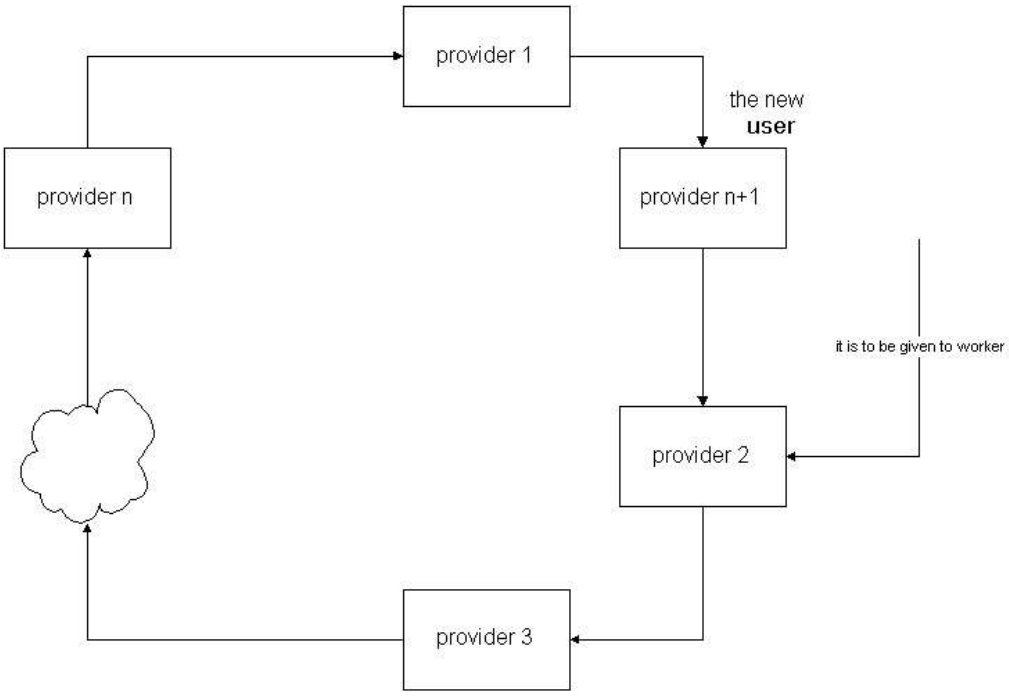
Εικόνα 17 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -2

Στην περίπτωση που κάποιος καινούργιος user θέλει να δώσει έναν υπολογισμό



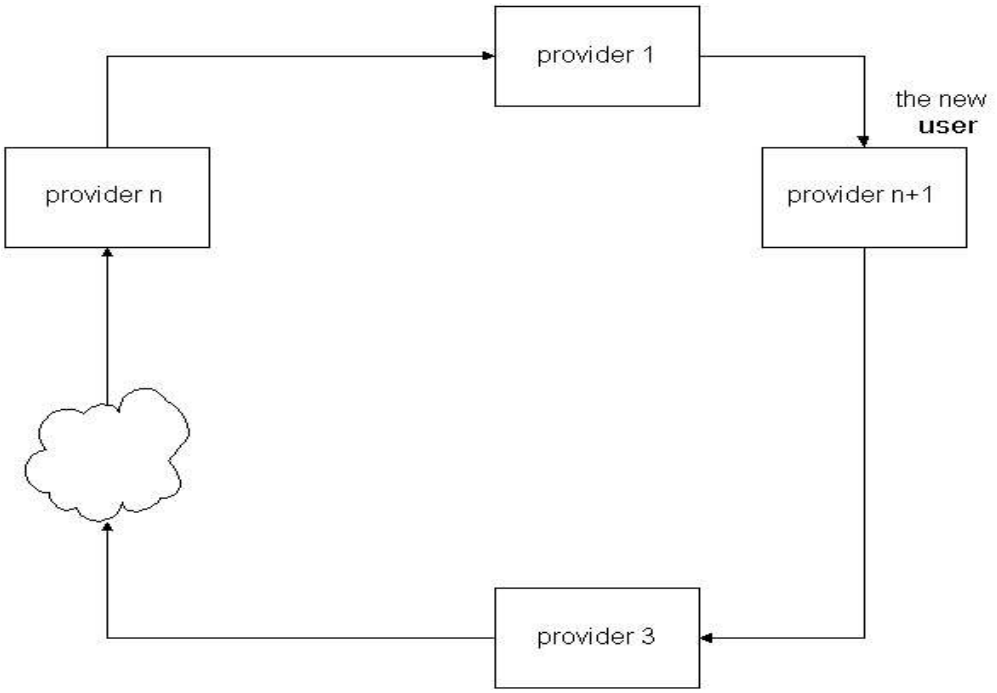
Εικόνα 18 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -3

τότε η θέση που θα λάβει στη λίστα θα είναι η τελευταία



Εικόνα 19 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -4

κάθε φορά που θα επιστρέφουν όλες οι εργασίες τότε και μόνο θα υπάρχει η διαγραφή του αντίστοιχου url από την λίστα αυτή.



Εικόνα 20 Η λειτουργία της διανομής και απόθεσης καινούργιων εργασιών -5

Σημαντικό είναι να αναφέρουμε πως η υλοποίηση δίνει προσοχή στο να μην υπάρχει κάποιος worker ο οποίος να ζητά να εκτελέσει υπολογισμό και να μην του δίνεται ,γιατί δεν υπάρχει διαθέσιμος εκείνη τη στιγμή, έτσι ακόμα και να μην είναι διαθέσιμος εκείνη τη στιγμή κάποιος υπολογισμός του δίνεται ξανά κάποια άλλη εργασία η οποία έχει σταλεί αλλά δεν έχει ολοκληρωθεί μέχρι εκείνη τη στιγμή κατά την οποία ζητά εργασία .Με αυτό το τρόπο καλύπτουμε και τις περιπτώσεις όπου κάποιος worker έχει λάβει μια εργασία και αποτυγχάνει να επιστρέψει το αποτέλεσμα αλλά και εάν κάποιος worker έχει λάβει μια εργασία αλλά αργεί υπερβολικά να υπολογίσει και να στείλει το αποτέλεσμα του με αποτέλεσμα να αργοπορεί και τον συνολικό χρόνο υπολογισμού της εργασίας καθώς λόγω των εξαρτήσεων μεταξύ των υπολογισμών μπορεί να μην μπορεί να προχωρήσει η συνολική εργασία.

