GRID COMPUTING

ΥΠΟΛΟΓΙΣΤΙΚΑ ΠΛΕΓΜΑΤΑ ΚΑΙ ΕΦΑΡΜΟΓΕΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΠΑΠΑΔΑΤΟΣ ΑΛΕΞΑΝΔΡΟΣ

Μάρτιος 2011

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω όλους εκείνους που συνέβαλλαν και βοήθησαν στην πραγματοποίηση αυτής της διπλωματικής εργασίας.

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα Καθηγητή, Διδάκτορα κ. Ιωάννη Μπαρμπουνάκη για την πολύτιμη βοήθεια και καθοδήγησή του.

Θα ήθελα επίσης να ευχαριστήσω την ομάδα υποστήριξης χρηστών του HellasGrid και συγκεκριμένα τον κ. Βασίλειο Γκάμα καθώς επίσης και τον κ. Πασχάλη Κορόσογλου από την ομάδα υποστήριξης του Κέντρου Επιστημονικών Υπολογιστικών Υπηρεσιών του Α.Π.Θ. για την αποτελεσματική βοήθειά τους, κατά τη διάρκεια της εκπόνησης της διπλωματικής μου εργασίας.

Τέλος αισθάνομαι την ανάγκη να ευχαριστήσω την οικογένειά μου και τους φίλους μου για την υποστήριξη και την υπομονή τους.

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγή..... 10
   1.1. Το υπολογιστικό πλέγμα GRID..... 10
   1.2. Εξελισσόμενες τεχνολογίες GRID..... 12
   1.3. Από τους υπερυπολογιστές στο GRID..... 13
   1.4. Ευρωπαϊκά έργα με εφαρμογές GRID..... 14
   1.5. HellasGrid..... 15
   1.6. Διεθνείς Ερευνητικές Πρωτοβουλίες με Ελληνική Συμμετοχή..... 17

2. Πρόσβαση στην υποδομή του GRID..... 19
   2.1. Απόκτηση ψηφιακού πιστοποιητικού..... 19
   2.2. Εγκατάσταση πιστοποιητικού..... 22
   2.3. Εγγραφή σε εικονικό οργανισμό (Virtual Organization)..... 22
   2.4. User Interface..... 23
   2.5.Εξαγωγή του πιστοποιητικού..... 23
   2.6. Αντιγραφή του πιστοποιητικού στο UI..... 26
   2.7. Πρώτα βήματα και βασικές εντολές UNIX..... 28
   2.8. Proxy Certificate..... 30

3. FTP και GridFTP..... 33
   3.1. File Transfer Protocol (FTP)..... 33
      3.1.1. Τρόπος λειτουργίας..... 33
      3.1.2. Ενεργητική λειτουργία..... 34
      3.1.3. Παθητική λειτουργία..... 34
      3.1.4. Χρήση..... 35
      3.1.5. Ασφάλεια..... 35
      3.1.6. Προβλήματα NAT..... 36
      3.1.7. Έλλειψη κρυπτογράφησης..... 37
   3.2. Grid File Transfer Protocol (GridFTP)..... 39
      3.2.1. Ασφάλεια με GSI..... 40

3.2.2. Μεταφορές τρίτων προσώπων..... 41
3.2.3. Παράλληλη και εν μέρει μεταφορά..... 42
3.2.4. Μερική μεταφορά αρχείων..... 43
3.2.5. Ανοχή σφαλμάτων και επανεκκίνηση..... 44
3.2.6. Αυτόματη βελτιστοποίηση του TCP..... 44

4. Workload Management System (WMS)..... 46
4.1. WMProxy..... 46
4.1.1. Delegation..... 48
4.2. Workload Manager (WM)..... 49
4.3. Computer Element (CE) και Worker Node (WN)..... 50
4.4. Match Maker..... 50
4.5. Information Supermarket..... 50
4.6. Storage Element..... 51

5. Command Line Interface (CLI)..... 52
5.1. Ονομασίες αρχείων και ο εντοπισμός τους..... 52
5.2. Παράδειγμα μεταφοράς αρχείων..... 54

