ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΥΠΗΡΕΣΙΕΣ ΔΙΚΤΥΟΥ ΓΙΑ ΕΠΙΣΤΗΜΟΝΙΚΟΥΣ ΥΠΟΛΟΓΙΣΜΟΥΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΝΑΚΟΥ ΒΑΣΙΛΕΙΟΥ

Βόλος, Ιανουάριος 2012

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΥΠΗΡΕΣΙΕΣ ΔΙΚΤΥΟΥ ΓΙΑ ΕΠΙΣΤΗΜΟΝΙΚΟΥΣ ΥΠΟΛΟΓΙΣΜΟΥΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΝΑΚΟΥ ΒΑΣΙΛΕΙΟΥ

Επιβλέπων: Μανώλης Βάβαλης

Αναπληρωτής Καθηγητής Τ.Μ.Η.Υ.Τ.Δ

Εγκρίθηκε από τη διμελή εξεταστική επιτροπή τον Ιανουάριο 2012.

(Υπογραφή)	(Υπογραφή)
……	…….
	Μανώλης Βάβαλης
	Αναπληρωτής Καθηγητής Τ.Μ.Η.Υ.Τ.Δ

Βόλος, Ιανουάριος 2012

(Υπογραφή)

……

ΝΑΚΟΣ ΒΑΣΙΛΕΙΟΣ

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων
Πανεπιστήμιο Θεσσαλίας

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 2

Κεφάλαιο 1ο

Εισαγωγή

Με τον όρο επιστημονικός υπολογισμός αναφερόμαστε στην μελέτη και κατασκευή μαθηματικών μοντέλων και τεχνικών για την επίλυση επιστημονικών προβλημάτων με την βοήθεια της επιστήμης των υπολογιστών. Συνήθως έχει να κάνει με προβλήματα μεγάλης κλίμακας και απαιτεί μεγάλες ποσότητες από υπολογιστικούς πόρους. Ο χώρος του cloud computing θα μπορούσε πολύ εύκολα να συνδεθεί με αυτόν των επιστημονικών υπολογισμών, καθώς προσφέρει εικονικά άπειρους υπολογιστικούς πόρους. Μία πολύ καλή περίπτωση ενός τέτοιου cloud αποτελεί το Amazon Ec2 που προσφέρει την δυνατότητα στους χρήστες , να χρησιμοποιήσουν εικονικούς ηλεκτρονικούς υπολογιστές μέσω διαδικτύου.

Ένας τομέας του πεδίου των επιστημονικών υπολογισμών ο οποίος είναι αρκετά ενδιαφέρον είναι αυτός της επίλυσης συστημάτων γραμμικής άλγεβρας. Προς αυτήν την κατεύθυνση το repository της βιβλιοθήκης NETLIB παρέχει ελεύθερα τη βιβλιοθήκη της Lapack. Η Lapack είναι βιβλιοθήκη συναρτήσεων γραμμικής άλγεβρας και χαίρει της εκτίμησης των επιστημόνων που ασχολούνται με τον συγκεκριμένο τομέα. Απόδειξη αποτελούν οι 98 εκατομμύρια προσπελάσεων της Lapack[1] .

Όσον αφορά τους τελικούς χρήστες, θα προτιμούσαν να εξοικονομήσουν χρόνο, κόπο και υπολογιστική μνήμη που απαιτούνται για την εγκατάσταση του απαιτούμενου λογισμικού στον υπολογιστή τους. Οπότε φαντάζει ιδανική η επιλογή της παροχής λογισμικού υπό μορφής υπηρεσίας (Software as a Service). Η συγκεκριμένη τάση παροχής λογισμικού είναι από τις πλέον αναπτυσσόμενες στο διαδίκτυο καθώς παρακάμπτει την εγκατάσταση λογισμικού στον υπολογιστή του εκάστοτε χρήστη. Αυτή η προσφορά λογισμικού μέσω διαδικτύου αποτελεί τις λεγόμενες υπηρεσίες διαδικτύου (web services) που τελευταία χρόνια έχουν σημειώσει σημαντική ανάπτυξη. Είναι τμήματα λογισμικού τα οποία είναι ικανά να εκτελέσουν από απλές μέχρι πολύπλοκες δραστηριότητες ανεξάρτητα από την πλατφόρμα χρήσης.

[1] http://www.netlib.org/master_counts2.html

Η χρήση web services σε συνδυασμό με το cloud computing δημιουργεί κάποια ζητήματα σχετικά με το χώρο αποθήκευσης των δεδομένων αλλά και το σύστημα αρχείων που θα χρησιμοποιήσουμε. Μια λύση φαίνεται να αποτελεί το Apache Hadoop το οποίο υλοποιεί λογισμικό ανοιχτού κώδικα για αξιόπιστους , κλιμακούμενους κατανεμημένους υπολογισμούς. Ανάμεσα στα άλλα υποπρογράμματα, προσφέρει το HDFS (Hadoop Distributed File System) ένα κατανεμημένο σύστημα αρχείων δηλαδή με μεγάλο ταχύτητα πρόσβασης στα δεδομένα. Ακόμη , προσφέρει το MapReduce το οποίο αποτελεί ένα πλαίσιο λογισμικού για κατανεμημένη επεξεργασία μεγάλης σειράς δεδομένων από clusters υπολογιστών.

1.1 Αντικείμενο διπλωματικής

Η εγκατάσταση της lapack στον προσωπικό ηλεκτρονικό υπολογιστή , μπορεί να αποβεί μια χρονοβόρα διαδικασία και σίγουρα απαιτεί κάποιες γνώσεις τις οποίες ενδεχομένως να μην έχει ο επίδοξος χρήστης. Οπότε το να προσφέρουμε την lapack ως web service και μάλιστα μέσα από το Amazon ec2, αποτέλεσε μια πρόκληση. Έτσι λοιπόν ο εκάστοτε χρήστης θα μπορεί να μπαίνει χρησιμοποιεί μια registry όπου θα βλέπει ποιές συναρτήσεις είναι διαθέσιμες και να παίρνει το αντίστοιχο WSDL το οποίο τοποθετώντας το σε ένα πρόγραμμα πελάτη να δίνει τα δεδομένα του και να παίρνει τα αποτελέσματα της συνάρτησης. Το πώς ακριβώς θα διαχειρίζονται τα δεδομένα είναι άλλο ένα κομμάτι τις διπλωματικής και συγκεκριμένα κατά πόσο μας συμφέρει να έχουμε ένα HDFS cluster όπου θα αποθηκεύονται τα δεδομένα που θέλει να εισάγει ο χρήστης αλλά και οι λύσεις ώστε να αποφεύγουμε την κίνηση δεδομένων πάνω από το δίκτυο.

Στα επόμενα κεφάλαια πρόκειται να περιγράψουμε την συμβατική Lapack , αλλά και την lapack ως web service, να κάνουμε μία ανάλυση για το πώς υλοποιούμε μία lapack routine ως web service και να την «σηκώσουμε» σε έναν Glassfish Application Server που έχουμε εγκαταστήσει σε κάποιο Amazon Ec2 instance. Ακόμη θα παρουσιάσουμε τα αποτελέσματα σχετικά με την χρήση του HDFS για την αποθήκευση των δεδομένων μας και την προοπτική του MapReduce ως ένα πλαίσιο για κατανεμημένη επεξεργασία των ρουτινών μας.

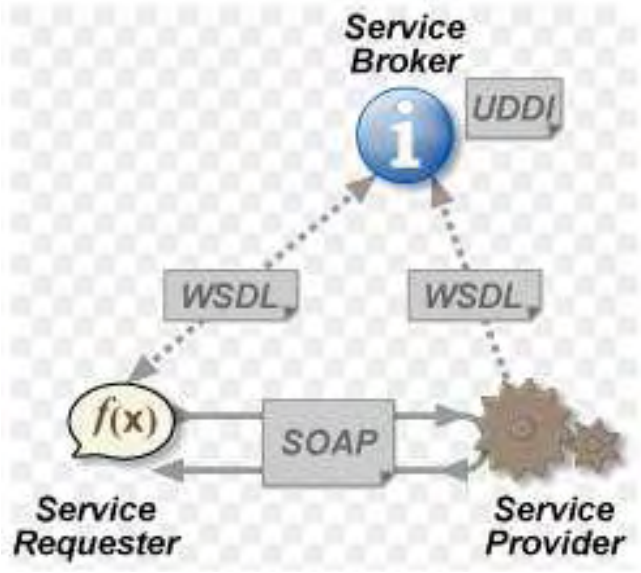
Κεφάλαιο 2ο

Τεχνολογικό Υπόβαθρο

2.1 WEB SERVICES

Ως web services (1) περιγράφεται ένα αυτοπεριγραφόμενο , αυτοδύναμο τμήμα λογισμικού, διαθέσιμο μέσω δικτύου όπως το Internet , το οποίο ολοκληρώνει εργασίες, λύνει προβλήματα ή διευθύνει συναλλαγές εκ μέρους ενός χρήστη ή μιας εφαρμογής. Οι web services αποτελούν μία κατανεμημένη υπολογιστική υποδομή από διαφορετικά τμήματα αλληλεπιδρώντων εφαρμογών τα οποία προσπαθούν να επικοινωνήσουν μεταξύ τους, πάνω από ιδιωτικό ή δημόσιο δίκτυο για να αποτελέσουν εν τέλει ένα λογικό σύστημα.

Αν θεωρήσουμε ότι κάποιος χρήστης (requestor) επιθυμεί να χρησιμοποιήσει κάποια web service, τότε θα πρέπει πρώτα να επικοινωνήσει με έναν κεντρικό κατάλογο των υπηρεσιών (registry) στον οποίο οι δημιουργοί δημοσιεύουν τις υπηρεσίες τους. Ο κατάλογος θα οδηγήσει τον χρήστη στον παροχέα της υπηρεσίας (provider) στον οποίο είναι υλοποιημένη και διαθέσιμη η υπηρεσία.



Εικόνα 2-1 Αρχιτεκτονική των web services

Όπως φαίνεται και από την Εικόνα 2-1, για να λειτουργήσει η όλη λογική των web services χρησιμοποιούνται κάποιες τεχνολογίες. Πέρα όμως από αυτές που φαίνονται στην Εικόνα, χρησιμοποιούνται και άλλες και αξίζει να αναφερθούμε σε κάποιες από αυτές.

Συγκεκριμένα στο επίπεδο μεταφοράς χρησιμοποιούνται κάποια πρωτόκολλα μεταφοράς όπως το HTTP (Hyper Text Transfer Protocol) πολύ γνωστό προτόκολλο επιπέδου εφαρμογής του web αλλά και το JMS (Java Message Service) που είναι μια διεπαφή (API) για ανταλλαγή μηνυμάτων μεταξύ δύο ή περισσοτέρων clients.

Στο επίπεδο ανταλλαγής μηνυμάτων χρησιμοποιούνται το προτόκολλο επικοινωνίας SOAP (Simple Object Access Protocol) και η XML (Extensible Markup Language). Όσον αφορά το SOAP είναι ένα πρωτόκολλο επικοινωνίας για την ανταλλαγή δομημένων πληροφοριών στο πλαίσιο των web services. Βασίζεται στην XML για την μορφή των μηνυμάτων που ανταλλάσσονται όπως και σε άλλα πρωτόκολλα ως μέσα μεταφοράς (HTTP, SMTP) για την επικοινωνία των μηνυμάτων και την μετάδοσή τους. Βασικοί στόχοι του SOAP αποτελούν η απλότητα και επεκτασιμότητα. Χρησιμοποιείται ανάμεσα στον client και τον παροχέα για να οριστεί ποιά πληροφορία θα σταλεί και πώς. Για την XML αρκεί να αναφερθεί ότι αποτελεί μία γλώσσα την οποία χρησιμοποιούμε για να την κωδικοποίηση κειμένων και στις web services τα μηνύματα που ανταλλάσσονται είναι κωδικοποιημέα με αυτήν.[2]

Βασική τεχνολογία στην οποία βασίζονται οι web services ειναι η WSDL (Web Services Description Language) η οποία είναι μια γλώσσα βασισμένη στην XML και χρησιμοποιείται για την περιγραφή της λειτουργικότητας που προσφέρεται από την εκάστοτε υπηρεσία. Η περιγραφή , παρέχεται σε μορφή αναγνωρίσιμη από τον υπολογιστή και περιλαμβάνει το πώς καλείται η υπηρεσία , τί παραμέτρους περιμένει να δεχθεί και τι δομές δεδομένων επιστρέφει.

Στον κεντρικό κατάλογο των υπηρεσιών χρησιμοποιείται το UDDI που είναι ένα μητρώο βασισμένο στην XML και ανεξάρτητο πλατφόρμας και είναι παγκοσμίως διαθέσιμο για την περιγραφή και ανακάλυψη επιχειρήσεων, οργανισμών και άλλων παροχέων web services, για τις ίδιες τις υπηρεσίες και τις διεπαφές μεσω των οποίων κάποιος χρήστης μπορεί να χρησιμοποιήσει μια υπηρεσία. Λειτουργεί όπως ένας χρυσός οδηγός με white pages , yellow pages και green pages. Οι white pages παρέχουν πληροφορίες όπως όνομα,διεύθυνση για μία επιχείρηση , οι yellow pages παρέχουν πληροφορίες για την

[2] http://www.w3.org/TR/wsdl

κατηγοριοποίηση των επιχειρήσεων ενώ οι green pages περιέχουν τεχνικές πληροφορίες για τις υπηρεσίες που παρέχει η εκάστοτε επιχείρηση.

Θα ήταν παράλειψη να μην αναφερθούμε στην BPEL (Business Process Execution Language) μία γλώσσα προγραμματισμού που δημιουργήθηκε από τον οργανισμό OASIS και χρησιμοποιείται για την ενορχήστρωση των web services. Ορίζει τις ροές εργασίας στο πλαίσιο των υπηρεσιών.

2.2 Lapack

Η Lapack[3] είναι μία βιβλιοθήκη λογισμικού η οποία υποστηρίζει την αριθμητική γραμμική άλγεβρα. Παρέχει ρουτίνες για την επίλυση συστημάτων γραμμικών εξισώσεων και γραμμικών ελαχίστων τετραγώνων, προβλήματα τιμής eigen και προβλήματα μοναδικής λύσης. Παρέχει επίσης τις σχετικές παραγοντοποιήσεις πινάκων (LU, Cholesky, QR, SVD, γενικευμένο Schur), όπως επίσης και σχετικοι υπολογισμοί. Μπορεί να χειριστεί πυκνούς και επεκτάσιμους πίνακες αλλά όχι γενικά αραιούς πίνακες. Σε κάθε περίπτωση παρέχει ίδια λειτουργικότητα για πραγματικούς και σύνθετους πίνακες, για απλή και διπλή ακρίβεια. Αξίζει να αναφερθεί ότι η Lapack είναι γραμμένη σε Fortran 90. Θα την αναλύσουμε περισσότερο στο επόμενο κεφάλαιο.

2.3 Cloud Computing

Με τον όρο cloud computing αναφερόμαστε στην παροχή υπολογιστικών συστημάτων ως υπηρεσία αντί για προϊόν, όπου κοινοί πόροι, λογισμικό και πληροφορίες παρέχονται στους υπολογιστές και άλλες συσκευές σαν παροχή (όπως το ηλεκτρικό ρεύμα) πάνω από δίκτυο. Όπως με το ηλεκτρικό ρεύμα, έτσι και στο cloud computing, ο τελικός χρήστης δεν γνωρίζει και ούτε χρειάζεται να κατανοήσει τι συσκευές χρησιμοποιούνται ή την δομή που απαιτείται για να παρέχεται η υπηρεσία.

Για να γίνουμε πιο συγκεκριμένοι, οι παροχείς των συγκεκριμένων τεχνολογιών, προσφέρουν εφαρμογές μέσω internet οι οποίες προσπελαύνονται απο web browsers απο σταθερούς υπολογιστές και εφαρμογές κινητών τηλεφώνων, ενώ λογισμικό και δεδομένα εταιρειών αποθηκεύεται σε servers σε απομακρυσμένες περιοχές.

[3] http://www.netlib.org/lapack/

Στην προσπάθειά μας, χρησιμοποιήσαμε το Amazon Web Services (AWS), που αποτελεί μία συλλογή από απομακρυσμένες υπολογιστικές υπηρεσίες (web servies) οι οποίες όλες μαζί αποτελούν μία πλατφόρμα cloud computing. Ξεκίνησε τον Ιούλιο του 2002 και από τότε παρέχει online υπηρεσίες για άλλες ιστοσελίδες ή εφαρμογές στην πλευρά πελάτη. Οι περισσότερες υπηρεσίες δεν είναι ευθέως διαθέσιμες για τους τελικούς χρήστες, αλλά προσφέρουν λειτουργικότητα την οποία μπορούν να εκμεταλευτούν developers. Οι υπηρεσίες του AWS είναι προσβάσιμες μέσω HTTP χρησιμοποιώντας REST (Representational State Transfer) και SOAP πρωτόκολλα. Ο χρήστης χρεώνεται για ότι χρησιμοποιεί με την χρέωση να διαφέρει ανάλογα με την υπηρεσία. Οι υπηρεσίες που προσφέρει είναι πολλές και ενδεικτικά θα αναφέρουμε μερικές απο κάθε τομέα.[4]

COMPUTE

AMAZON ELASTIC CLOUD (EC2) : Παρέχει κλιμκωτούς ιδιωτικούς εικονικούς servers. Αυτήν την υπηρεσία θα χρησιμοποιήσουμε και εμείς για να παρέχουμε τις lapack web services.

CONTENT DELIVERY

AMAZON CLOUDFRONT : Ένα δίκτυο διανομής περιεχομένου για κατανεμημένα αντικείμενα αποθηκευμένα στο S3 στις λεγόμενες «ακραίες περιοχές» κοντά στον αιτούμενο της υπηρεσίας.