3.9 Result retrieval

Ο κάθε user είναι υπεύθυνος να αναζητήσει το αποτέλεσμα της εργασίας του από τον server και για αυτό το λόγο στον κώδικα του έχει προβλεφθεί να αναζητά το αποτέλεσμα μιας work unit ή περισσοτέρων ανά ένα συγκεκριμένο χρονικό διάστημα το οποίο μπορεί να το καθορίσει ο ίδιος ο user.

3.10 Comnunication

Η επικοινωνία γίνεται με δυο τρόπους ο πρώτος είναι μέσο http όπου κάθε επικοινωνία έχει ως μορφή την ανταλλαγή αρχείων και ο δεύτερος είναι μέσω της CORBA και του πρωτοκόλλου που προσφέρει το οποίο πρωτόκολλο είναι το IIOP/GIOP.

Για την επικοινωνία υλοποιήσαμε ένα http server ο οποίος τρέχει στον χώρο του server και εξυπηρετεί όποιες κλήσεις δέχεται είτε από users είτε από workers.Οι διαθέσιμες του κλήσεις είναι οι get και put οι οποίες δέχονται ως όρισμα ένα url στο οποίο βρίσκεται το αρχείο και επιστρέφουν το ζητούμενο αρχείο.

get	Κλήση με την οποία ζητά κάποιο συγκεκριμένο αρχείο από τον http server δίνοντας το αντίστοιχο url του server.
put	Κλήση με την οποία δίνει ένα συγκεκριμένο αρχείο στον http server για να το τοποθετήσει στο δοσμένο url του server.

3.11 Garbage Collection

Όλα τα δεδομένα τα οποία χρησιμοποιεί ο worker για να εκτελέσει τους υπολογισμούς ,αφού ολοκληρωθεί ο υπολογισμός και σταλεί το αποτέλεσμα στον server διαγράφονται από τον χώρο όπου τοπικά είχαν αποθηκευτεί στον worker.Οι κώδικες και τα εκτελέσιμα προγράμματα τα οποία είχαν αποθηκευτεί στον worker δεν διαγράφονται καθώς εάν ο worker λάβει μέρος ξανά στον ίδιο υπολογισμό να μην χρειάζεται να του αποσταλεί ξανά ο κώδικας του υπολογισμού.

Κεφάλαιο 4 Τα πειράματα

Στα πειράματα που κάναμε χρησιμοποιήσαμε δυο πίνακες με διαστάσεις 3000χ3000 τους οποίους πολλαπλασιάσαμε με διάφορους τρόπους και καταγράψαμε το συνολικό χρόνο που χρειάστηκε για να γίνει ο πολλαπλασιασμός.

4.1 Γινόμενο πινάκων

Για να γίνει αυτός ο πολλαπλασιασμός εκτός από τον απευθείας υπολογισμό χρησιμοποιήσαμε και έναν αριθμητικό αλγόριθμο γινομένου ο οποίος λειτουργεί ως εξής:

Δοθέντων δυο πινάκων X,Y, διαστάσεως nxn, το γινόμενο τους Z, ορίζεται ως

$$Z[i,j]=X[i]*Y[j] = \prod_{k\theta}^{n\ 1} X[i,k]Y[k,j]$$

Εφαρμόζοντας την τεχνική του διαίρει και βασίλευε και υποθέτοντας πως το n είναι δύναμη του δυο η παραπάνω σχέση μπορεί να γραφτεί με τη μορφή υποπινάκων διαστάσεως (n/2)x(n/2) ως εξής:

$$\begin{bmatrix} P & Q \\ R & S \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} * \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{bmatrix}$$

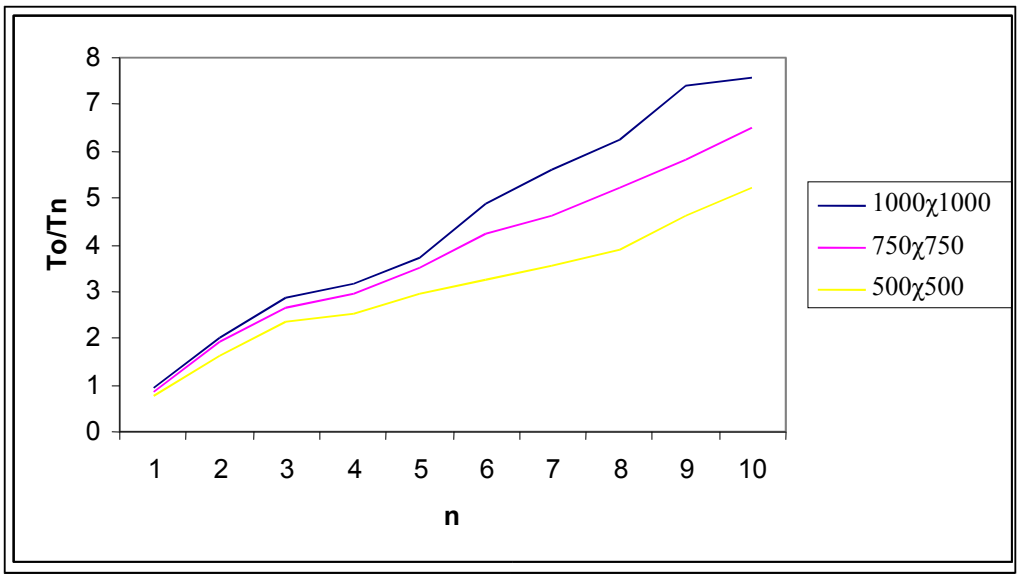
Έτσι βάση αυτής της τεχνικής πήραμε μετρήσεις για διάφορες διαστάσεις υποπινάκων και ειδικότερα για τον υπολογισμό των δυο πινάκων με διαστάσεις 3000χ3000 χρησιμοποιήσαμε υποπίνακες διαστάσεων 1000χ1000 , 750χ750 και 500χ500.