6. Job Description Language (JDL)..... 59
6.1. Attributes (Ιδιότητες)..... 60
6.2. Περιγραφή ιδιοτήτων των εργασιών..... 61
6.2.1. Type..... 61
6.2.2. JobType (Τύπος εργασίας)..... 62
6.2.3. Executable..... 64
6.2.4. Arguments..... 66
6.2.5. StdInput..... 68
6.2.6. StdOutput..... 69
6.2.7. StdError..... 70
6.2.8. InputSandbox..... 70
6.2.9. InputSandboxBaseURI..... 73
6.2.10. OutputSandbox..... 74

6.2.11. OutputSandboxDestURI..... 76
6.2.12. OutputSandboxBaseDestURI..... 77

7. Εργασίες..... 79
7.1. Παράδειγμα 1^{ο} : Normal Job 79
7.2. Παράδειγμα 2^{ο} : Normal Job με InputSandbox..... 84
7.3. Παράδειγμα 3^{ο} : Δημιουργία αρχείου στο UI..... 86
7.4. Παράδειγμα 4^{ο} : Collection Job..... 90
7.5. Παράδειγμα 5^{ο} : Parametric Job 1..... 93
7.6. Παράδειγμα 6^{ο} : Parametric Job 2..... 96
7.7. Παράδειγμα 7^{ο} : Interactive Job..... 99
7.8. Παράδειγμα 8^{ο} : MPICH Job..... 103

ΚΑΤΑΛΟΓΟΣ ΑΚΡΩΝΥΜΩΝ

CA : Certificate Authority

CE : Computer Element

CERN : Conseil Européen pour la Recherche Nucléaire

CLI : Command Line Interface

DAG : Directed Acyclic Graph

FTP : File Transfer Protocol

FXP : File Exchange Protocol

GSI : Grid Security Infrastructure

GUID : Grid Unique Identifier

HTTP : Hypertext Transfer Protocol

HTTPS : HTTP Secure

IP : Internet Protocol

IS : Information Service

JDL : Job Description Language

LCG : LHC Computing Grid

LFC : LCG File Catalog

LFN : Logical File Name

LHC : Large Hadron Collider

MIME : Multipurpose Internet Mail Extensions

MPI : Message Passing Interface

NAT : Network Address Translation

PBS : Portable Batch System

RA : Registration Authority

SCP : Secure Copy

SE : Storage Element

SEE : South Eastern Europe

SSH : Secure Shell

SSL : Secure Sockets Layer

SURL : Storage Uniform Resource Locator

TCP : Transmission Control Protocol

TURL : Transport Uniform Resource Locator

UI : User Interface

URI : Uniform Resource Identifier

VO : Virtual Organization

VOMS : Virtual Organization Membership Service

WM : Workload Manager

WMS : Workload Management System

WN : Worker Node

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1 : Το ψηφιακό πιστοποιητικό..... 19
Εικόνα 2 : Αίτημα ψηφιακού πιστοποιητικού στο HellasGrid..... 20
Εικόνα 3 : Φόρμα εγγραφής χρήστη στο HellasGrid..... 21
Εικόνα 4 : Η διαδικασία απόκτησης ψηφιακού πιστοποιητικού..... 21
Εικόνα 5 : Το πιστοποιητικό στον browser..... 24
Εικόνα 6 : Προβολή του πιστοποιητικού..... 25
Εικόνα 7 : Λεπτομέρειες πιστοποιητικού και εξαγωγή..... 25
Εικόνα 8 : Το πρόγραμμα puTTY..... 26
Εικόνα 9 : Εισαγωγή username και password στο UI..... 27
Εικόνα 10 Είσοδος στο πρόγραμμα WinSCP..... 27
Εικόνα 11 : Αντιγραφή του πιστοποιητικού στον UI λογαριασμό..... 28
Εικόνα 12 : Η εντολή cd..... 30
Εικόνα 13 Δημιουργία proxy πιστοποιητικού..... 31
Εικόνα 14 : Πληροφορίες για το proxy certificate..... 32
Εικόνα 15 : Ενεργητική λειτουργία σύνδεσης δεδομένων..... 34
Εικόνα 16 : Παθητική λειτουργία σύνδεσης δεδομένων..... 35
Εικόνα 17 : Network Address Translation (NAT)..... 36
Εικόνα 18 : Πρόβλημα NAT με firewall..... 37
Εικόνα 19 : Packet Sniffer..... 38
Εικόνα 20 : Secure Sockets Layer (SSL)..... 40
Εικόνα 21 : Secure SHell (SSH)..... 41
Εικόνα 21 : Εν μέρει μεταφορά δεδομένων με GridFTP..... 42
Εικόνα 22 : Παράλληλη μεταφορά δεδομένων με GridFTP..... 43
Εικόνα 23 : Μερική μεταφορά αρχείου..... 44
Εικόνα 24 : Workload Management System (WMS)..... 51
Εικόνα 25 : Αντιγραφή αρχείου στον UI λογαριασμό..... 54
Εικόνα 26 : Δημιουργία proxy πιστοποιητικού..... 55
Εικόνα 28 : Εμφάνιση αρχείων με την εντολή ls -l..... 55
Εικόνα 29 : Καθορισμός hostname..... 56
Εικόνα 30 : Αντιγραφή ενός αρχείου στο πλέγμα..... 56