DATABASE

AMAZON SIMPLEDB : Επιτρέπει στους χρήστες να εκτελούν ερωτήματα σε δομές δεδομένων. Λειτουργεί σε συνεργασία με το EC2 και το S3 για να παρέχει την βασική λειτουργικότητα μιας δομής δεδομένων.

DEPLOYMENT AND MANAGEMENT

AWS Elastic Beanstalk : Παρέχει γρήγορη ανάπτυξη και διαχείρηση εφαρμογών στο cloud.

[4] http://en.wikipedia.org/wiki/Amazon_S3

E-Commerce

Amazon Fulfillment Web Service (FWS) : Παρέχει ένα προγραμματιστικό web service για πωλητές ώστε να φορτώνουν αντικειμένα από και προς το Amazon χρησιμοποιώντας το Fulfillment της Amazon.

Messaging

Amazon Simple Queue Service (SQS) : Παρέχει ουράμηνυμάτων για web applications.

Monitoring

Amazon CloudWatch : παρέχει παρακολούθηση για τους πόρους και τις εφαρμογές του AWS cloud, ξεκινώντας με το EC2.

Networking

Amazon Route 53 : Παρέχει μία επεκτάσιμη Domain Name System (DNS) υπηρεσία, υψηλής διαθεσιμότητας.

Payment & Billing

Amazon Flexible Payments Service (FPS): Παρέχει μια διεπαφή για οικονομικές συναλλαγές κυρίως μικρών ποσών.

Storage

Amazon Simple Storage Service (S3) : Παρέχει αποθήκευση βασισμένη σε web service.

Support

AWS Premium Support : είναι ένα κανάλι γρήγορης και προσωπικής απόκρισης το οποίο λειτουργεί αδιάκοπα από έμπειρους μηχανικούς τεχνικής υποστήριξης.

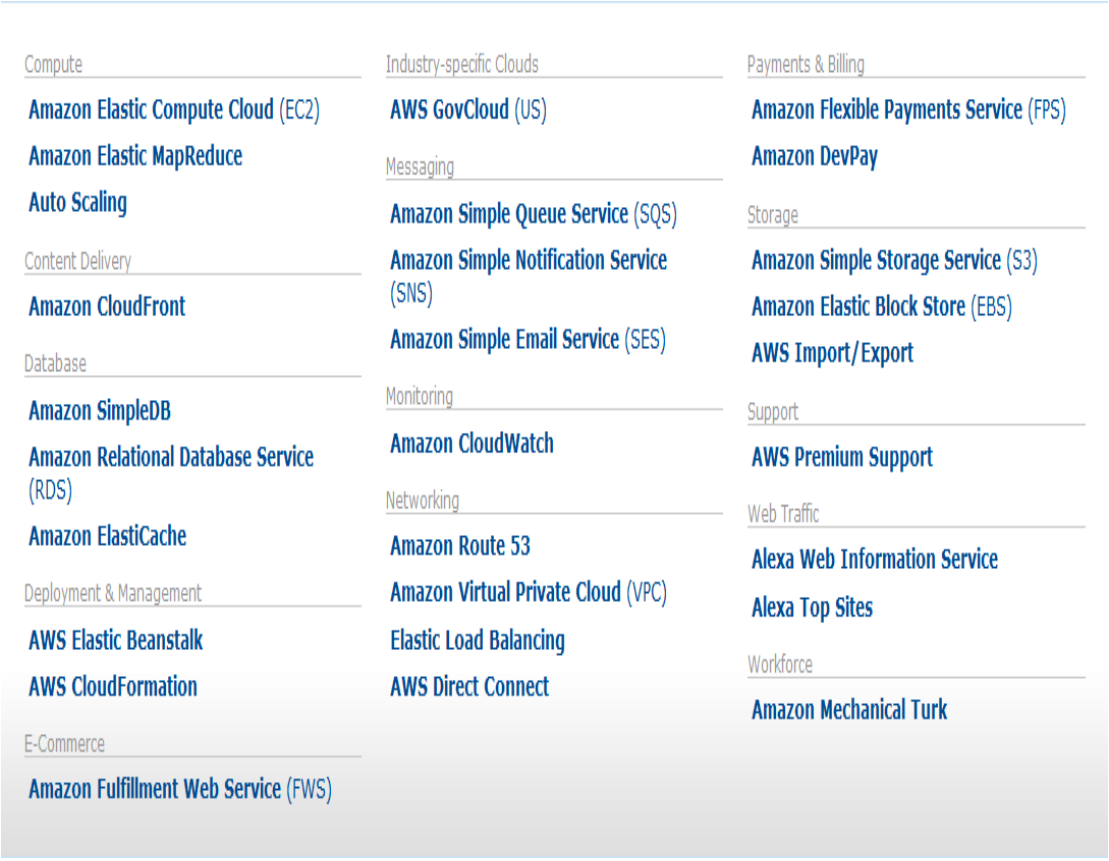
Web Traffic

Alexa Web Information Service : Παρέχει την τεράστια αποθήκη από πληροφορίες σχετικές με την κίνηση και την δομή του διαδικτύου της Alexa, σε προγραμματιστές.

Workforce

Amazon Mechanical Turk : Είναι μία αγορά εργασίας που απαιτεί ανθρώπινη λογική. Επιτρέπει σε εταιρείες να έχουν πρόσβαση στην αγορά προγραμματιστικά και το αντιστροφο. Διαχειρίζεται μικρά κομμάτια εργασίας, κατανεμημένα σε διάφορους ανρθώπους.

Γενικά όλες οι υπηρεσίες που προσφέρει το Amazon Web Service φαίνονται στην Εικόνα 2-2



Εικόνα 2-2 Υπηρεσίες του AWS

Για τις ανάγκες της εργασίας, χρησιμοποιούμε το EC2 όπου δεσμεύσαμε ένα Amazon Linux 32 bit εικονικό υπολογιστή. Θα τον διαχειροζόμαστε μέσω ενός τερματικού. Την ακριβή

διαδικασία την περιγράφουμε στο 5ο κεφάλαιο. Ακόμη θα χρησιμοποιήσουμε και το Amazon S3 για την αποθήκευση των επιθυμητών δεδομένων μας.

2.4 HADOOP

Το Hadoop είναι ένα πλαίσιο λογισμικού για την εκτέλεση εφαρμογών σε clusters υπολογιστών χωρίς την ανάγκη χρήσης εξειδικευμένου υλικού. Είναι γραμμένο σε γλώσσα Java και επιτρέπει σε κάποιον να δημιουργήσει και να εκτελέσει εφαρμογές που επεξεργάζονται μεγάλο όγκο δεδομένων. Το Hadoop έχει διάφορα πλεονεκτήματα όπως την κλιμάκωση που δίνει δυνατότητα αξιόπιστης αποθήκευσης και επεξεργασίας μέχρι και petabytes δεδομένων, την οικονομία, όπου κατανέμονται δεδομένα και έχουμε επεξεργασία σε clusters που αποτελούνται μέχρι και από χιλιάδες κοινούς υπολογιστές. Την αποδοτικότητα, που την αποκτούμε με την κατανομή των δεδομένων, και η επεξεργασία γίνεται παράλληλα σε όλους τους κόμβους, προσφέροντας γρήγορη εκτέλεση των εργασιών και την αξιοπιστία, η οποία επιτυγχάνεται μέσω της αυτόματης διατήρησης πολλαπλών αντιγράφων των δεδομένων, και αυτόματης ανάθεσης των εργασιών υπολογισμού σε νέους κόμβους σε περίπτωση βλάβης.

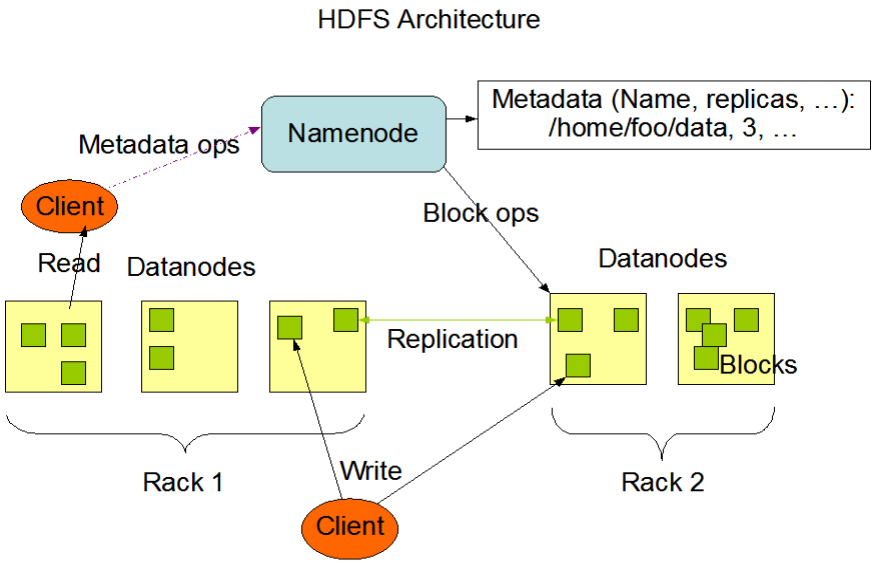
Τα κύρια χαρακτηριστικά του Hadoop είναι η υλοποίηση του Map- Reduce programming paradigm, μοιράζοντας την εφαρμογή σε μικρά blocks εργασίας, η χρήση Hadoop Distributed File System, για διατήρηση αντιγράφων δεδομένων σε διάφορους υπολογιστές και η χρήση του σε cluster 10,000 υπολογιστών. Όμως, υπάρχουν μεγάλες απαιτήσεις σε κύρια και δευτερεύουσα μνήμη. Η βέλτιστη απόδοση γίνεται μόνο μέσω καλής κατανομής των δεδομένων στους κόμβους και έχουμε ανάγκη γνώσης της τοπολογίας του δικτύου και χρήση αποκλειστικών switches για καλύτερο έλεγχο του bandwidth. [5]

Το HDFS είναι ένα κατανεμημένο σύστημα αρχείων, σχεδιασμένο να τρέχει σε κοινό υλικό. Υπάρχουν πολλές ομοιότητες με υπάρχοντα DFS, με σημαντικές διαφορές όπως: η υψηλή ανεχτικότητα σε βλάβες, η δυνατότητα εκτέλεσης του σε υλικό χαμηλού κόστους, η ψηλή ρυθμοαπόδοση πρόσβασης σε δεδομένα εφαρμογών και είναι ιδανικό για εφαρμογές με μεγάλο όγκο δεδομένων. Οι στόχοι του HDFS είναι όσον αφορά τη βλάβη υλικού, η ανίχνευση και ταχεία, αυτόματη ανάνηψη από σφάλματα. Για τη streaming πρόσβαση σε Δεδομένα, είναι σημαντική η ψηλή ρυθμοαπόδοση για batch processing, παρά η χαμηλή καθυστέρηση που έχει σημασία στη διαδραστική επικοινωνία και για τα μεγάλα σύνολα

[5] http://hadoop.apache.org/

δεδομένα, είναι ρυθμισμένο για υποστήριξη μεγάλου όγκου δεδομένων και τυχαίου αριθμού από κόμβους υπολογισμού. Όσον αφορά τη μετακίνηση της επεξεργασίας παρά των δεδομένων, υπάρχει μείωση συμφόρησης δικτύου από μεγάλα σύνολα δεδομένων και αύξηση απόδοσης και έχουμε μεταφερσιμότητα μεταξύ ετερογενούς υλικού και λογισμικού, που είναι σχεδιασμένο για λειτουργία σε διάφορες πλατφόρμες.

Το HDFS είναι δομημένο με Master- Slave Αρχιτεκτονική, όπου κάθε cluster αποτελείται από ένα NameNode και ένα σύνολο από DataNodes, συνήθως έχουμε ένα για κάθε κόμβο στο cluster. Έχουμε ένα Master Server, όπου διαχείριζεται το NameSpace του Συστήματος Αρχείων, όπου εκτελούνται οι λειτουργιες σε αρχεία και φακέλους και υπάρχει αντιστοιχιση των blocks δεδομένων σε DataNodes. Διαχειρίζεται τις ρυθμίσεις του cluster και το μηχανισμό δημιουργίας αντίγραφων των blocks. Οι Datanodes διαχειρίζονται την αποθήκευση στον κόμβο που εκτελούνται, την εξυπηρέτηση των αιτήσεων γραφής και ανάγνωσης από τους clients και την εκτέλεση λειτουργιών στα blocks, μετά από αιτήσεις του NameNode.



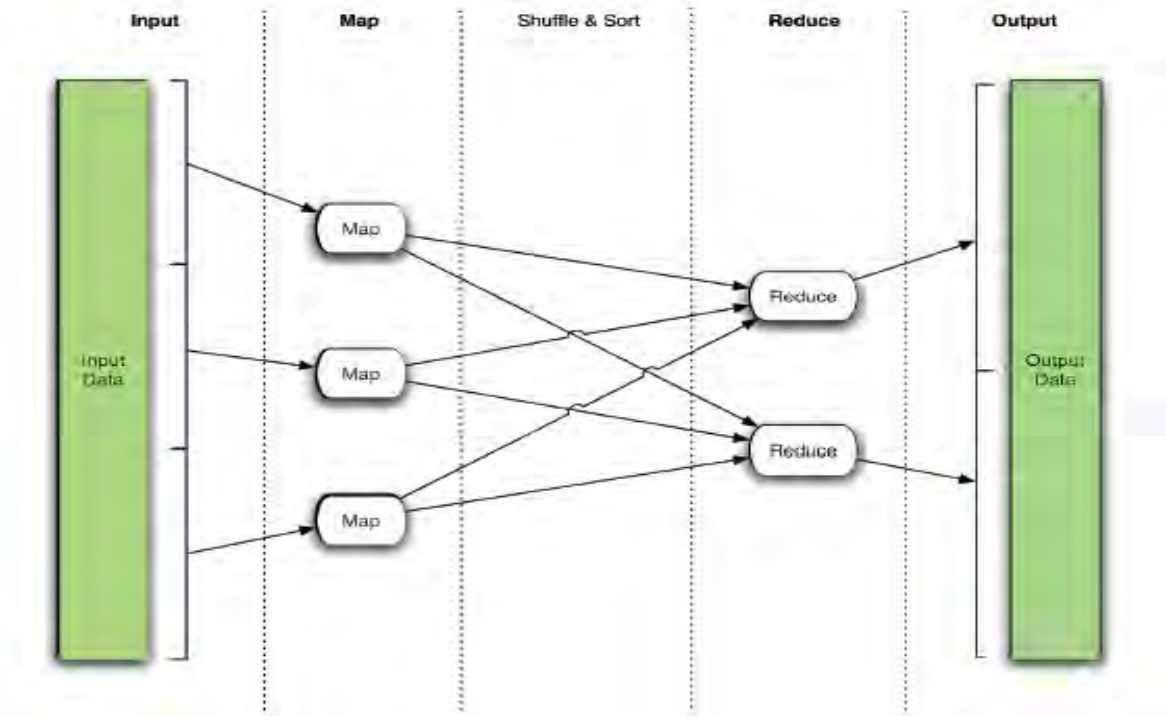
Εικόνα 2-3 Αρχιτεκτονική του HDFS

Τα προγράμματα NameNode και DataNode εκτελούνται σε κοινούς υπολογιστές. Το HDFS είναι υλοποιημένο σε Java, συνεπώς εκτελείται σε κάθε μηχανή που υποστηρίζει τη γλώσσα και ο NameNode έχει το ρόλο του συντονιστή και του repository για όλα τα HDFS metadata.

Το Map / Reduce είναι ένα πλαίσιο κατανεμημένου υπολογισμού μεγάλου όγκου δεδομένων με την χρήση cluster υπολογιστών. Η λειτουργία του μοιάζει με αυτή του pipe στα UNIX.

- cat input | grep | sort | unic –c | cat > output
- Input | Map | Shuffle & Sort | Reduce | Output

Το MapReduce χρησιμοποιείται κυρίως για: Log Processing, Web Index Building και Data mining και Machine Learning. [6]



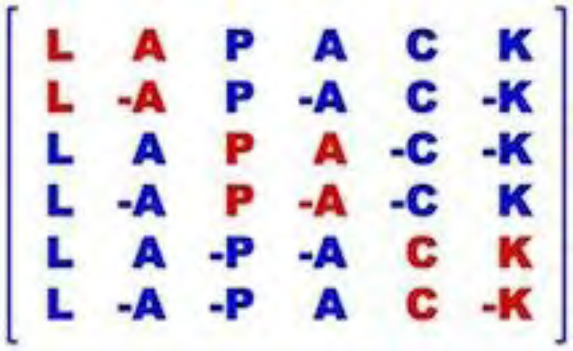
Εικόνα 2-4 Λειτουργία MapReduse

Οι βιβλιοθήκες που χρησιμοποιεί το framework MapReduce είναι γραμμένες σε γλώσσες Java, C++, Python καθώς και σε άλλες γλώσσες. Ο αλγόριθμος που χρησιμοποιείται αποτελείται από δύο βασικές λειτουργίες, την Map και την Reduce.

[6] http://wiki.apache.org/hadoop/

Κεφάλαιο 3ο

Βιβλιοθήκη Lapack



3.1 Συμβατική βιβλιοθήκη

Η LAPACK (Linear Algebra Package) είναι μια βιβλιοθήκη λογισμικού για την αριθμητική γραμμική άλγεβρα. Αρχικά κυκλοφόρησε το 1992 γραμμένη σε Fortran 77, ενώ η σημερινή έκδοση είναι γραμμένη σε Fortran 90. Εμπεριέχει ρουτίνες για την επίλυση συστημάτων γραμμικών εξισώσεων, προβλημάτων ελαχίστων τετραγώνων και προβλημάτων ιδιοτιμών, καθώς επίσης για την εφαρμογή παραγοντοποίησης LU και ανάλυσης Cholesky. Οι ρουτίνες της μπορούν να χειριστούν πραγματικούς και μιγαδικούς πίνακες τόσο απλής όσο και διπλής ακρίβειας. Οι ρουτίνες κατηγοριοποιούνται ως: driver routines, computational routines, utility routines, auxiliary routines, testing routines και timing routines.