Λόγο της ανομοιομορφίας των χαρακτηριστικών των υπολογιστών που υπήρχαν στο εργαστήριο και έλαβαν μέρος στο πείραμα για να είναι πιο αντιπροσωπευτικές οι τιμές πήραμε την περίπτωση όπου έχουμε δώσει δέκα φορές την εργασία και στην συνέχεια τον συνολικό χρόνο τον διαιρούμε με τον αριθμό των δοθέντων υπολογισμών δηλαδή το δέκα και απλά το χρόνο με μια φορά να έχει δοθεί η εργασία.

Στη γραφική παράσταση που φαίνεται πιο κάτω έχουμε την περίπτωση όπου δέκα υπολογισμοί γινομένου πινάκων με διαστάσεις 3000χ3000 ο καθένας , έχουν σπάσει, ο καθένας υπολογισμός , σε υποπίνακες με διαστάσεις 1000χ1000 και έχουν δοθεί στον server για την προωθήσει σε workers των εργασιών Έτσι ώστε να λάβουμε το αποτέλεσμα των υπολογισμών .Το ίδιο έχει συμβεί και στη περίπτωση που οι υποπίνακες έχουν διαστάσεις 750χ750 και 500χ500.

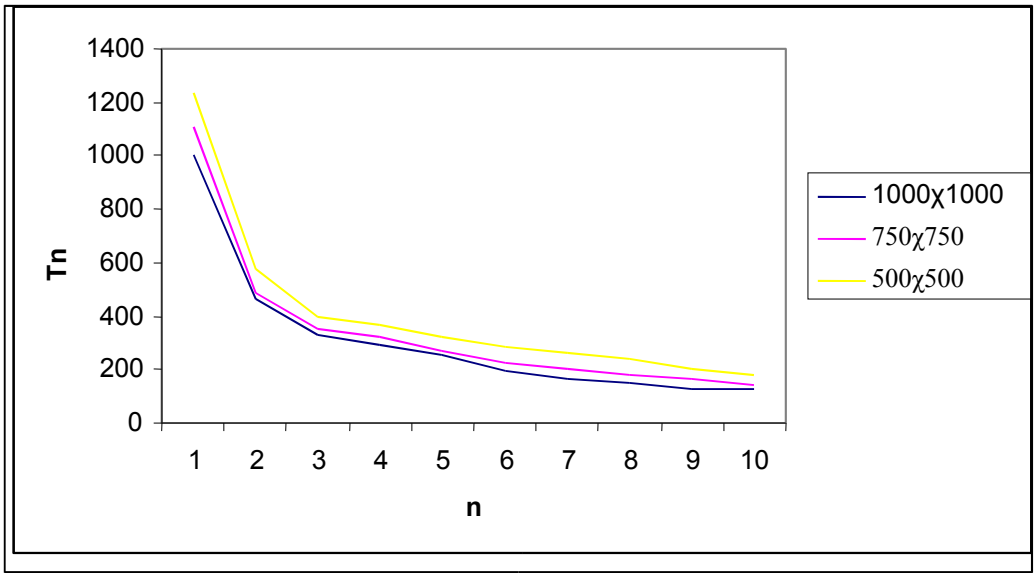
Οι τιμές του χρόνου που χρειάστηκε για να ολοκληρωθεί ο υπολογισμός διαιρεμένες με το δέκα (καθώς τόσοι ήσαν οι users) αντιπροσωπεύουν το Tn όπου n είναι ο αριθμός των υπολογιστών που είχαμε στην διάθεση μας ως workers.Ως To είναι ο χρόνος που χρειάστηκε για να υπολογιστεί το γινόμενο απευθείας σε έναν υπολογιστή και χωρίς να έχει γίνει κάποιο σπάσιμο σε υποπίνακες.

Ο x άξονας αντιπροσωπεύει τον αριθμό των workers και ο y άξονας αντιπροσωπεύει το πηλίκο της διαίρεσης του χρόνου To με το χρόνο Tn .



Εικόνα 21 το μέσο speedup για υπολογισμό, με σπασίματα πινάκων 1000χ1000,750χ750 και 500χ500, με δοσμένες εργασίες από 10 διαφορετικούς users ως προς τους workers.

Στην επόμενη γραφική παράσταση έχουμε ακριβώς τις ίδιες μετρήσεις με την παραπάνω γραφική παράσταση αλλά έχουμε διαφορετικό y άξονα και έτσι σε αυτή την περίπτωση ο y άξονας αντιπροσωπεύει το Tn (σε seconds) δηλαδή το μέσο χρόνο που χρειάστηκε για να ολοκληρωθεί το γινόμενο των πινάκων ταυτόχρονα από δέκα users που έδωσαν τον ίδιο υπολογισμό.



Εικόνα 22 οι μέσοι χρόνοι που χρειάστηκαν για να ολοκληρωθούν 10 σταλμένες εργασίες, για σπασίματα πινάκων 1000χ1000,750χ750 και 500χ500 ,από users ως προς τους workers.