Εικόνα 31 : Εύρεση surl και turl αρχείου στο πλέγμα..... 57
Εικόνα 32 : Εμφάνιση πληροφοριών αρχείου..... 58
Εικόνα 33 : Αντιγραφή αρχείου από το πλέγμα στο τοπικό σύστημα του χρήστη.. 58
Εικόνα 34 : Οι διαδρομές μιας εργασίας DAG..... 61
Εικόνα 35 : Η διαδρομή του InputSandbox στο πλέγμα..... 71
Εικόνα 36 : Το OutputSandbox στην ανάκτηση της εξόδου..... 75
Εικόνα 37 : Οι πιθανές καταστάσεις μιας εργασίας..... 80
Εικόνα 38 : Η εντολή glite-wms-job-list-match..... 81
Εικόνα 39 : Υποβολή μιας εργασίας..... 82
Εικόνα 40 : Έλεγχος κατάστασης μιας εργασίας..... 83
Εικόνα 41 : Ανάκτηση εξόδου εργασίας..... 83
Εικόνα 42 : Εμφάνιση εξόδου εργασίας..... 84
Εικόνα 43 : Εμφάνιση της εξόδου..... 85
Εικόνα 44 : Δημιουργία φακέλου και αρχείου στο UI..... 86
Εικόνα 45 : Δημιουργία ενός αρχείου..... 87
Εικόνα 46 : Το jdl αρχείο..... 87
Εικόνα 47 : Έξοδος από την δημιουργία αρχείου..... 88
Εικόνα 48 : Ανάκτηση εξόδου και αποθήκευση στον φάκελο results..... 89
Εικόνα 49 : Εμφάνιση αποτελεσμάτων εργασίας..... 89
Εικόνα 50 : Τα jdl αρχεία μιας εργασίας collection..... 91
Εικόνα 51 : Έλεγχος κατάστασης μιας εργασίας collection..... 92
Εικόνα 52 : Εμφάνιση αποτελεσμάτων μιας εργασίας collection..... 93
Εικόνα 53 : Τα αρχεία μιας Parametric εργασίας..... 95
Εικόνα 54 : Τα αποτελέσματα μιας Parametric εργασίας..... 96
Εικόνα 55 : Η εντολή i2glogin..... 99
Εικόνα 56 : Τα παράθυρα επικοινωνίας και υποβολής..... 101
Εικόνα 57 : Έναρξή επικοινωνίας με τον WN..... 101
Εικόνα 58 : Οι εντολές hostname και ls..... 102
Εικόνα 59 : Εκτέλεση του script και επικοινωνία με τον χρήστη..... 102
Εικόνα 60 : Το εμβαδό που υπολογίζεται..... 106
Εικόνα 61 : Τα αποτελέσματα της MPI εργασίας από τις τρείς διαφορετικές CPU. 109

1. Εισαγωγή

1.1. Το υπολογιστικό πλέγμα GRID

Τι είναι το GRID (υπολογιστικό πλέγμα); Ένας απλός ορισμός θα μπορούσε να ήταν ο ακόλουθος: μία συλλογή από υπολογιστές με απώτερο σκοπό να προσφέρει την αίσθηση ενός πανίσχυρου και αυτοοργανωμένου ιδεατού υπολογιστή ο οποίος στην πραγματικότητα είναι μία μεγάλη συλλογή από διασυνδεδεμένους ετερογενείς υπολογιστές. Το υπολογιστικό πλέγμα δεν μας δίνει απλά την δυνατότητα χρήσης κάποιων αρχείων που είναι αποθηκευμένα στους διασυνδεδεμένους κόμβους τους, όπως το Δίκτυο, αλλά είναι πολύ πιο ισχυρό και ευέλικτο, προσφέροντάς μας την δυνατότητα να εκμεταλλευτούμε την υπολογιστική δύναμη των διασυνδεδεμένων υπολογιστών με διαφανή τρόπο.