Όσον αφορά τις Driver routines, η κάθε μία από αυτές λύνει ένα ολόκληρο πρόβλημα, όπως την επίλυση ενός γραμμικού συστήματος. Στις Computational routines, κάθε μία από αυτές ολοκληρώνει μία ξεχωριστή υπολογιστική δραστηριότητα, όπως μία LU παραγοντοποίηση, χρησιμοποιούνται απευθείας για να συμβάλλουν στην ολοκλήρωση μιας δραστηριότητας την οποία δεν μπορούν να χειριστούν οι driver ρουτίνες. Οι Utility routines υλοποιούν μη αριθμητικές λειτουργίες όπως ο χειρισμός σφαλμάτων και η σύγκριση χαρακτήρων. Οι Auxiliary routines καλούνται από τις ρουτίνες της Lapack για υπολογισμούς χαμηλού επιπέδου, όπως ο υπολογισμός της νόρμας ενός πίνακα, ενώ οι Testing routines χρησιμοποιούνται για την επαλήθευση των αριθμητικών αποτελεσμάτων

των computational ρουτινών. Τέλος, οι Timing routines χρησιμοποιούνται για την αξιολόγηση της απόδοσης των computational ρουτινών.[7]

Το όνομα της κάθε ρουτίνας είναι ενδεικτικό των χαρακτηριστικών της. Αποτελείται από 5-6 χαρακτήρες ο καθένας από τους οποίους υποδηλώνει λεπτομέρειες για τα αριθμητικά μεγέθη.

Πιο αναλυτικά, ο 1ος χαρακτήρας υποδηλώνει τον τύπο των αριθμητικών δεδομένων και μπορεί να έχει τις ακόλουθες τιμές :

- S για τους πραγματικούς αριθμούς απλής ακρίβειας
- D για τους πραγματικούς αριθμούς διπλής ακρίβειας
- C για τους μιγαδικούς αριθμούς απλής ακρίβειας και
- Z για τους μιγαδικούς αριθμούς διπλής ακρίβειας

Ο 2ος και ο 3ος χαρακτήρας υποδηλώνουν τον τύπο του πίνακα ή του πιο σημαντικού πίνακα αν υπάρχουν περισσότεροι από ένας, όπως στην περίπτωση της επίλυσης συστημάτων ενώ οι υπόλοιποι χαρακτήρες υποδηλώνουν το είδος του υπολογισμού που εκτελείται .

Για παράδειγμα, το όνομα της ρουτίνας DPOSV υποδηλώνει ότι πρόκειται για πραγματικούς αριθμούς διπλής ακρίβειας (D = Double), ο υπολογισμός είναι η επίλυση συστήματος (SV = Solve) και ο κύριος πίνακας είναι θετικά ορισμένος (PO = Positive).

Οι ρουτίνες της Lapack χρησιμοποιούν για την εκτέλεση των υπολογισμών τους τις ρουτίνες της BLAS (Basic Linear Algebra Subprograms). Η BLAS είναι μία βιβλιοθήκη που δημιουργήθηκε το 1979 και παρέχει ρουτίνες για την εκτέλεση απλούστερων υπολογισμών, όπως ο πολλαπλασιασμός διανυσμάτων και πινάκων και ο υπολογισμός της νόρμας. Η λειτουργικότητά τους χωρίζεται σε 3 επίπεδα: στο $1^{ο}$ επίπεδο, το οποίο περιλαμβάνει υπολογισμούς που αφορούν διανύσματα, το $2^{ο}$ επίπεδο, που περιλαμβάνει υπολογισμούς που αφορούν διάνυσμα και πίνακα και στο $3^{ο}$ επίπεδο περιλαμβάνονται υπολογισμοί μεταξύ πινάκων.

Όσον αφορά τη χρήση της βιβλιοθήκης Lapack ακολουθούμε κάποια συγκεκριμένα βήματα. Το πρώτο βήμα είναι η εγκατάσταση της βιβλιοθήκης στο μηχάνημα του χρήστη,

[7] http://www.netlib.org/lapack/

διαδικασία που ίσως αποδειχθεί απαιτητική. Αρχικά, πρέπει ο χρήστης να κατεβάσει το πακέτο της βιβλιοθήκης στο μηχάνημά του και στη συνέχεια να «χτίσει» εκεί τη βιβλιοθήκη. Εάν στο μηχάνημα βρίσκεται εγκατεστημένο λειτουργικό σύστημα Unix είναι απαραίτητη η εγκατάσταση gfortran και φυσικά ο χρήστης χρειάζεται να είναι εξοικειωμένος με τις εντολές κονσόλας. Εάν το μηχάνημα λειτουργεί με Windows, η διαδικασία γίνεται πιο δύσκολη καθώς απαιτείται η εγκατάσταση και χρήση διάφορων άλλων λογισμικών για το «χτίσιμο» της βιβλιοθήκης, όπως το Microsoft12 Visual Studio και Intel Fortran Compiler for Windows, όπου πρέπει να δοθεί πολλή προσοχή στη συμβατότητα των διαφόρων εκδόσεων των παραπάνω εργαλείων. Βέβαια, το συγκεκριμένο βήμα θα μπορούσε να παραληφθεί μέσω της απόκτησης έτοιμης και ήδη χτισμένης βιβλιοθήκης. Στο συγκεκριμένο τρόπο, όμως, υπάρχει ο εξής περιορισμός: τα χαρακτηριστικά του μηχανήματος στο οποίο έχει χτιστεί η συγκεκριμένη έκδοση πρέπει να είναι ίδια με του δικού μας μηχανήματος. Για παράδειγμα, αν το μηχάνημά μας έχει 32-bits λειτουργικό σύστημα δεν μπορούμε να χρησιμοποιήσουμε βιβλιοθήκη που έχει χτιστεί για σύστημα 64bits. Για αυτό το λόγο είναι προτιμότερο η βιβλιοθήκη που θα χρησιμοποιήσουμε να έχει χτιστεί στο δικό μας μηχάνημα. Μόλις η παραπάνω διαδικασία ολοκληρωθεί, ο χρήστης πρέπει να διαλέξει τη ρουτίνα η οποία ικανοποιεί τις απαιτήσεις του και συμβαδίζει με τα χαρακτηριστικά του προβλήματος.

Για παράδειγμα, στην επίλυση ενός συστήματος της μορφής AX=B, οι ιδιότητες του πίνακα A είναι αυτές που καθορίζουν ποια ρουτίνα πρέπει να χρησιμοποιηθεί. Στη συνέχεια, πρέπει να δημιουργηθεί πρόγραμμα γραμμένο σε κάποια γλώσσα προγραμματισμού της επιλογής του χρήστη, συνήθως σε Fortran, C ή C++ στο οποίο θα πρέπει να δηλωθεί ο φάκελος που έχει αποθηκευτεί η βιβλιοθήκη ώστε να είναι ορατή από τον κώδικα. Αυτό είναι εφικτό μόνο αν ο χρήστης έχει γνώσεις προγραμματισμού, διαφορετικά η χρήση της είναι αδύνατη.

3.2 Αναγκαιότητα για μετάβαση σε Lapack Web Service

Όπως έχει προαναφερθεί, η εγκατάσταση της Lapack στον προσωπικό υπολογιστή του εκάστοτε χρήστη ίσως αποδειχθεί μία αρκετά απαιτητική διαδικασία. Καταρχάς θα πρέπει στο μηχάνημα να εγκατασταθεί συμπληρωματικό λογισμικό όπως κάποια διανομή της java της fortran και γενικά ίσως υπάρξουν προβλήματα με τις εκδόσεις μεταξύ αυτών. Χρειάζεται ο χρήστης να είναι εξοικειομένος με το λειτουργικό σύστημα τύπου UNIX, τις εντολές κονσόλας και την εγκατάσταση λογισμικού σε αυτό. Άρα λοιπόν η εγκατάσταση της

βιβλιοθήκης μπορεί να αποβεί αρκετά χρονοβόρα. Στην περιπτωση που ο υπολογιστής τρέχει Windows, τότε η διαδικασία είναι αρκετά πιο απαιτητική.

Ένα άλλο πρόβλημα με την συμβατική Lapack είναι ότι είναι γραμμένη σε fortran οπότε και ο χρήστης θα πρέπει να γράψει κώδικα σε fortran για να την χρησιμοποιήσει. Γενικά οι περισσότεροι χρήστες και ειδικά οι νέοι, είναι περισσότερο εξοικειομένοι με γλώσσες προγραμματισμού όπως η java και η C, οπότε το να γράψουν τα προγράμματά τους σε fortran προσθέτει ένα επιπλέον βάρος, αυτό της εκμάθησης μίας επιπλέον γλώσσας προγραμματισμού. [8]

Επιπρόσθετα, θα πρέπει ο χρήστης να διαθέτει έναν υπολογιστή ικανό να τρέξει τα προγράμματα τα οποία μπορεί να αποδειχθούν και αρκετά απαιτητικά ανάλογα με τον όγκο των δεδομένων που θέλει να διαχειριστεί. Για παράδειγμα ίσως κάποιος χρήστης να θέλει να επεξεργαστεί έχει πίνακα Α 100x100 και ανάλογα δεξιά μέλη. Σε τέτοια περίπτωση ο υπολογιστής του ίσως να μην μπορούσε να ανταπεξέλθει ή σε καλύτερη περίπτωση, να αργούσε πάρα πολύ μέχρι να εκτελέσει τους ανάλογους υπολογσιμούς. Ειδικά αν αυτή η περίπτωση αναχθεί σε μία εταιρεία που χρείαζεται άμεσα να τρέξει κάποιους υπολογισμούς, τότε θα μιλούσαμε για καταστροφικά αποτελέσματα. Όπως εύκολα αντιλαμβάνεται κανείςη αγορά εξοπλισμού κοστίζει σε χρήματα κάτι το οποίο σίγουρα δεν θέλουν οι περισσότεροι χρήστες. Πέρα όμως από την υπολογιστική ικανότητα του υπολογιστή, σίγουρα υπάρχει και θέμα φορητότητας.Για παράδειγμα, μπορεί κάποιος να ξεκινήσει μία εργασία στον προσωπικό του σταθερό υπολογιστή, να θέλει να συνεχίσει στον φορητό του, και στην συνέχεια σε κάποιον υπολογιστή στον χώρο εργασίας του ή στο πανεπιστήμιο αν είναι φοιτητής. Στην περίπτωση της συμβατικής Lapack θα έπρεπε ο χρήστης να εγκαταστήσει τα απαραίτητα σε κάθε υπολογιστή με τα προβλήματα που ήδη αναφέραμε να πολλαπλασιάζονται. Τέλος είναι γνωστά τα προβλήματα διαθεσιμότητας που προκύπτουν συχνά σε προσωπικούς υπολογιστές λόγωιών ή όταν φορτώνονται με πολλά προγράμματα και βιβλιοθήκες. Τότε συνήθως αρχίζουν να καθυστερούν και στην χειρότερη περίπτωση να χρειάζεται να διεγραφούν όλα τα δεδομένα και να επανεγκατασταθούν όλα από την αρχή.

[8] http://en.wikipedia.org/wiki/LAPACK

3.3 Στόχοι της Lapack WS

Στην προηγούμενη ενότητα, αναφέρθηκαν τα προβλήματα τα οποία ενδέχεται να αντιμετωπίσει ένας χρήστης της συμβατικής Lapack. Προσφέροντας λοιπόν αυτήν και τις ρουτίνες της ως υπηρεσίες, τα προβλήματα αυτά απομακρύνονται από τον χρήστη.Συγκεκριμένα κάποιος θα έχει ήδη εγκαταστήσει τις βιβλιοθήκες και το απαραίτητο λογισμικό στο cloud θα έχει υλοποιήσει τις ρουτίνες έτσι ώστε να τρέχουν εκεί και θα προσφέρει τα wsdl της κάθε μίας μέσω κάποιου restistry έτσι ο επίδοξος χρήστης να εκτελεί μία κλήση προς αυτές ή να υλοποιεί ένα client πρόγραμμα σε Java ,C ή fortran. Ακόμη η υπολογιστική δύναμη των υπολογιστών που θα τρέχουν θα είναι αρκετή ώστε τα αποτελέσματα να είναι σχετικά άμεσα.

Μέσα στους στόχους είναι να προσφερθούν όσο το δυνατόν περισσότερες ρουτίνες της Lapack ως υπηρεσίες για να διευκολύνονται όλο και περισσότεροι χρήστες. Οι χρήστες θα επισκέπτονται την registry και εκεί θα βλέπουν τις υλοποιημένες. Ακόμη, επιθυμούμε να προσφέρονται διάφοροι τρόποι προσπέλασης των υπηρεσιών. Οι δύο πιο βασικοί, είναι αυτοί της υλοποίησης προγράμματος πελάτη και της απευθείας κλήσης μέσα απο κώδικα σαν συνάρτηση. Στην πρώτη περίπτωση, θα πρέπει να δημιουργήσει ένα java project μέσω του NetBeans, να το κάνει να είναι Web Service client και συγκεκριμένα να εισάγει το wsdl της ρουτίνας που θέλει να χρησιμοποιήσει. Αυτόματα θα εμφανιστεί ο κώδικας της κλησης και των δεδομένων που πρέπει να εισάγει. Στην απευθείας κλήση , τότε ο χρήστης εκτελέι μία κλήση προς την επιθυμητή ρουτίνα μέσα από ένα πρόγραμμα σαν να ήταν ρουτίνα που έχει υλοποιήσει παραπάνω στο ίδιο πρόγραμμα. Αλλος τρόπος προσπέλασης των υπηρεσιών είναι να επισκεφτεί ο χρήστης το site lapack.ws και απόεκεί μέσα από κάποιες φόρμες να δηλώσει ποιά συνάρτηση επιθυμεί να χρησιμοποιήσει, να εισάγει τα δεδομένα είτε απευθείας από το πληκτρολόγιο είτε να δώσει το url ενός αρχείου που περιέχει τα δεδομένα είτε να τα ανεβάσει από τον υπολογσιστή του. Αναλυτικά το πώς δημιουργούμε και εκτελούμε κλήσεις θα περιγραφούν ακι στο επόμενο κεφάλαιο.[9]

Ένας άλλος στόχος είναι να ανεβάσουμε τις υπηρεσίες στο amazon ws. Το amazon όπως προαναφέρθηκε, προσφέρει κάποιες υπηρεσίες και μία από αυτές είναι το EC2. Εκεί μας δίνεται η δυνατότητα να δεσμεύσουμε κάποιον εικονικό ηλεκτρονικό υπολογιστή με Linux και να το διαχειριζόμαστε μέσω της κονσόλας από τον δικό μας υπολογιστή. Εκεί λοιπόν θα εγκαταστήσουμε βιβλιοθήκες και λογισμικό που είναι απαραίτητο για να τρέχουν εκεί οι

[9] http://www.cs.colorado.edu/~jessup/lapack/

ρουτίνες που θέλουμε. Από την στιγμή που θα αρχίσουν να τρέχουν θα μπορεί ο οποιοσδήποτε να τις καλεί και να πέρνει τα αποτελέσματά του. Όλοι οι υπολογισμοί θα γίνονται στο Amazon οπότε ο υπολογιστής του χρήστη δεν επιβαρύνεται ούτε για την εγκατάσταση λογισμικού, ούτε και την εκτέλεση υπολογισμών. Ακόμη οι υπηρεσίες είναι συνεχώς διαθέσιμες, καθώς στο cloud και συγκεκριμένα το Amazon δεν απειλούνται από διακοπές ρεύματος ή από ιούς που ενδεχομένως να επηρέαζαν την λειτουργία των υπηρεσιών μας. Ακόμη μπορεί ο χρήστης να αποθηκεύει τα δεδομένα του στο Amazon S3 ή στο HDFS το οποίο θα σχολιαστεί παρακάτω και άρα να τα έχει και αυτά συνεχώς διαθέσιμα και μάλιστα πολύ εύκολα προσβάσιμα καθώς είναι πολύ εύκολο να επικοινωνήσει το ec2 με το S3 ή το HDFS το οποίο θα έχει εγκατασταθεί πάνω στο ec2.

Τελικά, θα θέλαμε να δημιουργήσουμε ένα δίκτυο ανάμεσα στους χρήστες των υπηρεσιών Lapack που το επιθυμούν, ώστε να ανταλλάσουν δεδομένα ή τα αποτελέσματα που είχαν από προηγούμενες κλήσεις που εκτέλεσαν. Θα μπορεί για παράδειγμα κάποιοι χρήστες να επιτρέπουν σε άλλους χρήστες να δουν τι δεδομένα έχουν χρησιμοποιήσει ώστε να τα χρησιμοποιήσουν και εκείνοι ή να παρουν κατευθείαν τα αποτελέσματα. Αυτό θα γίνεται διότι θα τα δεδομένα όσων το επιπθυμούν θα αποθηκεύονται στο S3 ή στο HDFS και θα επιτρέπουν την πρόσβαση και άλλων σε αυτά. Με το HDFS, το οποίο όπως είπαμε είναι ένα κατανεμημένο σύστημα αρχείων, θα υπάρχει ένας κεντρικός κόμβος master, ο λεγόμενος και Namenode, ο οποίος θα είναι στημένος σε ένα εικονικό μηχάνημα στο Amazon EC2 και από εκεί και πέρα θα μπορούμε να προσθέτουμε όσους κόμβους θέλουμε σαν slaves του master. Έτσι δίνεται η δυνατότητα να προσθέσει κάποιος σαν κόμβο ακόμα και τον ίδιο του τον προσωπικό υπολογιστή στον οποίο κρατάει τα δεδομένα του, και να τα παρέχει σε άλλους χρήστες προς χρησιμοποίηση. Έτσι δημιουργείται το δίκτυο ή cluster υπολογιστών και χρηστών των υπηρεσιών που προσφέρουμε.