Οι αναλυτικοί χρόνοι που μετρήσαμε και εξαγάγαμε τις πιο πάνω γραφικές είναι :

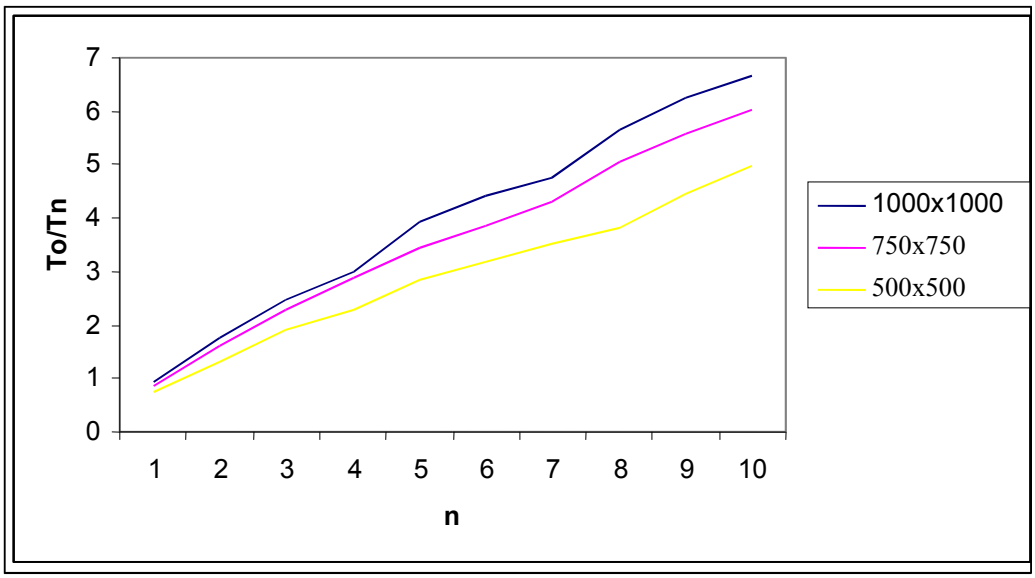
Για την γραφική 21

n	1000x1000 To/Tn	750x750 To/Tn	500x500 To/Tn
1	0,94	0,84991	0,759903
2	2,030238	1,918367	1,640489
3	2,848485	2,647887	2,367758
4	3,186441	2,9375	2,540541
5	3,730159	3,520599	2,9375
6	4,895833	4,234234	3,263889
7	5,595238	4,607843	3,54717
8	6,266667	5,222222	3,884298
9	7,401575	5,802469	4,630542
10	7,580645	6,482759	5,222222

Για την γραφική 22

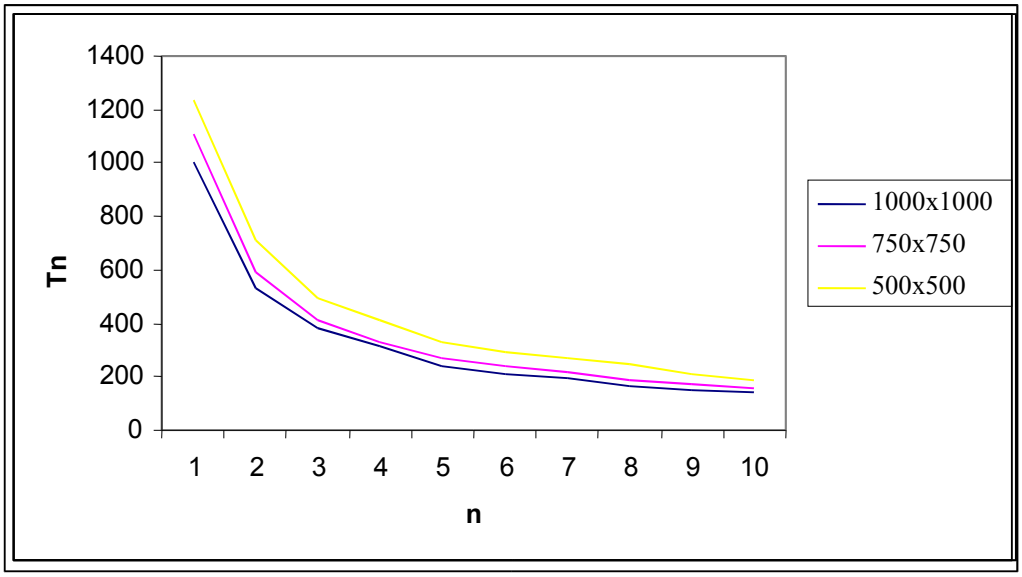
n	1000x1000 Tn	750x750 Tn	500x500 Tn
1	1000	1106	1237
2	463	490	573
3	330	355	397
4	295	320	370
5	252	267	320
6	192	222	288
7	168	204	265
8	150	180	242
9	127	162	203
10	124	145	180

Στην επόμενη γραφική δεν έχουμε πλέον δέκα users αλλά έναν και πάλι ο y άξονας όπως και στην γραφική 1 αντιπροσωπεύει το To/Tn και ο x άξονας το n και οι περιπτώσεις των υποπινάκων που έχουμε πάρει είναι πάλι οι ίδιες δηλαδή 1000x1000,750x750 και 500χ500.



Εικόνα 23 το speedup για υπολογισμό, με σπασίματα πινάκων 1000χ1000,750χ750 και 500χ500, με δοσμένες εργασίες από 1 user ως προς τους workers

Η επόμενη γραφική προκύπτει από την γραφική 3 αλλά η διάφορα είναι πως δεν έχουμε τον ίδιο y άξονα αλλά σε αυτή τη γραφική ο y άξονας αντιπροσωπεύει τον χρόνο Tn.Και πάλι οι μετρήσεις είναι με τρεις διαφορετικές διαστάσεις των υποπινάκων και ειδικότερα για 1000x1000,750x750 και 500x500.