Κεφάλαιο 4ο

Προϋπάρχουσα έρευνα

Έχει προταθεί μια διαφορετική προσέγγιση για την εκτέλεση επιστημονικών υπολογισμών, η οποία δίνει λύση σε διάφορα διαπιστωμένα προβλήματα. Το μοντέλο, που έχει εισαχθεί για την εκτέλεση επιστημονικών υπολογισμών, περιλαμβάνει την έκθεση μεθόδων (legacy code) για επιστημονικούς υπολογισμούς ως Web Services και την ανάπτυξη εφαρμογών για τη διενέργεια αυτών με τη χρήση επιστημονικών ροών εργασίας.

Βάση έχει δοθεί στην έκθεση βιβλιοθηκών για scientific computing γραμμένων σε παραδοσιακές γλώσσες προγραμματισμού, συγκεκριμένα σε fortran77, ως Web Services. Με την παροχή βιβλιοθηκών, ως υπηρεσιών προσβάσιμων μέσω στάνταρντ πρωτοκόλλων του Διαδικτύου, όπως το HTTP και το SOAP, οι οποίες αποτελούν και τον κύριο κορμό του scientific computing, αλλάζει ριζικά η υφιστάμενη κατάσταση στον τρόπο με τον οποίο εκτελούνται οι επιστημονικοί υπολογισμοί.

Οι υπηρεσίες αυτές μπορούν να ανακαλυφθούν δυναμικά από μία εφαρμογή κατά το χρόνο εκτέλεσης μέσω κάποιου UDDI repository (βλ. 3.1.3). Επίσης, υιοθετούν ένα μοντέλο χαλαρά συνδεδεμένων συστημάτων, το οποίο επιτυγχάνει διαλειτουργικότητα (ανεξαρτησία γλώσσας πλατφόρμας) σε ένα ανομοιογενές περιβάλλον, όπως είναι το Διαδίκτυο.

Η έκθεση της προγραμματιστικής λογικής τέτοιων βιβλιοθηκών ως Web Services επιτυγχάνει μεγάλη διαφάνεια για τον τελικό χρήστη, καθώς μπορεί να έχει πρόσβαση σε αυτές μέσω στάνταρντ πρωτοκόλλων του Διαδικτύου. Η ίδια υπηρεσία μπορεί, για παράδειγμα, να τρέχει τοπικά σε κάποιο server, σε κάποιο computational cluster ή σε grid περιβάλλον, λεπτομέρειες που δεν απασχολούν τον τελικό χρήστη, καθώς ο τρόπος πρόσβασης παραμένει ίδιος.

Με τη χρήση αυτούσιων βιβλιοθηκών επιστημονικού υπολογισμού αποφεύγουμε τον επανασχεδιασμό τους, ο οποίος έχει μεγάλο κόστος, και εκμεταλλευόμαστε την αποδεδειγμένη υψηλή αποδοτικότητα. Επιπλέον, με τη χρήση ενός τέτοιου μοντέλου, ο

τελικός χρήστης αποκτά πρόσβαση σε υπολογιστική ισχύ (cluster ή grid περιβάλλον) με διαφανή τρόπο.

Γενικά έγινε επεξεργασία των παρακάτω ζητημάτων:

1. Παράθεση των μεθόδων της LAPACK, γραμμένων σε fortran77, ως Web Services.

2. Σχεδιασμός των ροών εργασίας με τη χρήση της γλώσσας BPEL.

3. Μελέτη των δυνατοτήτων για ανάπτυξη Web Services στην υποδομή του Hellas Grid EGEE.

Τα προβλήματα που αντιμετωπίστηκαν, με αποτέλεσμα να μην υλοποιηθεί ολοκληρωτικά η ζητούμενη υπηρεσία, είναι ότι κατά την προσπάθεια μεταφοράς των αρχείων από την εφαρμογή μας στον WMS server, ενώ υπήρχε ασφαλής σύνδεση ssl, δεν επιτρεπόταν η αποστολή του αρχείου.

Το EGEE, και κατ' επέκταση το Hellas Grid, το οποίο είναι υποσύνολό του, είναι μία πολλά υποσχόμενη τεχνολογία. Διαπιστώθηκε ότι το ένα από τα προβλήματα, που παρουσιάζει, είναι η δυσκολία στη χρήση και στην ανάπτυξη εφαρμογών που θα εκμεταλλεύονται αυτήν την τεράστια υπολογιστική ισχύ. Πιο συγκεκριμένα, την παρούσα χρονική στιγμή λείπουν τα απαραίτητα εργαλεία που θα κάνουν ευκολότερη τη χρήση και την ανάπτυξη εφαρμογών σε αυτήν. Προς τα εκεί μπορούν να στραφούν ερευνητικές προσπάθειες για την ανάπτυξη εργαλείων και API's, που κρύβουν από το χρήστη την πολυπλοκότητα και την ευμεταβλητότητα ενός Grid περιβάλλοντος και θα το διαδώσουν ευρέως στην επιστημονική κοινότητα. [3]

Σε μια άλλη έρευνα, έγινε αντιληπτό κατά την επεξεργασία της βιβλιοθήκης της Lapack, ότι πρόκειται για μια χρήσιμη βιβλιοθήκη για την επίλυση αριθμητικών προβλημάτων της οποίας η εγκατάσταση και χρήση στο μηχάνημα του χρήστη ίσως αποδειχθεί δύσκολη ή και ακατόρθωτη σε περίπτωση που ο χρήστης δε διαθέτει κατάλληλες γνώσεις. Εξαιτίας της πολυπλοκότητας της παραπάνω διαδικασίας, θεωρήθηκε σκόπιμο να είναι διαθέσιμη η βιβλιοθήκη αυτή ως υπηρεσία διαδικτύου. Με αυτό τον τρόπο, παρακάμπτεται το βήμα της εγκατάστασης της στο μηχάνημα και πλέον ο χρήστης αρκεί να καλέσει την αντίστοιχη υπηρεσία διαδικτύου. Για την υλοποίηση υψηλής απόδοσης υπολογισμών οι υπηρεσίες δεν προσφέρονται μέσω ενός απλού desktop personal computer αλλά μέσω ενός desktop cluster 48 πυρήνων όπου έχει εγκατασταθεί ο Glassfish Application Server. Για την

ανακάλυψη και κλήση των προσφερόμενων υπηρεσιών απαιτείται η δήλωση και δημοσίευσή τους σε ένα registry. Μέχρι πρότινος, η συγκεκριμένη διαδικασία γινόταν μέσω του UDDI που είχε δημιουργηθεί για αυτό ακριβώς το σκοπό. Να εξυπηρετεί δηλαδή στην εγγραφή και δήλωση οργανισμών έτσι ώστε να μπορεί οποιαδήποτε επιχείρηση να ανακαλύψει τον κατάλληλο συνεργάτη και να επικοινωνήσει μαζί του. Στη συγκεκριμένη εργασία αντί του συγκεκριμένου registry χρησιμοποιείται το ebXML, ένα framework ηλεκτρονικού εμπορίου με σύνθετα χαρακτηριστικά και περισσότερες δυνατότητες συγκριτικά με το UDDI.

Επίσης, για την αυτοματοποίηση της διαδικασίας και διευκόλυνση του χρήστη, σχεδιάστηκε και υλοποιήθηκε ένα γραφικό περιβάλλον όπου μπορεί ο κάθε χρήστης να επιλέξει την κατάλληλη ρουτίνα μέσα από μία σειρά ρουτινών επίλυσης γραμμικών συστημάτων και να την καλέσει. Γενικά η εργασία ασχολήθηκε με τα παρακάτω ζητήματα:

1. Περιγραφή της διαδικασίας για την κλήση Lapack ρουτινών μέσω υπηρεσιών διαδικτύου

2. Μελέτη των χαρακτηριστικών του ebXML και χρήση του ως web service registry

3.Σχεδίαση και υλοποίηση γραφικού περιβάλλοντος για την κλήση υπηρεσιών. [2]

Έγινε μια περιγραφή της βιβλιοθήκης της Lapack και μελετήθηκαν τα προβλήματα που προκύπτουν κατά τη χρήση της. Έγινε αναφορά στην έννοια του επιστημονικού υπολογισμού και τις απαιτήσεις του σε υπολογιστική ισχύ. Έγινε επεξεργασία της εξέλιξης των διαδικτυακών εφαρμογών και περιγραφή των συστατικών των υπηρεσιών διαδικτύου όπως και της διαδικασίας με την οποία μπορεί μία υπηρεσία διαδικτύου υλοποιημένη σε γλώσσα προγραμματισμού Java να καλεί ρουτίνα γραμμένη σε Fortran. Μελετήθηκαν τα χαρακτηριστικά του ebXML Registry το οποίο και χρησιμοποιήθηκε για τη δήλωση των υπηρεσιών μας και αναλύσαμε τις δυνατότητες που προσφέρει η χρήση του.

Υλοποιήθηκε μία γραφική διεπαφή χρήστη για την κλήση των υπηρεσιών που δημιουργήθηκαν, η οποία του προσφέρει επίσης τη δυνατότητα να ανακαλύψει την κατάλληλη για τους υπολογισμούς του ρουτίνα. Υπάρχουν πολλές προοπτικές εξέλιξης και σημεία που επιδέχονται βελτίωσης. Παρακάτω αναφέρονται τομείς που πρέπει να βελτιωθούν.

- ✓ Υλοποίηση περισσότερων υπηρεσιών της Lapack.
- ✓ Υλοποίηση βοηθητικών υπηρεσιών της Blas.

- ✓ Συνδυασμός υπηρεσιών για τη δημιουργία ροών εργασιών με τη χρήση της WS-BPEL.
- ✓ Βελτίωση του GUI.
- ✓ Οργάνωση των μεταδεδομένων στο ebXML Registry.
- ✓ Δυνατότητα πρόβλεψης του χρόνου ολοκλήρωσης ενός υπολογισμού.
- ✓ Εγκατάσταση των υπηρεσιών και σε άλλα υψηλής απόδοσης συστήματα όπως το Hellas Grid[10]

[10] http://www.hellasgrid.gr/infrastructure/index.php?language=el

Κεφάλαιο 5ο

5.1 Γενικό πλαίσιο της παρούσας εργασίας

Για την παρούσα εργασία , χρειάστηκε πρώτα από όλα να μελετηθεί το επίπεδο στο οποίο βρισκόταν μέχρι στιγμής η προσφορά της Lapack ως web service. Γι αυτόν τον λόγο μελετήθηκαν τα βασικά περί των web services, όπως το τι ακριβώς περιλαμβάνουν και πώς υλοποιούνται αλλά και προαπαιτούμενες έννοιες όπως αυτή της xml. Πάνω σε αυτά υλοποιήθηκαν μικρά παραδείγματα για εξοικείωση. Χρειάστηκε να μελετηθούν προηγούμενες μελέτες όπως οι Υψηλής Απόδοσης Υπηρεσίες διαδυκτύου για Επιστημονικούς Υπολογισμούς της Γρηγορίας Κατσογιάννου και οι Επιστημονικοί Υπολογισμοί στον Παγκόσμιο Ιστό του Σαλούστρου. Ακόμη, μελετήθηκαν και χρησιμοποιήθηκαν το site που έχει στηθεί για την Lapack web services και ρουτίνες που είχαν ήδη υλοποιηθεί και τρέχαν στο cluster του πανεπιστημίου. Στην συνέχεια αναπτύχθηκαν δύο ρουτίνες της Lapack ως υπηρεσίες διαδικτύου. Η διαδικασία περιγράφεται παρακάτω. Ακόμη, οι ρουτίνες ανέβηκαν σε application server που είχε στηθεί σε ένα micro instance του amazon ec2. Στόχος ήταν να μελετηθεί το πώς μπορεί να συνδεθεί η χρήση των υπηρεσιών διαδικτύου που θα παρέχαμε με την χρήση του HDFS σαν κατανεμημένο σύστημα αρχείων. Έτσι μελετήθηκαν διάφορα κομμάτια του Hadoop όπως το HDFS, το Cassandra και το MapReduce. Τελικά μόνο το HDFS ήταν που χρειάστηκε να μελετηθεί στο πλαίσιο της μελέτης μας. Για να καταλήξουμε στο πώς αλλά και στο αν τελικά αξίζει να χρησιμοποιηθεί το HDFS για την αποθήκευση των δεδομένων που θέλει ο εκάστοτε χρήστης να αποθηκεύσει, όπως για παράδειγμα πινάκων Α και δεξιών μελών Β, έπρεπε να στηθεί το παραπάνω σύστημα αρχείων στο ec2 και σε άλλους υπολογιστές. Για να εκτελέσουμε όμως δοκιμές με πίνακες μεγάλου μεγέθους δεν επαρκούσε το micro instance λόγω της περιορισμένης μνήμης που παρέχει. Καταλήξαμε στο να αναβάσουμε κάποια συγκεκριμένη ρουτίνα, η οποία δέχεται τετραγωνικούς πίνακες ως είσοδο, στο cluster του πανεπιστημίου όπου είναι σαφώς μεγαλύτερων δυνατοτήτων από το micro instance του ec2. Στήθηκε το HDFS και εκτελέστηκαν οι δοκιμές που είχαμε σχεδιάσει. Ακόμη εκτελέστηκαν αντίστοιχες δοκιμές και σε ec2 instance μεγαλύτερης χωρητικότητας ώστε να καταλήξουμε στο αν αξίζει να τρέχουν εκεί οι Lapack web services. Στην συνέχεια περιγράφονται αναλυτικά τα βήματα που ακολουθήθηκαν για να συντελεστούν τα παραπάνω.

5.2 Υλοποίηση Lapack web service

Αρχικά να αναφερθεί ότι για την υλοποίηση όλων των παρακάτω βημάτων, χρησιμοποιήθηκε λειτουργικό σύστημα ubuntu 10.04 και NetBeans 6.7.1.

Για να υλοποιηθεί οποιαδήποτε ρουτίνα ως web service, θα πρέπει να υπάρχουν οι βιβλιοθήκες της Lapack και του Blas εγκατεστημένες στο μηχάνημα που θα γίνει η υλοποίηση. Κατεβάζουμε το κατάλληλο αρχείο από το http://www.netlib.org/lapack/ και κατεβάζουμε το αρχείο lapack .tgz. Το κάνουμε untar με την εντολή : gunzip –c lapack.tgz | tar xvf -. Στην συνέχεια εκτελούμε στον φακελο που δημιουργήθηκε το cp make.inc.example make.inc και τέλος την εντολή make. Αν δεν δημιουργηθεί η βιβλιοθήκη του blas τότε εκτελούμε make blaslib. Μεταφέρουμε τα αρχεία blas_LINUX.a, και lapack_LINUX.a στο /usr/lib. Στην συνέχεια είμαστε έτοιμοι να δημιουργήσουμε το service για όποια ρουτίνα επιθυμούμε. Ανοίγουμε το NetBeans και δημιουργούμε ένα νέο project σαν java application και συγκεκριμένα Web Application. Δίνουμε το όνομα που θέλουμε (για παράδειγμα DgesvWs) και στην συνέχεια πατάμε Finish. Όταν δημιουργηθεί κάνουμε πάνω του δεξί κλικ και New Web Service. Δίνουμε το όνομα που θέλουμε να έχει η υπηρεσία και σαν package βάζουμε uth.sws4hpsc.lapack.

Στην συνέχεια πηγαίνουμε μέσω του τερματικού στο φάκελο που έχει δημιουργηθεί εκεί που αποθηκεύονται τα projects του NetBeans και συγκεκριμένα στον φάκελο java του project που μόλις δημιουργήσαμε. Εκεί δημιουργούμε ένα αρχείο c με όνομα c_degesv.c το οποίο χρησιμοποιείται ως wrapper αρχείο μεταξύ της java και της fortran στην οποία είναι γραμμένη η συνάρτηση που μας αφορά. Μέσα σε αυτό το αρχείο καλείται η fortran routine, της δίνουμε τα δεδομένα που παίρνουμε από την java και επιστρέφουμε στο service τα αποτελέσματα που επιστρέφει η fortran. Στην συνέχεια προσθέτουμε την κλήση προς την native routine της C που μόλις δημιουργήσαμε στο project στο NetBeans και αποθηκεύμουμε. Κάνουμε compile το .java αρχείο με την εντολή javac DgesvWs.java και μετά στον φάκελο που είναι το .c αρχείο δημιουργούμε ένα .h αρχείο με την εντολή javah –classpath .-jni uth.sws4hpsc.lapck.DgesvWs. Προσθέτουμε την κλήση προς αυτό το αρχείο μέσα στο wrapper της c καιεκτελούμε :

```
Cc –c –I/<path_to_jdk>/include -I/<path_to_jdk>/include/linux
c_dgesv.c.
```

Τέλος δημιουργούμε την βιβλιοθήκη για την υπηρεσία ώστε να είναι δυνατή η κλήση της fortran routine με την εντολή:

```
Ld –G-z defs c_dgesv.o –o libsws4hpscDgesv.so
<path_to_lapack_library> /usr/lib/libgfortran.so.3 –lm –lc.
```

Από εδώ και πέρα συνεχίζουμε με την προσθήκη της λειτουργικότητας στο service που έχουμε δημιουργήσει στο NetBeans. Για να γίνει αυτό, πρώτα πρέπει το service μας να γίνει WebServiceClient (δεξί κλίκ στο project->New->WebServiceClient) και εκεί προσθέτουμε το url των IdManager, DataManager,DirectoryManager και PathManager. Το url τους αν πρόκειται για αυτά που τρέχουν στον application server του cluster είναι της μορφής :

http://195.251.17.2:8080/<manager_name>/<manager_name>Service?wsdl

Ακόμη, πρέπει να προσθέσουμε στις libraries του project μας και το Lapack Base Service το οποίο κατεβάζουμε από την registry που έχει αποθηκευτεί και μπορούμε να την βρούμε στο lapack.ws έχοντας προηγουμένως προσθέσει στους hosts του υπολογιστή μας την διεύθυνση 195.251.17.2 με το όνομα lapack.ws. Το LapackBaseService θα αποθηκευτεί σαν .jar αρχείο και το προσθέτουμε στο project κάνοντας δεξί κλικ->properties->Libraries->add jar file, το προσθέτουμε και πατάμε ok. Τελειώνουμε προσθέτοντας ότι λειτουργικότητα θέλουμε να έχει το service μας.