Εικόνα 24 οι χρόνοι που χρειάστηκαν για να ολοκληρωθεί 1 σταλμένη εργασία από user,για σπασίματα πινάκων 1000χ1000,750χ750 και 500χ500 , ως προς τους workers**.**

Οι αναλυτικοί χρόνοι που μετρήσαμε και εξαγάγαμε τις πιο πάνω γραφικές είναι :

Για την γραφική 23

	1000x1000	750x750	500x500
n	To/Tn	To/Tn	To/Tn
1	0,94	0,84991	0,759903
2	1,776938	1,59322	1,323944
3	2,473684	2,270531	1,910569
4	2,984127	2,865854	2,276029
5	3,916667	3,455882	2,831325
6	4,413146	3,868313	3,197279
7	4,771574	4,311927	3,533835
8	5,662651	5,053763	3,821138
9	6,266667	5,56213	4,454976
10	6,666667	6,025641	4,973545

Για την γραφική 24

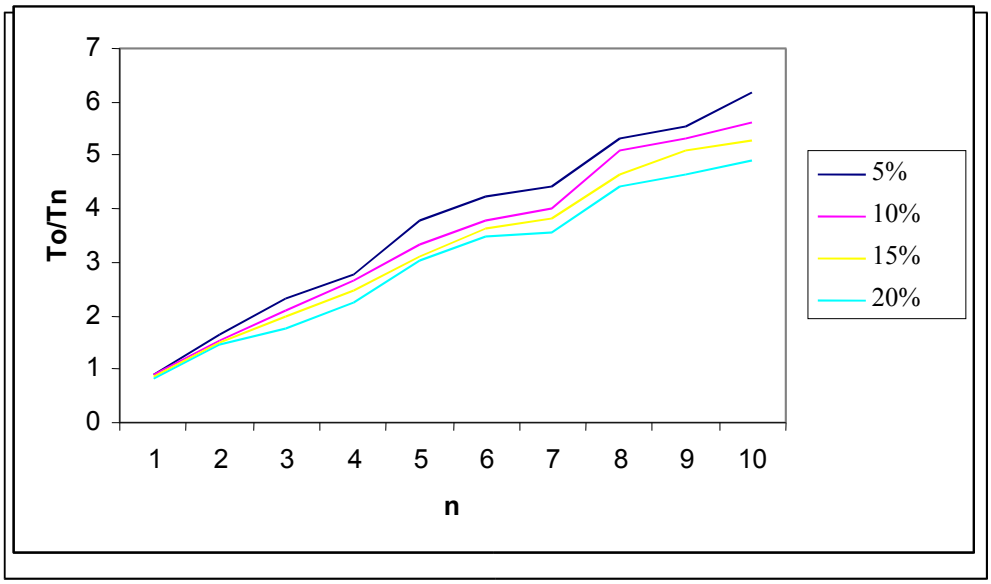
	1000x1000	750x750	500x500
n	Tn	Tn	Tn
1	1000	1106	1237
2	529	590	710
3	380	414	492
4	315	328	413
5	240	272	332
6	213	243	294
7	197	218	266
8	166	186	246
9	150	169	211
10	141	156	189

Στις επόμενες γραφικές έχουμε έναν user ο οποίος δίνει έναν υπολογισμό γινόμενου δυο πινάκων 3000x3000 και έχουμε κάνει δυο σπασίματα υποπινάκων , το πρώτο με διαστάσεις 1000x1000 και το δεύτερο με διαστάσεις 500x500.

Κάθε φορά που κάνουμε τους υπολογισμούς υποθέτουμε πως ο κάθε worker που αναλαμβάνει να εκτελέσει κάποια εργασία που του έχει δοθεί έχει μια πιθανότητα p (που ξεκινάει από 5% και αυξάνεται ανά 5% έως το 20%) να αποτύχει να επιστρέψει το αποτέλεσμα του υπολογισμού στο server. Αυτός ο worker θα σταματήσει να εκτελεί τον υπολογισμό και μετά από ένα ν χρονικό διάστημα (15 sec όταν έχουμε σπάσιμο σε υποπίνακα 1000x1000 και 3 sec όταν έχουμε σπάσιμο σε υποπίνακα 500x500) θα επανέλθει ζητώντας από τον server μια καινούργια εργασία για υπολογισμό.

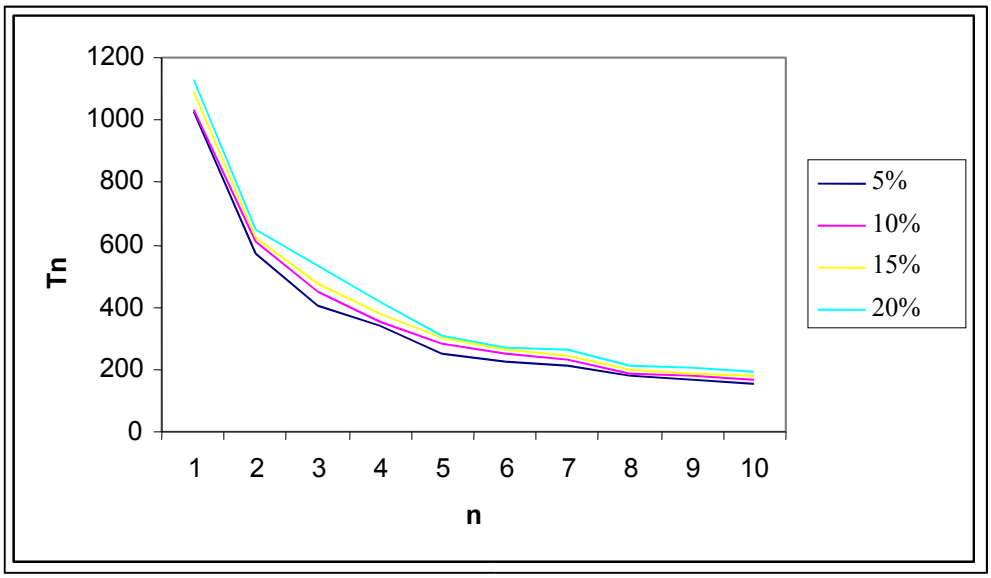
Ουσιαστικά με αυτό το τρόπο εξομοιώνουμε την περίπτωση όπου δεν έχουμε αξιόπιστους workers και δεν εκτελούν τον υπολογισμό είτε αποτυγχάνουν να τον εκτελέσουν και με την συγκεκριμένη καθυστέρηση επιτυγχάνουμε να έχουμε ένα σταθερό αριθμό workers που εκτελούν εργασίες υποθέτοντας πως έχουμε μια συνεχή ροή καινούργιων workers κάθε ν χρονικό διάστημα από τη στιγμή που αποτυγχάνει ένας worker.