Για να δοκιμάσουμε αν το service έχει ολοκληρωθεί με επιτυχία, αλλά και για να προσφέρουμε τις υπηρεσίες του και σε άλλους, πρέπει να το ανεβάσουμε σε έναν application server και να τρέχει εκεί. Εμείς επιλέξαμε τον glassfish 2.1. Για να τον κατεβάσουμε μπαίνουμε στο site glassfish.java.net/public/downloadsindex.html και κατεβάζουμε την έκδοση που μας ενδιαφέρει. Στον φάκελο που κατέβηκε, εκτελούμε

```
Ld Java –Xmx256m –jar.
```

Μεταβαίνουμε στον φάκελο glassfish και εκτελούμε

```
chmod –R +x lib/ant/bin
lib/ant/bin/ant –f setup.xml.
```

Τέλος κάνουμε deploy το .war αρχείο που δημιουργείται στον φάκελο dist μέσα στον φάκελο του project μας στον glassfish server, δημιουργούμε ένα client java program, κάνουμε add WebServiceClient με δεξί κλικ στο νεό project και προσθέτουμε το wsdl του service από τον glassfish. Ολοκληρώνουμε προσθέτοντας τα απαιτούμενα στον client τρέχοντας τον παίρνουμε τα ανάλογα αποτελέσματα.

5.3 Deploy της υπηρεσίας μας στο ec2

Για να δώσουμε επιπλέων δυνατότητες στο στην υπηρεσία μας όπως καλήτερη διαθεσιμότητα και περισσότερη υπολογιστική δύναμη, αποφασίσαμε ότι το ec2 του amazon είναι μία πολύ καλή λύση καθώς όλοι έχουν πρόσβαση προς αυτό, είναι σχεδόν απίθανο να «πέσει» το instance που φιλοξενεί το service μας και γενικά συμβαδίζει με τη έννοια του cloud και τα προτερήματά του τα οποία έχουν ήδη αναλυθεί.

Για ξεκινήσουμε με το ec2, αρχικά θα πρέπει να ανοίξουμε έναν λογαριασμό. Καλό θ ήταν να ενημερωθούμε για τις χρεώσεις που έχει το amazon από το αντίστοιχο θέμα. Γενικά ακολουθεί την λογική του «πληρώνεις ανάλογα με αυτό που χρησιμοποιείς». Στην συνέχεια είμαστε έτοιμοι να δεσμεύσουμε έναν εικονικό υπολογιστή, βάση των δυνατοτήτων και χαρακτηριστικών που θέλουμε να έχει. Εμείς επιλέξαμε το micro instance που είναι δωρεάν. Αρχικά πηγαίνουμε στην κονσόλα και επιλέγουμε ec2. Πατάμε στο Launch Instance. Στην συνέχεια επιλέγουμε με βάση τα χαρακτηριστικά που θέλουμε να έχει το instance. Εμείς θα επιλέξουμε το Basic 32-bit Amazon Linux AMI. Στην επόμενη σελίδα επιλέγουμε το micro instance αν θέλουμε να είναι δωρεάν και στην συνέχεια πατάμε σε όλα next μέχρι που ολοκληρώνεται η δέσμευση. Είμαστε έτοιμοι να χρησιμοποιήσουμε το instance που δεσμεύσαμε. Για να γίνει αυτό, ανοίγουμε την κονσόλα του υπολογιστή μας και πληκτρολογούμε ssh –i <permission_key_name>.pem ec2-user@<instance_pubic_DNS>. Το permission key και το public DNS τα παίρνουμε από την console στο amazon. Όταν γίνει η σύνδεση με ssh η κονσολα μας πλέον αναφέρεται στο instance. Είναι σαν να χρησιμοποιούμε έναν υπολογιστή που έχει linux λειτουργικό πάνω.

Εμείς θέλουμε ο συγκεκριμένος υπολογιστής να έχει τον glassfish server και γι αυτόν τον λόγο τον εγκαθιστούμε στο path /usr/local. Η εγκατάσταση γίνεται ακριβώς όπως έγινε και στον δικό μας υπολογιστή μόνο που για να κατεβάσουμε το .jar αρχείο χρησιμοποιούμε την εντολή sudo wget <download_path>.

Στην συνέχεια ανοιγουμε το web interface του application server γράφοντας στον browser <instance_ip>:4848. Για να ανεβάσουμε την υπηρεσία μας πρώτα πρέπει να περάσουμε την βιβλιοθήκη που θα χρησιμοποιε. Για να γίνει αυτό εκτελούμε μία sftp σύνδεση με το instance. Πρώτα όμως προσθέτουμε το κλειδί που χρησιμοποιούμε και για το ssh στο αρχείο ssh_config στον φάκελο /etc/ssh. Αφού επιτευχθεί η σύνδεση γράφουμε στο terminal

```
Put <path_to_library_>/libsws4hpscDgesv.so /home/ec2-user
```

Στο service μας, αλλάζουμε την κλήση προς την βιβλιοθήκη και βάζουμε το νέο path που αντιστοιχεί στο /home/ec2-user. Κάνουμε deploy το service μας όπως και πριν και παίρνουμε το wsdl το οποίο προσθέτουμε στον client για να καλεί το service που ανεβάσαμε στο instance και εκτελούμε το client program.

5.4 Εγκατάσταση Hadoop

Στην συνέχεια περιγράφουμε την εγκατάσταση του Hadoop σε ένα μηχάνημα ή για λειτουργία multi node. Στην μελέτη μας χρησιμοποιήσαμε την έκδοση 0.20.2 η οποία κυκλοφόρησε τον Φεβρουάριο του 2010. Πρέπει να έχουμε εγκατεστημένη μία εκδοση της java από την 1.5.x και μετά. Συνίσταται να έχουμε την 1.6.x. Θα χρησιμοποιήσουμε ένα ειδικό λογαριασμό χρήστη Hadoop για ν ατρέξουμε το Hadoop. Αν και δεν απαιτείται, συνιστάται επειδή βοηθά να διαχωρίσει την εγκατάσταση του Hadoop από άλλες εφαρμογές λογισμικού και από τους λογαριασμούς χρηστών που τρέχουν στο ίδιο μηχάνημα (ασφάλεια, δικαιώματα, αντίγραφα ασφαλείας, κλπ).

```
$ Sudo addgroup Hadoop
$ Sudo adduser - ingroup Hadoop hduser
```

Αυτό θα προσθέσει ο χρήστης hduser και η ομάδα Hadoop σε τοπικό υπολογιστή σας.

Διαμόρφωση SSH

Το Hadoop απαιτεί πρόσβαση SSH για τη διαχείριση των κόμβων, δηλαδή τα απομακρυσμένα μηχανήματα και τον τοπικό υπολογιστή σας, εάν θέλετε να χρησιμοποιήσετε το Hadoop σε αυτό. Για τον single-node setup του Hadoop, λοιπόν, θα πρέπει να διαμορφώσουμε τη SSH πρόσβαση στο localhost για τον hduser χρήστη που δημιουργήσαμε στην προηγούμενη ενότητα.

Υποθέτουμε ότι έχουμε εγκατεστημένο το SSH και τρέχει στον υπολογιστή μας και έχει ρυθμιστεί να επιτρέπει το SSH δημόσιο κλειδί ταυτότητας. Αν όχι, υπάρχουν διαθέσιμες αδηγίες.

Κατ 'αρχάς, πρέπει να δημιουργήσουμε ένα SSH κλειδί για τον hduser χρήστη.

```
user@ubuntu:~$ su - hduser
hduser@ubuntu:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
```

Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
9b:82:ea:58:b4:e0:35:d7:ff:19:66:a6:ef:ae:0e:d2 hduser@ubuntu
The key's randomart image is:
[...snipp...]
hduser@ubuntu:~$

Η δεύτερη γραμμή θα δημιουργήσει ένα ζεύγος κλειδιών RSA με κενό κωδικό πρόσβασης. Σε γενικές γραμμές, χρησιμοποιώντας ένα κενό κωδικό πρόσβασης, δεν συνιστάται, αλλά στην περίπτωση αυτή είναι απαραίτητη για να ξεκλειδώσετε το κλειδί χωρίς τη δική σας εμπλοκή (δεν θέλετε να εισάγετε την συνθηματική φράση κάθε φορά που Hadoop αλληλεπιδρά με τους κόμβους του).

Δεύτερον, πρέπει να ενεργοποιήσετε τη SSH πρόσβαση στο τοπικό υπολογιστή σας με αυτό το κλειδί που μόλις δημιουργήθηκε.

hduser @ ubuntu: ~ $ cat $ HOME / .ssh / id_rsa.pub>> $ HOME / .ssh / authorized_keys

Το τελικό βήμα είναι να δοκιμαστεί η εγκατάσταση του SSH μέσω της σύνδεσης με τον τοπικό υπολογιστή σας με το hduser χρήστη. Σε αυτό το βήμα είναι επίσης απαραίτητο να αποθηκεύσετε τα τοπικά κλειδιά του hduser χρήστη ως known_hosts αρχείο. Αν έχετε οποιεσδήποτε ειδικές SSH διαμόρφωσης για τον τοπικό υπολογιστή σας σαν μια μη non-standard SSH port, μπορείτε να ορίσετε επιλογές host-specific στο $ HOME / .ssh / config (βλ. man ssh_config για περισσότερες πληροφορίες).

hduser@ubuntu:~$ ssh localhost
The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is d7:87:25:47:ae:02:00:eb:1d:75:4f:bb:44:f9:36:26.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Linux ubuntu 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC 2010 i686 GNU/Linux
Ubuntu 10.04 LTS
[...snipp...]
hduser@ubuntu:~$

Αν η SSH σύνδεση αποτύχει, αυτές οι γενικές συμβουλές μπορεί να βοηθήσουν:

- Ενεργοποίηση εντοπισμού σφαλμάτων με ssh -vvv localhost και αναλυτική διερεύνηση του σφάλματος

- Ελέγξτε τη διαμόρφωση του διακομιστή SSH στο /etc/ssh/sshd_config , και ιδίως τις επιλογές PubkeyAuthentication (η οποία θα πρέπει να ρυθμιστεί στο yes) και AllowUsers (αν αυτή η επιλογή είναι ενεργή, προσθέστε το hduser σε αυτή). Αν έχετε πραγματοποιήσει αλλαγές στο αρχείο ρυθμίσεων του διακομιστή SSH, μπορείτε να επιβάλετε μια διαμόρφωση reload με sudo / etc / init.d / ssh reload .

Απενεργοποιώντας το IPv6

Ένα πρόβλημα με το IPv6 στο Ubuntu είναι ότι η χρήση 0.0.0.0 για τα διάφορα δίκτυα που σχετίζονται με τις επιλογές διαμόρφωσης Hadoop θα έχει ως αποτέλεσμα τη σύνδεση με τις IPv6 διευθύνσεις του Ubuntu.

Στην περίπτωσή αυτή, δεν υπάρχει κανένα πρακτικό νόημα της υιοθέτησης του IPv6 όταν δεν είμαστε συνδεδεμένοι σε οποιοδήποτε δίκτυο IPv6. Ως εκ τούτου, απενεργοποιώ το IPv6 στο Ubuntu μηχανήμά μου.

Για να απενεργοποιήσετε το IPv6 στο Ubuntu 10.04 LTS, ανοίξτε το αρχείο / etc / sysctl.conf στον editor της επιλογής σας και προσθέστε τις ακόλουθες γραμμές στο τέλος του αρχείου:

```
#disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Θα πρέπει να επανεκκινήσετε τον υπολογιστή σας για να κάνετε τις αλλαγές αποτελεσματικές.
Μπορείτε να ελέγξετε εάν το IPv6 είναι ενεργοποιημένο στον υπολογιστή σας με την ακόλουθη εντολή:

```
$ Cat / proc/sys/net/ipv6/conf/all/disable_ipv6
```

Μια τιμή που επιστρέφει 0 σημαίνει ότι το IPv6 είναι ενεργοποιημένο, η τιμή 1 σημαίνει ότι είναι απενεργοποιημένο (που είναι το ζητούμενο).

Εναλλακτική λύση

Μπορείτε επίσης να απενεργοποιήσετε το IPv6 μόνο για το Hadoop, όπως τεκμηριώνεται στο **Hadoop-3437** . Μπορείτε να το κάνετε προσθέτοντας την ακόλουθη γραμμή στο conf/ Hadoop-env.sh :

```
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
```

Εγκατάσταση

Θα πρέπει να **κατεβάσετε Hadoop** από το **Apache Download Mirrors** και να αποσυμπιέστε τα περιεχόμενα του πακέτου Hadoop σε μια θέση της επιλογής σας. Επέλεξαμε /usr/local/Hadoop . Φροντίστε να αλλάξετε την κυριότητα όλων των αρχείων στο hduser χρήστη και στο Hadoop group, για παράδειγμα:

```
$ Cd / usr / local
$ Sudo tar xzf Hadoop-0.20.2.tar.gz
$ Sudo mv Hadoop-0.20.2 Hadoop
$ Sudo chown-R hduser: Hadoop Hadoop
```

Ενημέρωση $ HOME / .bashrc

Προσθέστε τις ακόλουθες γραμμές στο τέλος του $ HOME / .bashrc αρχείο του χρήστη hduser . Εάν χρησιμοποιείτε ένα shell, εκτός από bash, θα πρέπει φυσικά να ενημερώσετε τα κατάλληλα αρχεία ρυθμίσεων του αντί . bashrc .

```
# Set Hadoop-related environment variables
export HADOOP_HOME=/usr/local/hadoop

# Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later on)
export JAVA_HOME=/usr/lib/jvm/java-6-sun

# Some convenient aliases and functions for running Hadoop-related commands
unalias fs &> /dev/null
alias fs="hadoop fs"
unalias hls &> /dev/null
alias hls="fs -ls"

# If you have LZO compression enabled in your Hadoop cluster and
# compress job outputs with LZOP (not covered in this tutorial):
# Conveniently inspect an LZOP compressed file from the command
# line; run via:
#
# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
```

```
#
# Requires installed 'lzop' command.
#
lzohead () {
   hadoop fs -cat $1 | lzop -dc | head -1000 | less
}

# Add Hadoop bin/ directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin
```

Από δύο single-node clusters σε ένα multi-node cluster

Εμείς θα οικοδομήσουμε ένα multi-node cluster δύο υπολογιστές με Ubuntu. Ο καλύτερος τρόπος για αρχή είναι να εγκαταστήσουμε, να ρυθμίσουμε και να δοκιμάσουμε ένα "local" Hadoop setup για καθένα από Ubuntu boxes, και σε ένα δεύτερο βήμα για να τα συγχωνεύετε σε ένα multi-node cluster στο οποίο το ένα Ubuntu box θα γίνει master (αλλά επίσης να ενεργεί ως slave σε σχέση με την αποθήκευση και επεξεργασία δεδομένων), και το άλλο κουτί θα γίνει μόνο ένας slave. Είναι πολύ πιο εύκολο να εντοπίσετε τυχόν προβλήματα που μπορούν να οφείλονται στη μείωση της πολυπλοκότητας κατά την εγκατάσταση ενός μόνο κόμβου του συμπλέγματος για πρώτη φορά σε κάθε μηχάνημα.

Συνιστάται να χρησιμοποιήσετε τις ίδιες ρυθμίσεις (π.χ., θέσεις εγκατάστασης και μονοπάτια) και στις δύο μηχανές, ή αλλιώς μπορεί να αντιμετωπίσετε προβλήματα αργότερα όταν θα μεταναστεύσουν οι δύο μηχανές στον τελικό multi-node cluster setup.

Απλά να έχετε κατά νου κατά τη δημιουργία του single-node clusters που αργότερα θα συνδεθεί και θα συγχωνεύσει τα δύο μηχανήματα, οπότε επιλέξτε λογικές ρυθμίσεις δικτύου τώρα για μια ομαλή μετάβαση αργότερα.

Τώρα που έχετε δύο single-node clusters και τρέχουνε, θα τροποποιήσεουμε τη διαμόρφωση του Hadoop ώστε να κάνουμε το ένα Ubuntu box "master" (το οποίο θα ενεργεί επίσης ως σκλάβος) και το άλλο Ubuntu box ως "slave".

Θα καλέσετε την προκαθορισμένη συσκευή master μόνο ώς **master** από τώρα και στο εξής και το άλλο μηχάνημα το **slave** . Θα δώσετε επίσης τις δύο μηχανές αντίστοιχα hostnames στο **/ etc / hosts** .

Τερματιζετε κάθε απλό κόμβο συμπλέγματος με το / bin / stop-all.sh πριν συνεχίσετε, εάν δεν το έχετε κάνει ήδη.

Δικτύωση

Πρέπει να επισημανθεί ότι και οι δύο μηχανές θα πρέπει να είναι σε θέση να επικοινωνήσουν μέσω του δικτύου. Το πιο εύκολο είναι να βάλετε τα δύο μηχανήματα στο ίδιο δίκτυο, σε σχέση με υλικό και τη διαμόρφωση του λογισμικού, για παράδειγμα, συνδέστε τις δύο συσκευές μέσω ενός μόνο διανομέα ή διακόπτη και να ρυθμίσετε τις κάρτες δικτύου για να χρησιμοποιήσετε ένα κοινό δίκτυο, όπως 192.168.0.x/24 .

Απλοποιώντας, θα εκχωρήσετε τη διεύθυνση IP 192.168.0.1 για το master μηχάνημα και 192.168.0.2 για το slave μηχάνημα. Ενημερώστε / etc / hosts και στα δύο μηχανήματα με τις ακόλουθες γραμμές:

```
# / Etc / hosts (για master και slave)
192.168.0.1 master
192.168.0.2 slave
```

SSH πρόσβαση

Ο hduser χρήστης στο master (γνωστός και ως @ hduser master) πρέπει να μπορούν να συνδέονται

α) στο λογαριασμό χρήστη του σχετικά με το master - δηλαδή ssh master και στο πλαίσιο αυτό δεν είναι απαραίτητα ssh localhost - και

β) με τον hduser λογαριασμό χρήστη στον SLAVE (γνωστός και ως @ hduser slave) μέσω ενός κωδικού-λιγότερο SSH login.

Πρέπει να προσθέσετε το @ hduser master 's SSH δημόσιο κλειδί (που θα πρέπει να είναι σε $ HOME /. ssh / id_rsa.pub στο authorized_keys αρχείο του hduser @ slave (σε αυτή την χρήστη $ HOME /. ssh / authorized_keys). Μπορείτε να το κάνετε αυτό με το χέρι ή να χρησιμοποιήσετε την **ακόλουθη εντολή SSH** :

```
hduser@master:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@slave
```

Αυτή η εντολή θα σας ζητήσει τον κωδικό σύνδεσης για το χρήστη hduser σε slave , στη συνέχεια αντιγράψετε το δημόσιο κλειδί SSH για σας, δημιουργώντας το σωστό κατάλογο και καθορίζει τα δικαιώματα ανάλογα με τις ανάγκες.

Το τελικό βήμα είναι να δοκιμαστεί η SSH εγκατάσταση με τη σύνδεση με το χρήστη hduser από το master στο λογαριασμό χρήστη hduser στο slave . Το βήμα είναι επίσης απαραίτητο για την αποθήκευση slave‘s host key fingerprint στο hduser@master‘sknown_hosts αρχείο.

Έτσι, συνδέει από master σε master

```
hduser@master:~$ssh master
The authenticity of host 'master (192.168.0.1)' can't be established.
RSA key fingerprint is
3b:21:b3:c0:21:5c:7c:54:2f:1e:2d:96:79:eb:7f:95.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master' (RSA) to the list of known hosts.
Linux master 2.6.20-16-386 #2 Thu Jun 7 20:16:13 UTC 2007 i686
...
hduser@master:~$
```

... Και από το master σε slave .

```
hduser@master:~$ssh  slave
The authenticity of host 'slave (192.168.0.2)' can't be established.
RSA key fingerprint is
74:d7:61:86:db:86:8f:31:90:9c:68:b0:13:88:52:72.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave' (RSA) to the list of known hosts.
Ubuntu 10.04
...
hduser@slave:~$
```

Στον φάκελο hadoop/conf του master αλλάζουμε το αρχείο slaves και προσθέτουμε τα εξής master, slave και λοιπά ονόματα υπολογιστών που τρέχουν hadoop και θέλουμε να προσθέσουμε στο cluster μας. Ακόμη, θα αλλάξουμε τα core-site.xml όπου αντί για localhost πλέον θα γράψουμε master <value>hdfs://master:54310</value> και στο hdfs-site.xml όπου αλλάζουμε το replication ανάλογα με το πόσα μηχανήματα έχουμε στο cluster. Για 3 μηχανήματα το κάνουμ 3. Ακολουθήσαμε την παραπάνω διαδικασία για στο instance του ec2, σε προσωπικό υπολογιστή και στο cluster του πανεπιστημίου και καταφέραμε να τα συνδέσουμε στο HDFS.

Για να χρησιμοποιήσουμε το HDFS μέσα από Java προγράμματα, χρειάζεται να βάλουμε τις ατνίστοιχες βιβλιοθήκες στο NetBeans. Ανοίγουμε το NetBeans, πηγαίνουμε Tools > Plugins. Στο Update Center πάμε στα Settings και πατάμε Add. Βάζουμε τα ακόλουθα Name και URL.

Name: Karmasphere Studio for Hadoop

URL: http://hadoopstudio.org/updates/updates.xml

Στην συνέχεια πηγαίνουμε στο Available Plugins, βρίσκουμε το "**Karmasphere Studio for Hadoop**" και το τσεκάρουμε. Πατάμε το install, Next και accept licence agreement. Πατάμε install , continue και μόλις τελειώσει κάνουμε ένα restart το IDE.

Για να το χρησιμοποιήσουμε μέσα σε κώδικα όπως στο service μας για να διαβάζει τα αρχεία που περιέχουν τον πίνακα Α για παράδειγμα, κάνουμε τα εξής:

- Στο project κάνουμε add new Library (Properties > Library > Add library) βρίσκουμε το Hadoop από την λίστα και το επιλέγουμε.

- Import τις βιβλιοθήκες που θέλουμε. Εμείς χρειαζόμαστε τα εξής :

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.Path;
```

- Χρησιμοποιούμε τις αντίστοιχες εντολές σρησιμοποιώντας το configuration του Hadoop. Ένα παράδειγμα χρήσης είναι το εξής[11] :

```
import org.apache.hadoop.fs.Path;
public class HDFSHelloWorld {
public static final String theFilename = "hello.txt";
public static final String message = "Hello, world!\n";
public static void main (String [] args) throws IOException {
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(conf);
   conf.addResource(new Path("/usr/local/hadoop/conf/hdfs-site.xml"));
   conf.addResource(new Path("/usr/local/hadoop/conf/core-site.xml"));
   Path filenamePath = new Path(theFilename);
          try {
             if (fs.exists(filenamePath)) {
                // remove the file first
       fs.delete(filenamePath);
        }
     FSDataOutputStream out = fs.create(filenamePath);
    out.writeUTF(message;
     out.close();
     FSDataInputStream in = fs.open(filenamePath);
    String messageIn = in.readUTF();
```

[11] http://developer.yahoo.com/hadoop/tutorial/module2.html#programmatically

```
    System.out.print(messageIn);
     in.close();
    } catch (IOException ioe) {
     System.err.println("IOException during operation: " + ioe.toString());
     System.exit(1);
    }
   }
}
```

Με τις γραμμές

```
import org.apache.hadoop.fs.Path;
            conf.addResource(new Path("/usr/local/hadoop/conf/hdfs-site.xml"));
          conf.addResource(new Path("/usr/local/hadoop/conf/core-site.xml"));
```

γίνεται η ανάγνωση των αρχείων hdfs-site και core-site ώστε να βρεί το configuration του Hadoop και να αποθηκεύει ή διαβάζει τα αρχεία στο HDFS.

5.5 Εκτέλεση πειραμάτων

Για να καταλήξουμε αν τελικά αξίζει να χρησιμοποιήσει κάποιος τις παραπάνω τεχνολογίες όπως το HDFS σε συνδιασμό με υπολογιστική δύναμη αρκετά μεγάλη σκαι συνεχώς παρεχόμενη, για να τρέχουν εκεί οι lapack services έπρεπε να εκτελεστούν κάποια τεστς. Επειδή το micro instance που διαλέξαμε λόγω ότι παρέχεται δωρεάν, δεν επαρκούσε για μεγάλα δεδομένα, εκτελέσαμε κάποια από τα τεστ με τις υπηρεσίες να τρέχουν στον application server του πανεπιστημίου. Για τα τεστ επιλέξαμε την συνάρτηση Dgesv, η οποία λύνει σύστημα από γραμμικές εξισώσεις της μορφής A*X=B, όπου ο A είναι ένας τετραγωνικός πίνακας n*n, οι στήλες του B είναι η κάθε μία και ένα δεξί μέλος και οι στήλες του X είναι οι αντίστοιχες λύσεις.

Από την συνάρτηση χρησιμοποιείται LU παραγοντοποίηση , με μερική οδήγηση και εναλλαγές γραμμών, για τον πίνακα A ώστε να προκύψει A=P*L*U. Ο P είναι ο πίνακας μετάθεσης, ο L είναι μοναδιαίος κάτω τριγωνικός και ο U είναι άνω τριγωνικός. Η νέα μορφή του πίνακα A χρησιμοποιείται για να λυθεί το σύστημα A*X=B.

Στα τεστ που τρέξαμε θέλαμε να μελετήσουμε:

1. Τον χρόνο που χρειάζεται το service για να διαβάσει τα δεδομένα. Τα δεδομένα στην πρώτη περίπτωση είναι αποθηκευμένα σε HDFS, στην δεύτερη σε απλό

filesystem του μηχανήματος που τρέχει το service και στην τρίτη περίπτωση στέλνουμε τα δεδομένα από τον client στο service.

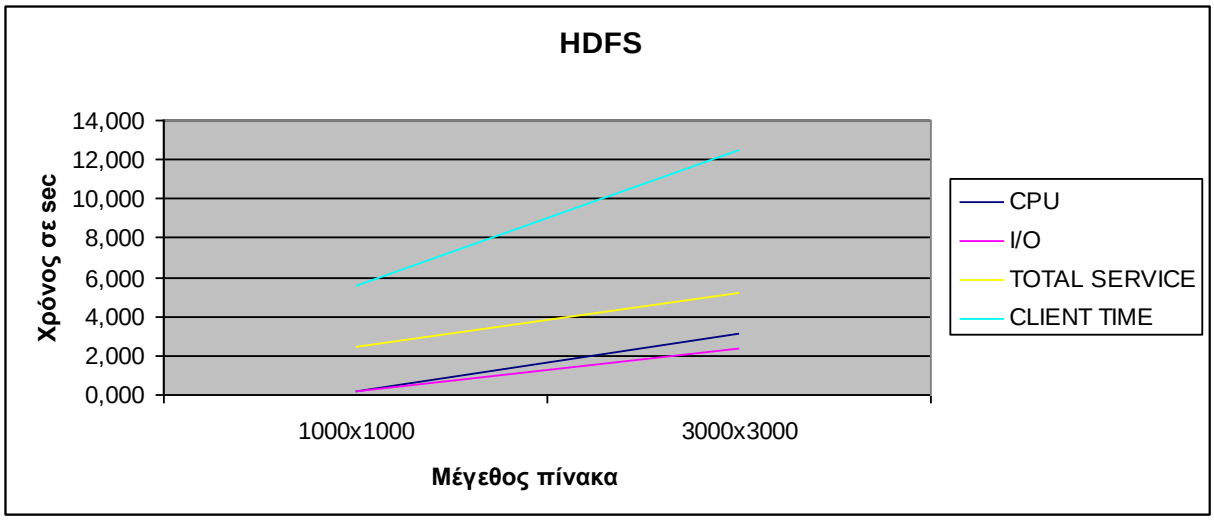
2. Τον cpu time που χρειάζεται το service για να τρέξει την lapack routine.
3. Τον ολικό χρόνο που καταναλώνει το service από την στιγμή που το καλέσουμε, μέχρι την στιγμή που επιστρέφει τα αποτελέσματα.
4. Τον ολικό χρόνο που τρέχει ο client.

Όπως βλέπουμε στον χρόνο που κάνει η υπηρεσία να διαβάσει τα δεδομένα, διακρίνουμε τρείς περιπτώσεις. Έτσι εκτελέσαμε τα ατνίστοιχα τρία τεστς στο καθένα από τα οποία μελετήσαμε τους παραπάνω χρόνους.

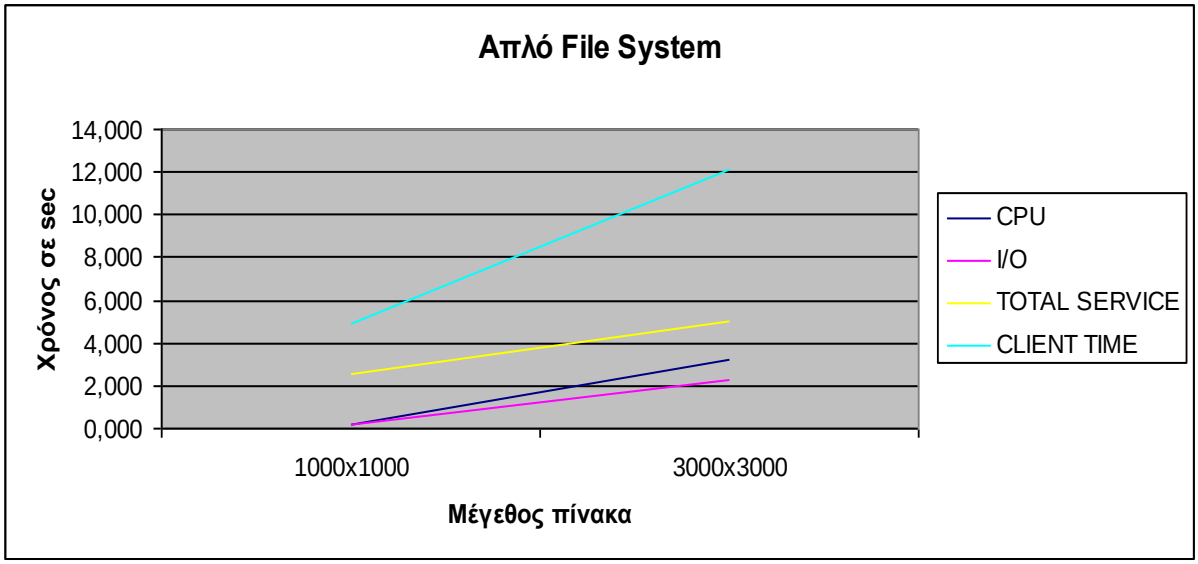
Στο ec2 καταφέραμε να τρέξουμε με σχετικά πολύ μικρούς πίνακες λόγω των χαρακτηριστικών του (613 MB memory, I/O Performance: Low) [12]. Έτσι καταφέραμε να δημιουργήσουμε πίνακες μέχρι 3.00x3.000 στοιχεία. Γι αυτόν τον λόγο, τρέξαμε δύο τεστ, ένα για πίνακα Α 1000x1000 και ένα για πίνακα Α 3000x3000. Ταποτελέσματα διακρίνται στον παρακάτω πίνακα και διάγραμμα.

HDFS				
	ΧΡΟΝΟΣ ΣΕ SECOND			
ΜΕΓΕΘΟΣ ΠΙΝΑΚΑ	**CPU**	**I/O**	**TOTAL SERVICE**	**CLIENT TIME**
1000x1000	0,200	0,210	2,500	5,580
3000x3000	3,100	2,380	5,200	12,500

[12] http://aws.amazon.com/ec2/instance-types/

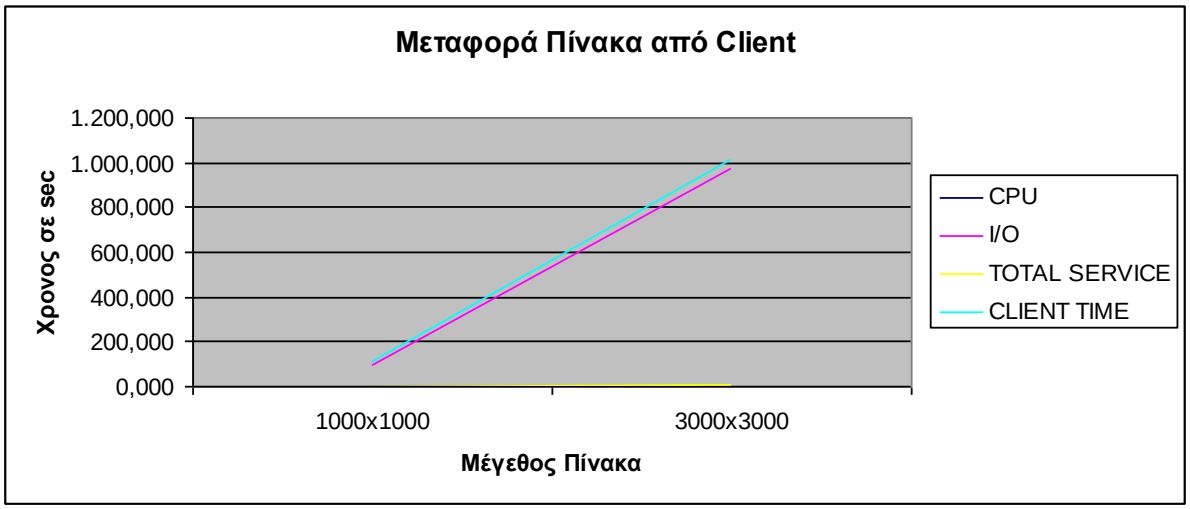


Απλό File System				
	ΧΡΟΝΟΣ ΣΕ SECOND			
ΜΕΓΕΘΟΣ ΠΙΝΑΚΑ	CPU	I/O	TOTAL SERVICE	CLIENT TIME
1000x1000	0,190	0,190	2,600	4,900
3000x3000	3,200	2,240	5,010	12,100



Μεταφορά Πίνακα από Client				
	ΧΡΟΝΟΣ ΣΕ SECOND			
ΜΕΓΕΘΟΣ ΠΙΝΑΚΑ	CPU	I/O	TOTAL SERVICE	CLIENT TIME