Στην παρακάτω γραφική παράσταση έχουμε το γινόμενο δυο 3000x3000 πινάκων που έχουν σπάσει σε υποπίνακες διαστάσεων 1000x1000 .Για κάθε worker έχουμε μια πιθανότητα p (5%->20%) που αποτυγχάνει να εκτελέσει τον υπολογισμό .Στον x άξονα έχουμε το n (αριθμό από workers) και στον y άξονα έχουμε το πηλίκο To με το Tn όπου To ο χρόνος που χρειάζεται για να εκτελεστεί το γινόμενο δυο 3000x3000 σε έναν υπολογιστή και Tn ο χρόνος που χρειάζεται σε n υπολογιστές περιλαμβάνοντας και την πιθανότητα p να αποτύχει κάποιος υπολογιστής να εκτελέσει τον υπολογισμό.



Εικόνα 25 το speedup για υπολογισμό, με σπάσιμο πινάκων 1000χ1000, με δοσμένες εργασίες από 1 user ως προς τους workers και πιθανότητα μη επιστροφής αποτελεσμάτων από τους workers 5% έως 20%.

Στην επόμενη γραφική έχουμε ακριβώς τις ίδιες μετρήσεις αλλά αλλάζει ο y άξονας ο οποίος είναι ο χρόνος Tn που απαιτείται για να ολοκληρωθεί το γινόμενο δυο πινάκων 3000x3000 με σπάσιμο σε υποπίνακες διαστάσεων 1000x1000.Ο x άξονας παραμένει ο ίδιος και αντιπροσωπεύει τον αριθμό των υπολογιστών που έχουμε στη διάθεση μας για να γίνει ο υπολογισμός.



Εικόνα 26 ο χρόνος για υπολογισμό, με σπάσιμο πινάκων 1000χ1000, με δοσμένες εργασίες από 1 user ως προς τους workers και πιθανότητα μη επιστροφής αποτελεσμάτων από τους workers 5% έως 20%.

Οι αναλυτικοί χρόνοι που μετρήσαμε και εξαγάγαμε τις πιο πάνω γραφικές είναι :

Για την γραφική 25

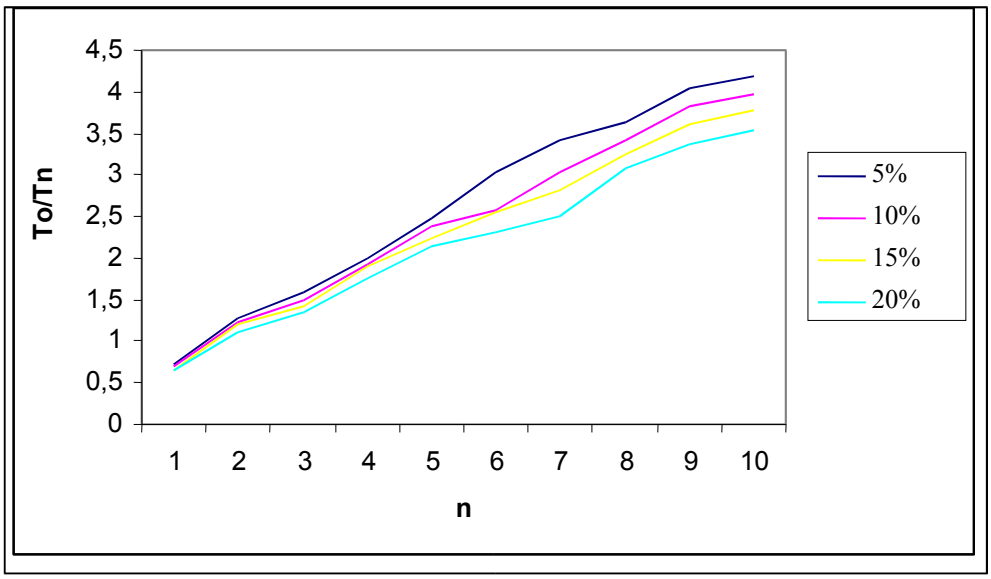
n	5% To/Tn	10% To/Tn	15% To/Tn	20% To/Tn
1	0,913508	0,908213	0,860018	0,833333
2	1,652021	1,538462	1,508828	1,450617
3	2,338308	2,088889	1,97479	1,763602
4	2,764706	2,670455	2,473684	2,248804
5	3,7751	3,345196	3,122924	3,032258
6	4,234234	3,7751	3,615385	3,481481
7	4,413146	4,017094	3,805668	3,54717
8	5,310734	5,108696	4,653465	4,413146
9	5,529412	5,310734	5,081081	4,630542
10	6,184211	5,628743	5,280899	4,895833