1000x1000	0,190	98,200	2,500	110,340
3000x3000	3,500	974,580	5,600	1.014,720

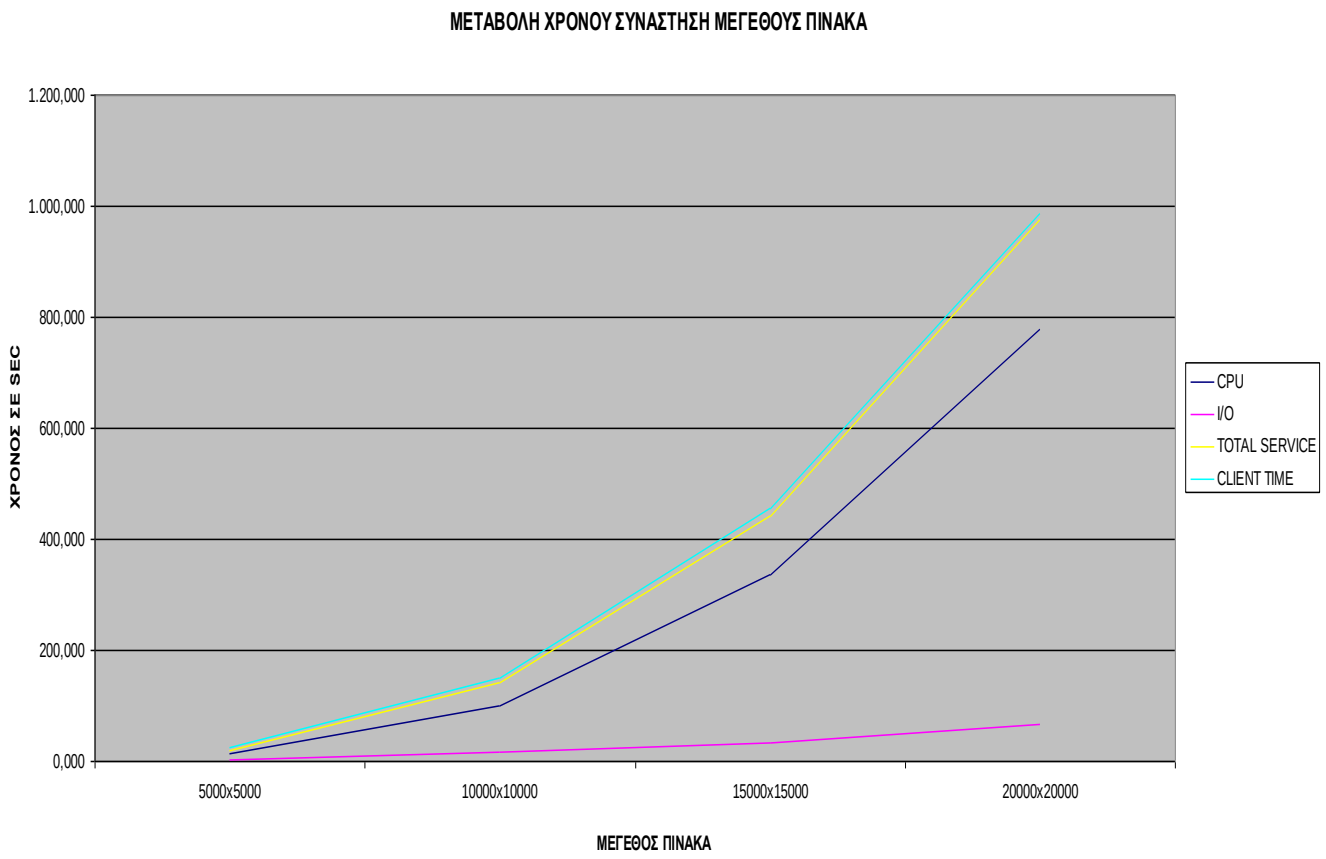


Λόγω τον μικρών δεδομένων δεν μπορούμε να βγάλουμε ασφαλή συμπεράσματα αλλά θα μπορούσαμε να πούμε ότι η υπηρεσία χρειάζεται λίγο λιγότερο χρόνο για να διαβάσει τα δεδομένα από το απλό σύστημα αρχείων σε σχέση με το HDFS. Όταν τα πέρνει από τον client τότε η διαφορά είναι εμφανέστατη. Ο χρόνος για να διαβάσει τα δεδομένα και ο χρόνος του client είναι πολύ μεγαλύτεροι.

Λόγω των μικρών δεδομένων που μπορούσαμε να δειαχειρηστούμε στο ec2 micro instance, μεταφερθήκαμε στο cluster του πανεπιστημίου το όποίο έχοντας 20 GigaByte RAM μπορούσε να διαχειριστεί πολύ μεγαλύτερους πίνακες. Στο πρώτο από τα τεστ, στήθηκε στο cluster του πανεπιστημίου το HDFS (Hadoop Distributed File System). Αποθηκεύτηκαν στο συγκεκριμένο σύστημα αρχείων οι πίνακες A και τα δεξιά μέλη B. Συγκεκριμένα χρησιμοποιήθηκαν πίνακες A μεγέθους 5.000x5.000, 10.000x10.000, 15.000x15.000 και 20.000x20.000 και τα αντίστοιχου μεγέθους δεξιά μέλη. Στον glassfish application server σηκώσαμε ένα web service το οποίο καλεί την native ρουτίνα της DGESV. Σαν παραμέτρους παίρνει το μέγεθος n του τετραγωνικού A το πλήθος των δεξιών μελών και τα δύο ονόματα αρχείων για τους πίνακες A και B. Πέρα από το την λύση, το service υπολογίζει και τον χρόνο που χρειάζεται για να διαβάσει τους πίνακες από το HDFS, τον cpu time που χρειάζεται για τον υπολογισμό της λύσης και τον ολικό χρόνο που χρειάζεται το service από την στιγμή που το καλέσει ο client μέχρι την στιγμή που θα επιστρέψει. Καλούμε το service από τον client που έχουμε στήσει σε δικό μας υπολογιστή και το καλούμε δίνοντας του τα

ονόματα των πινάκων αλλά και το μέγεθος του προβλήματος. Τα αποτελέσματα σε χρόνο για κάθε εκτέλεση, αλλά και το πόσο αλλάζουν οι απαιτήσεις σε χρόνο ανάλογα με το μέγεθος του προβλήματος φαίνονται στους παρακάτω πίνακες.

1ο TEST : Data σε HDFS				
	ΧΡΟΝΟΣ ΣΕ SECOND			
ΜΕΓΕΘΟΣ ΠΙΝΑΚΑ	**CPU**	**I/O**	**TOTAL SERVICE**	**CLIENT TIME**
5000x5000	13,580	3,560	20,380	24,291
10000x10000	101,320	17,500	141,000	150,120
15000x15000	336,250	33,960	441,990	455.593,000
20000x20000	778,300	66,910	977,780	990,100

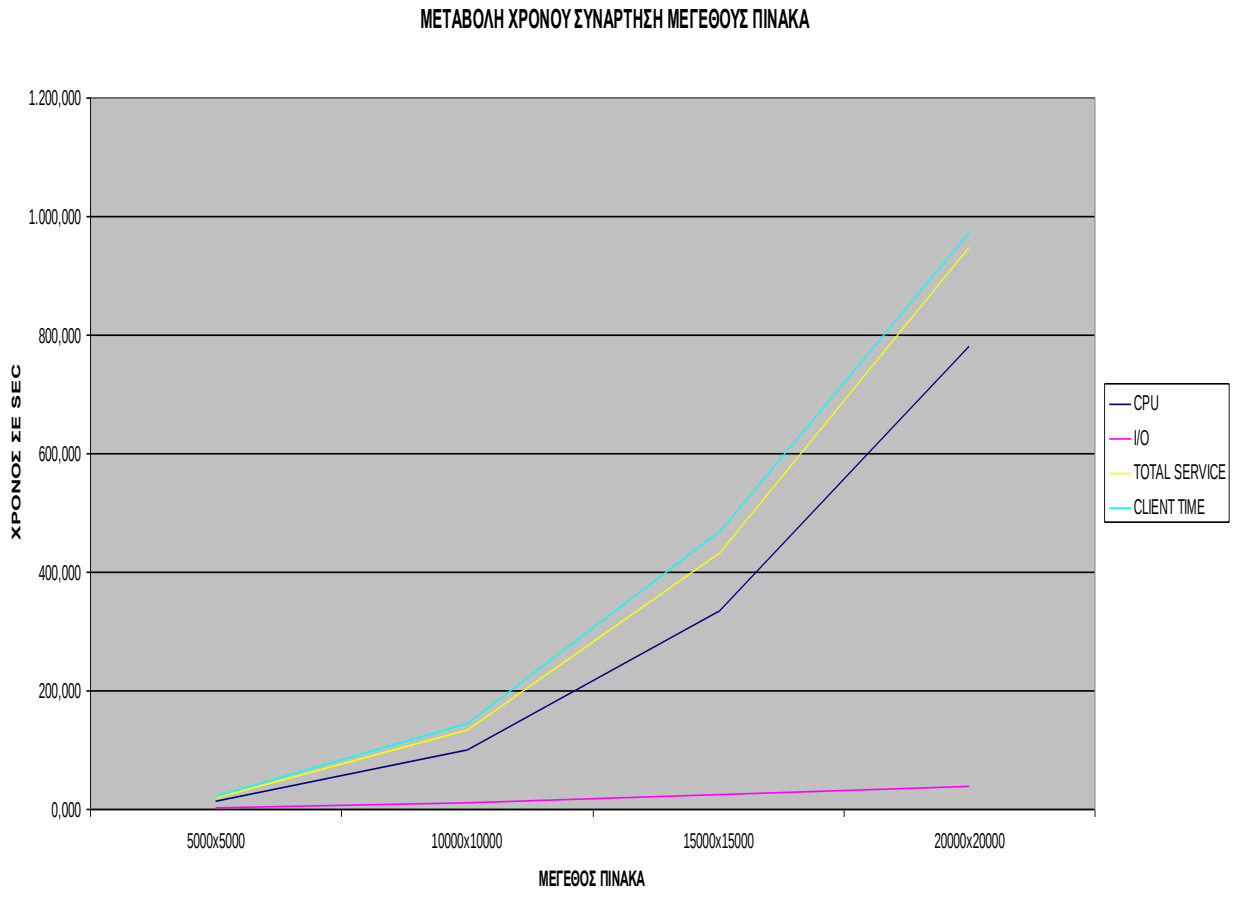


Παρατηρούμε ότι ενώ ο χρόνος για την ανάγνωση των πινάκων αυξάνεται σχετικά ομαλά όσο αυξάνεται το πλήθος των στοιχείων, οι υπόλοιποι χρόνοι, αυξάνονται πιο απότομα.

Μπορούμε εύκολα να συμπαιράνουμε πως περίπου θα αυξηθούν οι χρόνοι για πιο μεγάλα δεδομένα.

Στο δεύτερο τεστ, ακολουθούμε τν ίδια γραμμή με το πρώτο, μόνο που οι πίνακες δεν έχουν αποθηκευτεί στο HDFS αλλά στο σύστημα αρχείων που χρησιμοποίει το cluster. Τα αντίστοιχα αποτελέσματα φαίνονται παρακάτω.

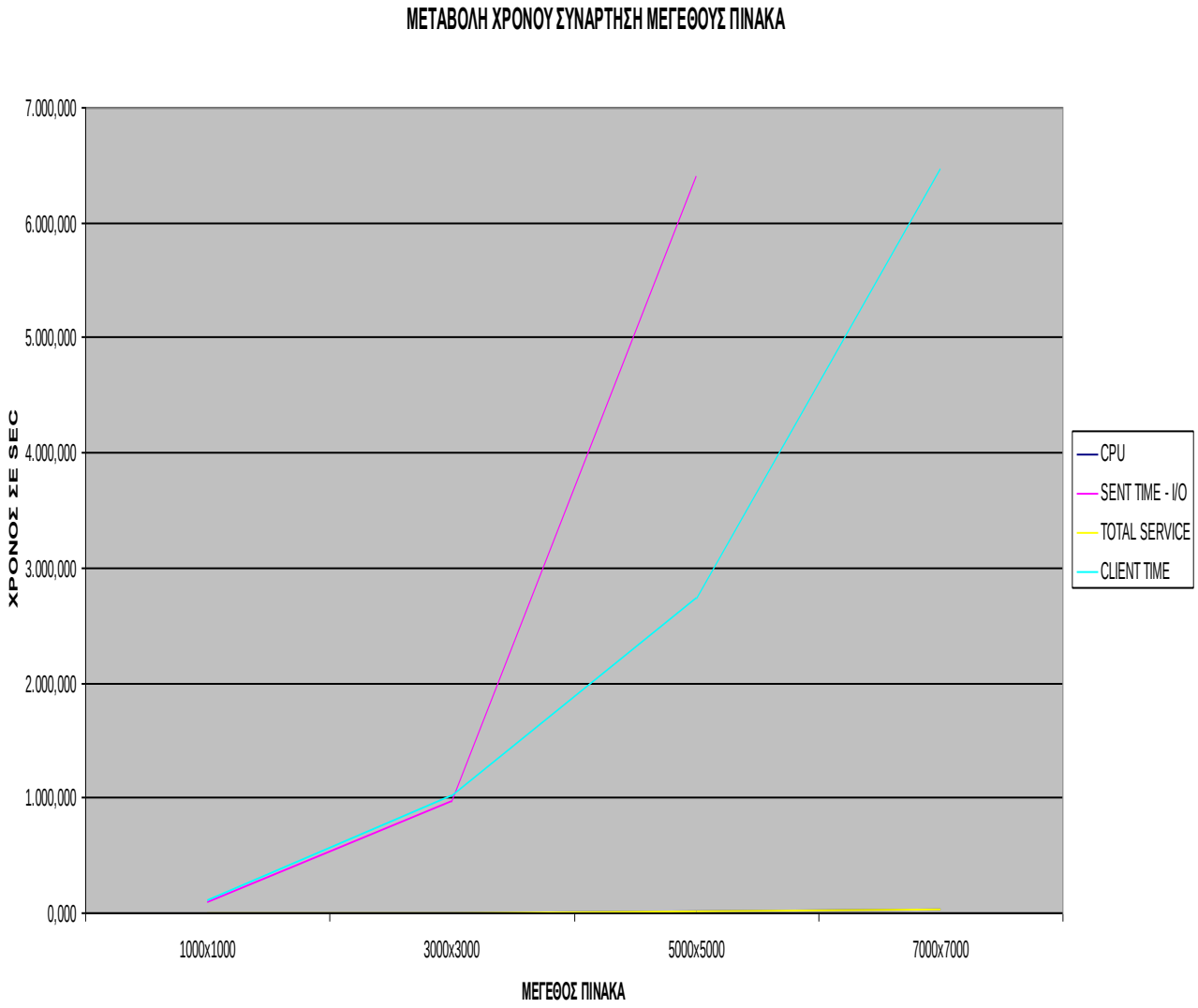
2ο TEST:Απλό File System				
	ΧΡΟΝΟΣ ΣΕ SECOND			
ΜΕΓΕΘΟΣ ΠΙΝΑΚΑ	**CPU**	**I/O**	**TOTAL SERVICE**	**CLIENT TIME**
5000x5000	15,000	2,360	18,080	22,200
10000x10000	98,740	11,510	134,130	143,454
15000x15000	334,380	24,030	430,060	466.049,000
20000x20000	781,530	39,090	951,040	972,100



Ανάλογα με το πρώτο τεστ είναι τα συμπεράσματα μας και εδώ. Οι χρόνοι εκτός του χρόνου για την ανάγνωση των δεδομένων αυξάνονται όλο και πιο απότομα ανάλογα με την αύξηση του μεγέθους των data.

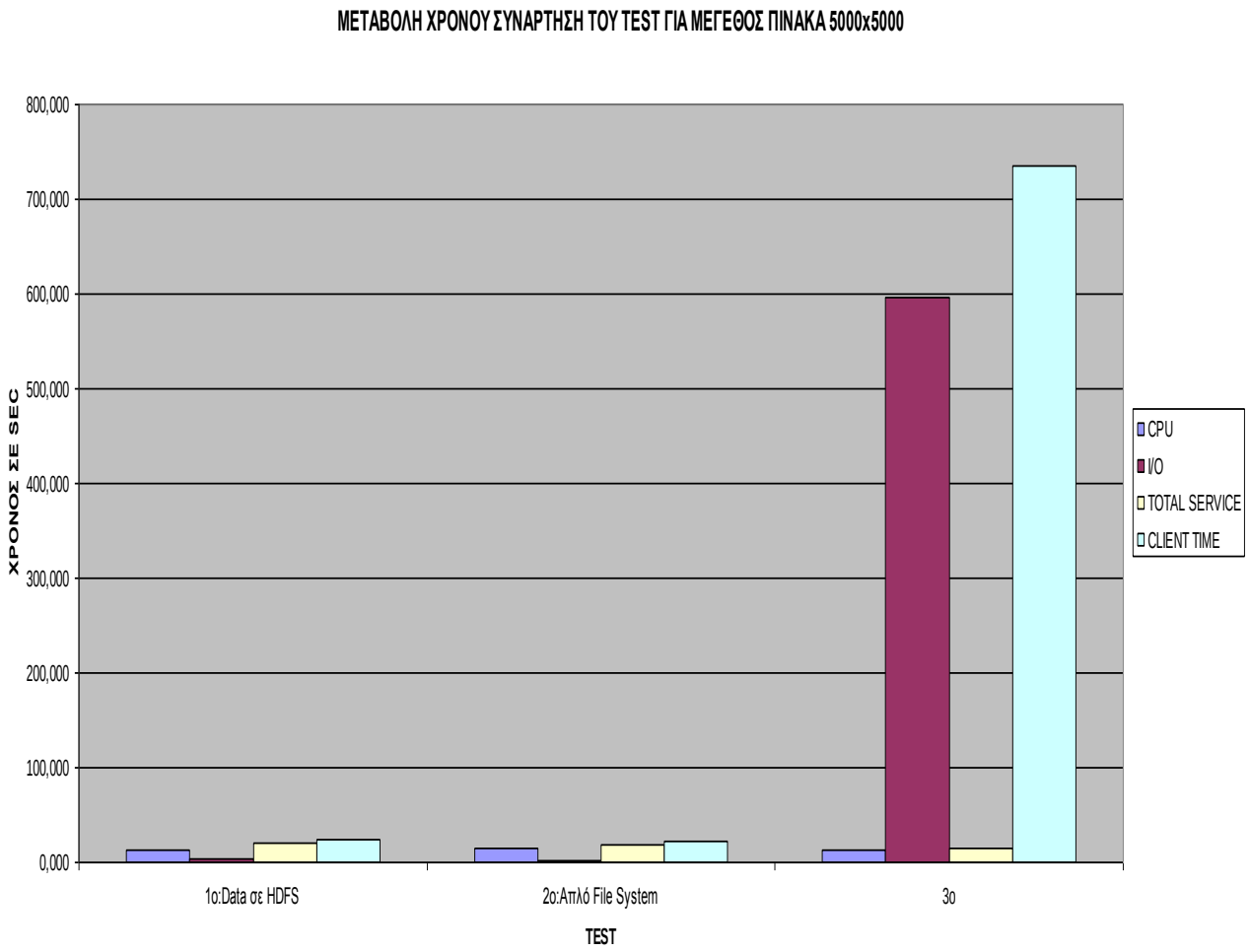
Στο τρίτο τεστ οι πίνακες υπάρχουν στην μεριά του χρήστη και θα σταλούν στο service πάνω από το δίκτυο. Αυτό έχει πολύ μεγάλο κόστος σε χρόνο, πέραν του ότι για να δημιουργήσουμε και μόνο τα δεδομένα πρέπει να έχουμε και την αντίστοιχη RAM. Μάλιστα, σε υπολογιστή με σχεδόν 2 GB RAM, δημιουργήσαμε μέχρι πίνακα Α μεγέθους 7.000x7.000. Οπότε τα τεστ που εκτελέσαμε ήταν για 1.000x1.000, 3.000x3.000 5.000x5.000 και 7.000x7.000.

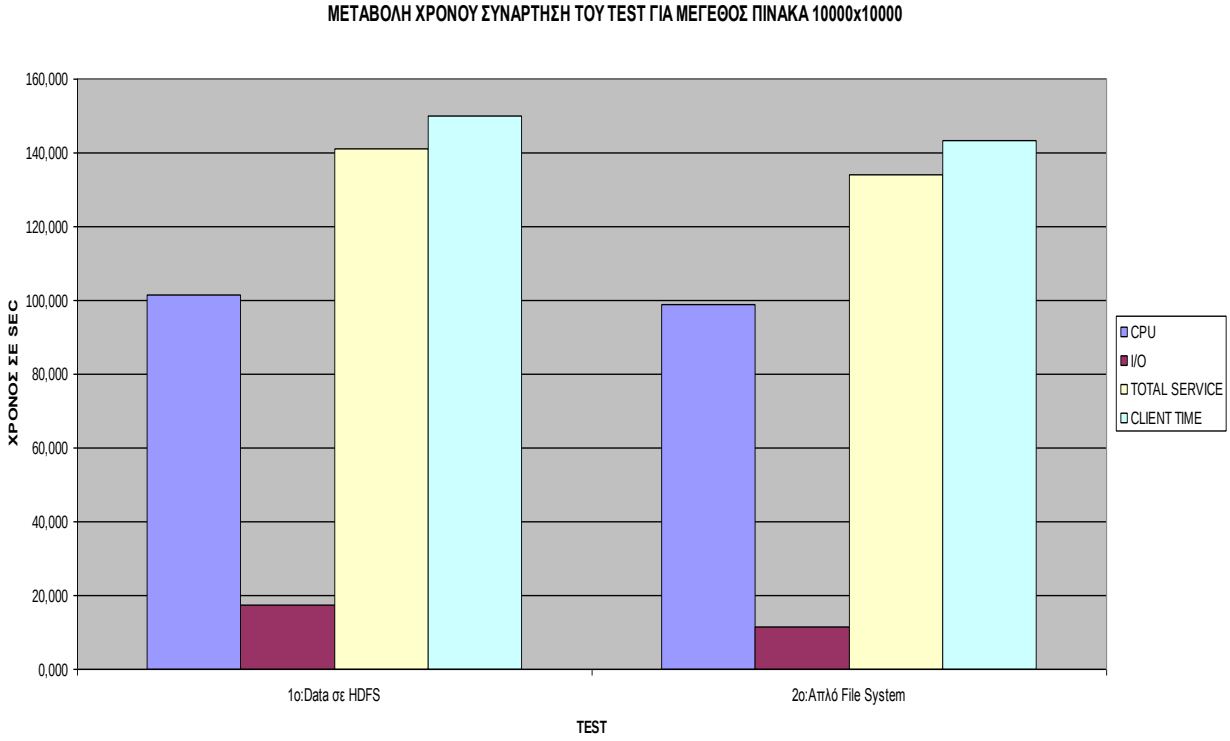
3ο TEST				
	ΧΡΟΝΟΣ ΣΕ SECOND			
ΜΕΓΕΘΟΣ ΠΙΝΑΚΑ	**CPU**	**SENT TIME - I/O**	**TOTAL SERVICE**	**CLIENT TIME**
1000x1000	0,140	97,240	0,190	112,345
3000x3000	3,200	975,444	3,611	1.013,732
5000x5000	13,820	2.595,380	15,060	2.735,238
7000x7000	36,960	6.423,480	39,430	6.474,475

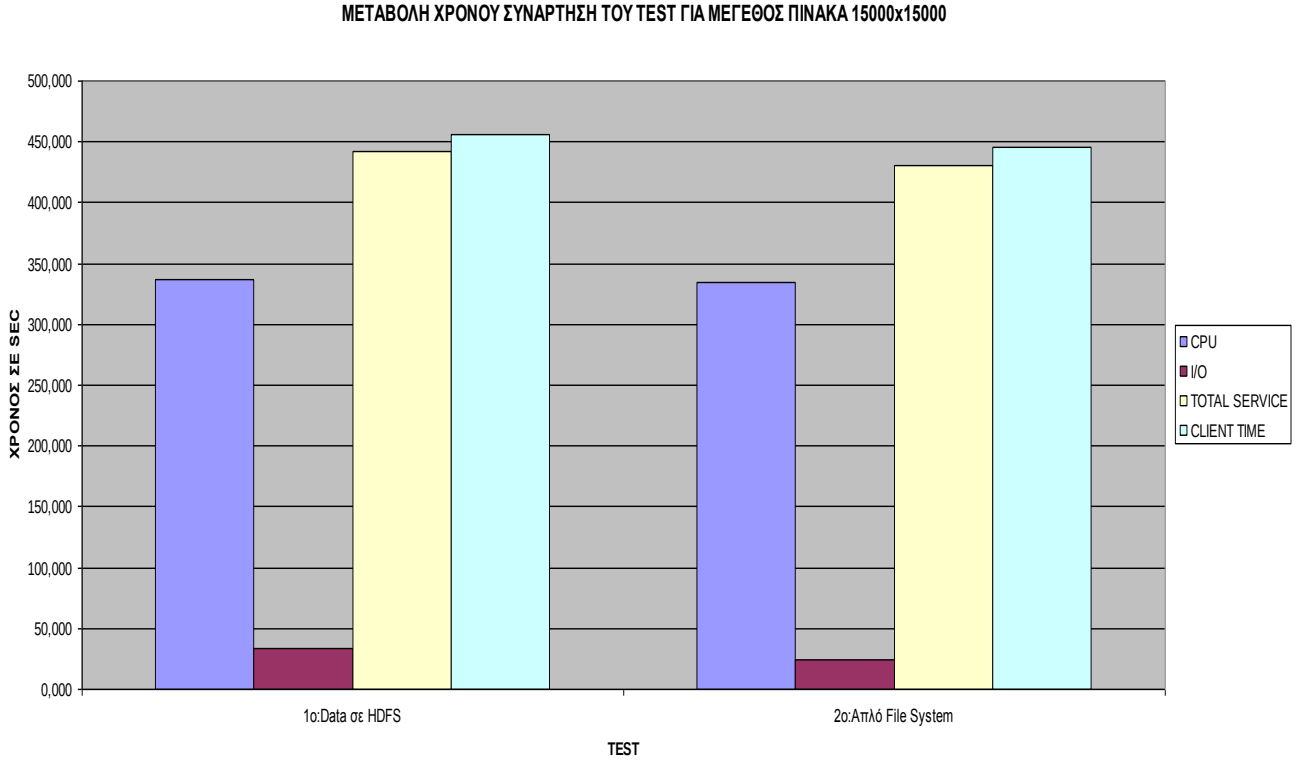


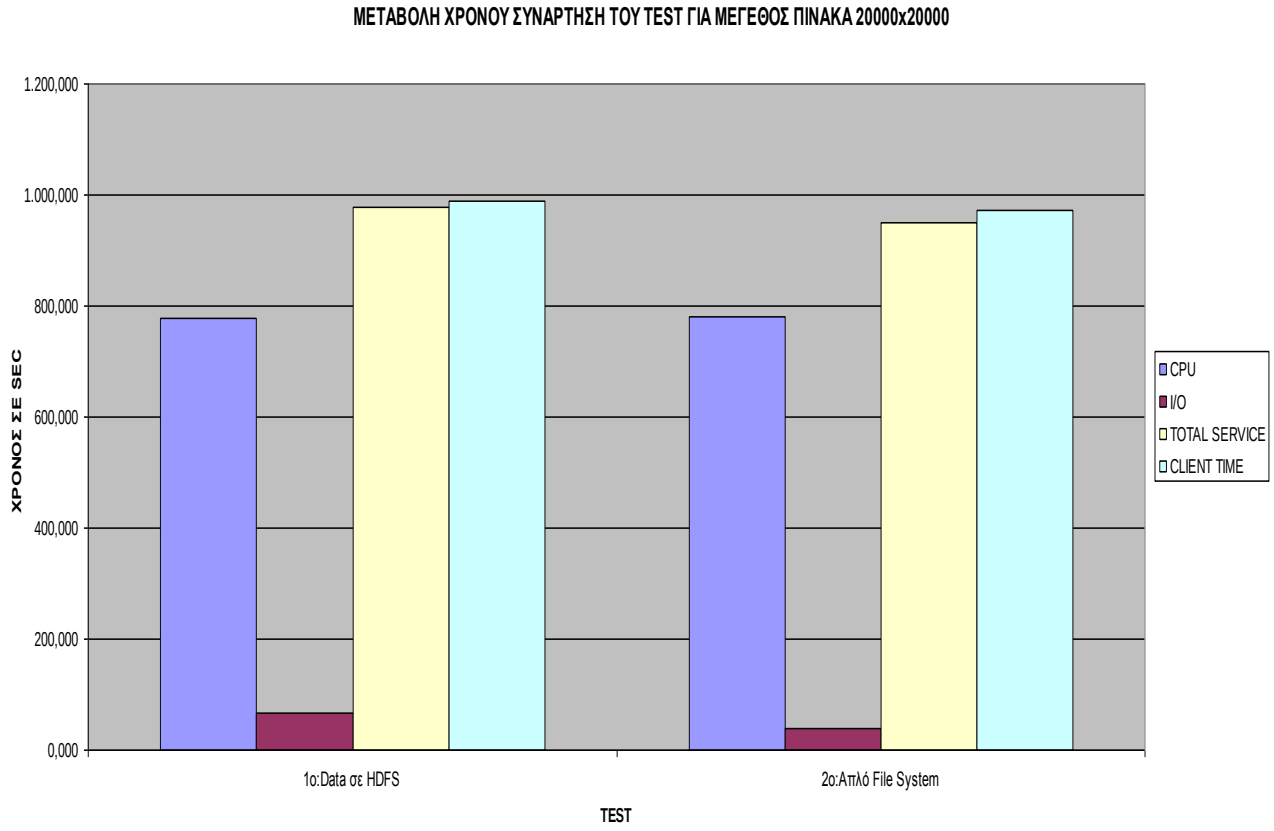
Εδώ, κυριαρχούν οι χρόνοι για την αποστολή των δεδομένων από τον client στο service και ο χρόνος που καταναλώνει ο πελάτης για να τερματίσει. Οι δύο αυτοί χρόνοι αυξάνονται απότομα για πιο μεγάλα δεδομένα ενώ ο cpu time και ο total service time είναι πολύ μικροί σε σχέση με τους άλλους και έτσι φαίνονται να είναι πολύ κοντά στο μηδέν. Παρατηρούμε ότι ενώ οι χρόνοι που αφορούν τον καθ' εαυτό υπολογισμό των αποτελεσμάτων και η επιστροφή τους είναι μικροί, πολύ κοντά στους αντίστοιχους χρόνους των προηγουμένων χρόνων, μεγάλο χρονικό διάστημα καταναλώνεται στο να στείλουμε τα δεδομένα και στις περισσότερες περιπτώσεις αυτό είναι σοβαρό μειωνέκτημα, όπως και το ότι οι πίνακες δημιουργούνται στην πλευρά του χρήστη και ίσως να μην μπορεί να δημιουργήσει τα data που θέλει λόγω περιορισμένης μνήμης.

Στα παρακάτω γραφήματα φαίνεται η σχέση του μεγέθους του πίνακα με το χρόνο και πώς τον επηρεάζει σε κάθε Test.




ΜΕΤΑΒΟΛΗ ΧΡΟΝΟΥ ΣΥΝΑΡΤΗΣΗ ΤΟΥ TEST ΓΙΑ ΜΕΓΕΘΟΣ ΠΙΝΑΚΑ 10000x10000
160,000
140,000
120,000
100,000
80,000
60,000
40,000
20,000
0,000
ΧΡΟΝΟΣ ΣΕ SEC
1ο:Data σε HDFS
2ο:Απλό File System
TEST
CPU
I/O
TOTAL SERVICE
CLIENT TIME


ΜΕΤΑΒΟΛΗ ΧΡΟΝΟΥ ΣΥΝΑΡΤΗΣΗ ΤΟΥ TEST ΓΙΑ ΜΕΓΕΘΟΣ ΠΙΝΑΚΑ 15000x15000
500,000
450,000
400,000
350,000
300,000
250,000
200,000
150,000
100,000
50,000
0,000
ΧΡΟΝΟΣ ΣΕ SEC
1ο:Data σε HDFS
2ο:Απλό File System
TEST
CPU
I/O
TOTAL SERVICE
CLIENT TIME



Τελικά από τα παραπάνω ιστογράμματα φαίνεται ότι το να χρησιμοποιήσουμε το HDFS σαν filesystem μας κοστίζει λίγο σε χρόνο σε σχέση με το κανονικό filesystem. Η χρονική διαφορά όμως είναι αρκετά μικρή και παρατηρούμε ότι όσο αυξάνονται τα δεδομένα σε μέγεθος, η χρονική διαφορά δεν αυξάνεται.Είναι μεν λίγο πιο αργό, αλλά για παράδειγμα για πίνακα A 20000x20000 στοιχεία η τελική διαφορά που παρατηρεί ο χρήστης είναι γύρω στα 15-20 δευτερόλεπτα σε χρονικό ορίζοντα των 1000 δευτερολέπτων περίπου. Ίσως δηλαδή όσο αυξάνονται τα δεδομένα, ο χρήστης να μην παρατηρεί καν ότι είναι πιο αργό. Ακόμη συνυπολογίζοντας και τα προτερήματα της χρήσης του HDFS,όπως του ότι μπορεί ο χρήστης να έχει ένα cluster υπολογιστών συνδεδεμένων με το σύστημα αρχείων και να κρατάει σε κάθε έναν από αυτούς αντίγραφα των δεδομένων του σε blocks των 64MB και η σύνδεσή του με το MapReduce του Hadoop, τότε ίσως να τον συμφέρει να το χρησιμοποιήσει σαν το συστημα αρχείων του για να καλεί Lapack web services.

6ο Κεφάλαιο

Σύνοψη και προοπτικές

Στην παρούσα μελέτη ασχοληθήκαμε με την υλοποίηση Lapack web services Lapack web services και την διαχείρηση των δεδομένων που αυτές χρησιμοποιούν. Υλοποιήθηκαν κάποιες Lapack routines και τοποθετήθηκαν τόσο σε micro instance του Amazon EC2, όσο και στο cluster του πανεπιστημίου. Μελετήθηκε το γενικό σχέδιο του Hadoop και ειδικά το HDFS. Στήθηκε ένα HDFS cluster και είδαμε πώς μπορούμε να το χρησιμοποιήσουμε. Εκτελέστηκαν κάποια τεστς και στο ec2 αλλά και στο cluster και βγάλαμε κάποια συμπεράσματα σχετικά με την χρήση του HDFS ώς ατικαταστάτη του κανονικού συστήματος αρχείων. Είδαμε τελικά ότι για μεγάλα δεδομένα δεν μας κοστίζει αρκετά επιπλέον σε χρόνο, αντίθετα και λόγω των θετικών στοιχείων που έχει, μας βολεύει η χρήση του.

Σχετικά με τς προοπτικές που ανοίγονται σε σχέση με την χρήση των Lapack web services πολλά θα μπορούσαν να αναφερθούν. Θα περιοριστούμε να αναφέρουμε κάποιους τομείς που χρήζουν βελτίωσης:

1. Υλοποίηση περισσοτέρων ρουτινών της Lapack ως υπηρεσίες.
2. Ανάπτυξη των παραπάνω ρουτινών σε κάποιον server όπως το cluster του πανεπιστημίου ή σε κάποιο instance του EC2 που όμως να μπορεί να διαχειριστεί μεγάλα σετ δεδομένων.
3. Στήσιμο HDFS cluster με συνδεση διαφόρων υπολογιστών ώστε να αποθηκεύονται εκεί τα δεδομένα.
4. Χρήση MapReduce[13] το οποίο είναι ένα πλαίσιο λογισμικού για την εύκολη συγγραφή εφαρμογών οι οποίες διαχειρίζονται μεγάλα σετ δεδομένων παράλληλα. Θα πρέπει να διερευνηθεί το πώς μπορεί να χρησιμοποιηθεί ώστε να μειώσουμε τον χρόνο που διαχειριζόμαστε να μεγάλα δεδομένα που χρησιμοποιούμε.

[13] http://hadoop.apache.org/common/docs/current/mapred_tutorial.html

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. **Papazoglou, M.** Web Services : Principles and technology.
2. **Κατσογιάννου, Γ.** Υψηλής Απόδοσης Υπηρεσίες διαδυκτύου για Επιστημονικούς Υπολογισμούς.
3. **Μπένης Δημήτρης,** Developers Guide: How to expose a Lapack Method as a Web Service.
4. **Σαλούστρου, Μ.** Επιστημονικοί Υπολογισμοί στον Παγκόσμιο Ιστό.
5. **Satish Narayana Srirama, Pelle Jakovits, Eero Vainikko,** Adapting scientific computing problems to clouds using MapReduce, Future Generation Computer Systems,12 June 2011.