Για την γραφική 26

n	5% Tn	10% Tn	15% Tn	20% Tn
1	1029	1035	1093	1128
2	569	611	623	648
3	402	450	476	533
4	340	352	380	418
5	249	281	301	310
6	222	249	260	270
7	213	234	247	265
8	177	184	202	213
9	170	177	185	203
10	152	167	178	192

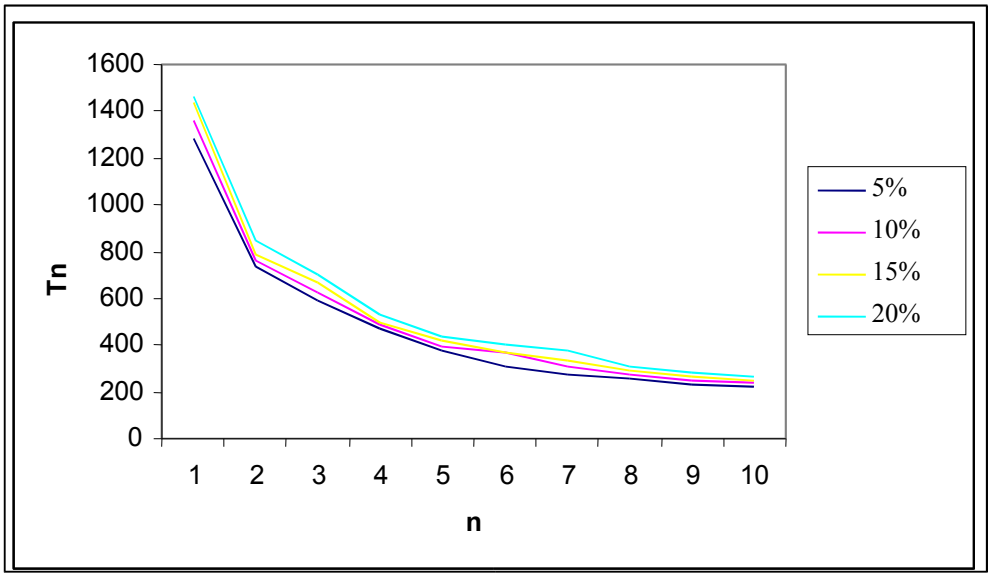
Στις πιο κάτω γραφικές παραστάσεις έχουμε καταγράψει τις μετρήσεις για το γινόμενο δυο 3000x3000 πινάκων αλλά αυτή τη φορά το σπάσιμο σε υποπινακες έχει γίνει σε υποπινακες με διαστάσεις 500x500.Και πάλι έχουμε μια πιθανότητα p σε κάθε worker που αναλαμβάνει να εκτελέσει μια εργασία και η πιθανότητα αυτή είναι από 5% έως 20% με βαθμιαία αύξηση 5%.

Στην παρακάτω γραφική ο x άξονας είναι το n δηλαδή ο αριθμός των workers και ο y To/Tn όπου To ο χρόνος που χρειάζεται για να εκτελεστεί το γινόμενο δυο 3000x3000 σε έναν υπολογιστή και Tn ο χρόνος που χρειάζεται σε n υπολογιστές (με σπάσιμο 500x500) περιλαμβάνοντας και την πιθανότητα p να αποτύχει κάποιος υπολογιστής να εκτελέσει τον υπολογισμό.



Εικόνα 27 το speedup για υπολογισμό, με σπάσιμο πινάκων 500χ500, με δοσμένες εργασίες από 1 user ως προς τους workers και πιθανότητα μη επιστροφής αποτελεσμάτων από τους workers 5% έως 20%.

Στην επόμενη γραφική έχουμε ακριβώς τις ίδιες μετρήσεις αλλά αλλάζει ο y άξονας ο οποίος είναι ο χρόνος Tn που απαιτείται για να ολοκληρωθεί το γινόμενο δυο πινάκων 3000x3000 με σπάσιμο σε υποπινακες διαστάσεων 500x500.Ο x άξονας παραμένει ο ίδιος και αντιπροσωπεύει τον αριθμό των υπολογιστών που έχουμε στη διάθεση μας για να γίνει ο υπολογισμός.



Εικόνα 28 ο χρόνος για υπολογισμό, με σπάσιμο πινάκων 500χ500, με δοσμένες εργασίες από 1 user ως προς τους workers και πιθανότητα μη επιστροφής αποτελεσμάτων από τους workers 5% έως 20%.

Οι αναλυτικοί χρόνοι που μετρήσαμε και εξαγάγαμε τις πιο πάνω γραφικές είναι :

Για την γραφική 27

n	5% To/Tn	10% To/Tn	15% To/Tn	20% To/Tn
1	0,732658	0,692704	0,654596	0,643836
2	1,27027	1,235217	1,192893	1,115065
3	1,59322	1,501597	1,415663	1,346705
4	2	1,92623	1,902834	1,766917
5	2,473684	2,379747	2,238095	2,136364
6	3,022508	2,575342	2,540541	2,315271
7	3,418182	3,022508	2,80597	2,513369
8	3,629344	3,418182	3,241379	3,081967
9	4,034335	3,836735	3,601533	3,357143
10	4,177778	3,966245	3,7751	3,54717

Για την γραφική 28

n	5% Tn	10% Tn	15% Tn	20% Tn
1	1283	1357	1436	1460
2	740	761	788	843
3	590	626	664	698
4	470	488	494	532
5	380	395	420	440
6	311	365	370	406
7	275	311	335	374
8	259	275	290	305
9	233	245	261	280
10	225	237	249	265

Οι υπολογιστές που χρησιμοποιήθηκαν για να καταγραφούν οι μετρήσεις είναι :

2 Pentium 3 ,733Mhz με 128mb SDRAM
1 Pentium 3 ,750Mhz με 128mb SDRAM
1 Pentium 3 ,600Mhz με 128mb SDRAM
1 Pentium 3 ,665Mhz με 128mb SDRAM
5 Celeron ,635Mhz με 128mb SDRAM

4.1.1Συμπέρασμα

Από τα αποτελέσματα των γραφικών παραστάσεων εύκολα αντιλαμβανόμαστε την δυνατότητα του κατανεμημένου υπολογισμού καθώς ακόμα και όταν είχαμε αρκετά μεγάλη πιθανότητα p (p=20%) και χρησιμοποιώντας για το πείραμα μια εφαρμογή η οποία κατά την διάσπαση της οι επί μέρους εργασίες οι οποίες δημιουργούνται δεν είναι ανεξάρτητες αλλά εξαρτιόνται ορισμένες μεταξύ τους καταφέρνουμε να πάρουμε ένα ικανοποιητικό speedup το οποίο φθάνει το πέντε με αριθμό από workers ίσο με δέκα .Ενώ στην περίπτωση όπου το p=0% το speedup λόγω των εξαρτήσεων αλλά και της αποστολής των δεδομένων καταλήγει στο 7 κάτι το οποίο είναι πολύ ικανοποιητικό.

Βιβλιογραφία – Αναφορές

1. αρχική σελίδα του distributed.net http://www.distributed.net/
2. αρχική σελίδα του seti@home http://setiathome.ssl.berkeley.edu/
3. πρόσθετες πληροφορίες για το seti@home http://setiweb.ssl.berkeley.edu/
4. σελίδα με πληροφορίες για το seti@home http://www.planetary.org/html/UPDATES/seti/
5. σελίδα με αποτελέσματα και στατιστικά τον υπολογισμών του folding@home http://folding.stanford.edu/results.html
6. σελίδα με αναφορές σε υπάρχοντες πλατφόρμες και projects αλλά και γενικές πληροφορίες γύρω από το distributed computing http://distributedcomputing.info/
7. αρχική σελίδα του parabon http://www.parabon.com/
8. αρχική σελίδα της πλατφόρμας boinc http://boinc.ssl.berkeley.edu/intro.php
9. αρχική σελίδα της πλατφόρμας cosm http://www.mithral.com/projects/cosm/
10. αρχική σελίδα της corba http://www.corba.com/
11. αρχική σελίδα της python http://www.python.org/
12. **Public Computing: Reconnecting People to Science**_Dr. David P. Andersony March 21, 2004
13. **BOINC: A System for Public-Resource Computing and Storage** David P. Anderson Space Sciences Laboratory University of California at Berkeley davea@ssl.berkeley.edu
14. **Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment** Jerome Verbeke, Neelakanth Nadgir, Greg Ruetsch, Ilya Sharapov Sun Microsystems, Inc., Palo Alto
15. **A new major SETI project based on Project Serendip data and 100,000 personal computers** W. T. Sullivan, III (U. Washington), D. Werthimer, S. Bowyer, J. Cobb (U.California, Berkeley), D. Gedye, D. Anderson (Big Science, Inc.) (1997)
16. **SETI@home: An Experiment in Public-Resource Computing**, David P. Anderson,Jeff Cobb,Eric Korpela,Matt Lebofsky,Dan Werthimer,Space Sciences Laboratory U.C. Berkeley
17. **SETI@home:Massively Distributed Computing for SETI** By Eric Korpela, Dan Werthimer, David Anderson, Jeff Cobb, and Matt Lebofsky
18. This is a simplified summary of the article by Michael R. Shirts and Vijay S. Pande in PHYSICAL REVIEW LETTERS - 28 MAY 2001 titled **MATHEMATICAL ANALYSIS OF COUPLED PARALLEL SIMULATIONS**) βρισκεται στον χωρο http://folding.stanford.edu/education/papers/index.html
19. **The SETI@home Sky Survey** βρισκεται στον χωρο http://setiathome.ssl.berkeley.edu/sciencepaper.html
20. **The Frontier® Application Programming Interface, Version 1.5.2**
21. **A Summary of Grid Computing Environments** Geoffrey , Dennis Gannon2,Mary Thomas4 1Community Grid Computing Laboratory, Indiana University 501 N Morton Suite 224,

Bloomington IN 47404 2Computer Science Department, Indiana University 3School of Informatics and Physics Department, Indiana University 4Texas Advanced Computing Center, The University of Texas at Austin, 10100 Burnet Road, Austin, Texas 78758 edu,,mthomas@tacc.utexas.edu gannon@cs.indiana.gcf@indiana.

22. **Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology** Stefan M. Larson1,2, Christopher D. Snow1,2, Michael Shirts1, and Vijay S. Pande1,2,*
23. **Peer-to-Peer Computing** Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose,Kiran Nagaraja1, Jim Pruyne, Bruno Richard,Sami Rollins 2 ,Zhichen Xu HP Laboratories Palo Alto HPL-2002-57 March 8th , 2002*
24. **MultiTiered Distributed Computing Platform** Andrew Page, Thomas Keane,Richard Allen, Thomas J. Naughton Department of Computer ScienceNational University of Ireland, Maynooth County Kildare, Ireland John Waldron Department of Computer Science Trinity College Dublin Dublin 2, Ireland
25. **Frontier®: The Premier Internet Computing Platform** Copyright © 1999-2004, Parabon Computation
26. **Implementation and Characterization of Protein Folding on a Desktop Computational Grid** B. Uk1, M. Taufer1, T. Stricker1, G. Settanni2, A. Cavalli2 1 Department of Computer Science 2 Department of Biochemistry ETH Zurich University of Zurich CH-8092 Zurich, Switzerland CH-8057 Zurich, Switzerland buk,taufer,tomstr@inf.ethz.ch settanni,cavalli@bioc.unizh.ch