Ουσιαστικά, ένας ερευνητής, που βρίσκεται συνεχώς συνδεδεμένος σε δίκτυο υψηλών ταχυτήτων, με τη χρήση του κατάλληλου λογισμικού μπορεί να μοιράζεται την υπολογιστική ισχύ των υπολογιστών του, τον αποθηκευτικό του χώρο και τους άλλους πόρους του εργαστηρίου του με χιλιάδες άλλους ερευνητές στον κόσμο. Ο διαμοιρασμός αυτός μπορεί να γίνει με ομοιόμορφο, ασφαλή και κατανεμημένο τρόπο σε παγκόσμιο επίπεδο.

Οι νέες αυτές μέθοδοι των υπολογιστικών συστημάτων Grid, αποτελούν σήμερα την τεχνολογία αιχμής σε παγκόσμιο επίπεδο για την ικανοποίηση (μεταξύ άλλων) υψηλών απαιτήσεων σε υπολογιστική ισχύ και χώρους αποθήκευσης δεδομένων. Το μοντέλο υλοποίησης υπέρ-υπολογιστικών υποδομών που επικρατεί σήμερα είναι ο συνδυασμός προσωπικών υπολογιστών σε συστοιχίες (cluster computing) διασυνδεδεμένοι με δίκτυα υπερ-υψηλών ταχυτήτων, σε αντίθεση με το παλαιότερο μοντέλο των μεγάλων σε μέγεθος και ασύμφορων υπερ-υπολογιστών λόγω του υψηλού κόστους αγοράς, εγκατάστασης, συντήρησης, λειτουργίας και αναβάθμισης. [1]

Ο πρόγονος του πλέγματος Grid είναι το Metacomputing. Αυτός ο όρος επινοήθηκε στις αρχές της δεκαετίας του '80 από το διευθυντή του NCSA (National Center for Supercomputer Applications), Larry Smarr. Η ιδέα του Metacomputing ήταν να διασυνδεθούν υπερυπολογιστικά κέντρα προκειμένου να επιτευχθεί η ανώτερη επεξεργασία των υπολογιστικών πόρων. Μια από τις πρώτες υποδομές σε αυτόν τον τομέα, που ονομάστηκε Information Wide Area Year (I-WAY), επιδείχθηκε το 1995. Αυτό το πρόγραμμα επηρέασε έντονα τις επόμενες δραστηριότητες υπολογιστικού πλέγματος. Στην πραγματικότητα ένας από τους ερευνητές και επικεφαλής του I-WAY προγράμματος ήταν ο Ian Foster ο οποίος μαζί με τον Carl Kesselman, δημοσίευσαν το 1997 ένα έγγραφο που συνδέει άμεσα το Globus Toolkit, ένα λογισμικό ανοιχτού κώδικα που χρησιμοποιείται για την κατασκευή συστημάτων και εφαρμογών Grid και το οποίο είναι αυτήν την περίοδο η καρδιά πολλών προγραμμάτων πλέγματος.

Το 1997 το δίδυμο Foster-Kesselman οργάνωσε στο Argonne National Laboratory, ένα εργαστήριο με τίτλο "Η οικοδόμηση ενός υπολογιστικού πλέγματος». Εκείνη τη στιγμή γεννήθηκε ο όρος "Grid". Το εργαστήρι ακλούθησε το 1998 η δημοσίευση του βιβλίου "Το Grid: προσχέδιο για μία νέα Πληροφορική Υποδομή" από τους ίδιους. Για τους λόγους αυτούς θεωρούνται οι πατέρες του Grid, αλλά και το βιβλίο τους, το οποίο εν τω μεταξύ ξαναγράφηκε πλήρως και δημοσιεύθηκε εκ νέου το 2003, θεωρείται επίσης η "Βίβλος του Grid". [2]

Η σάρωση των CPU (scavening), η χρήση δηλαδή μηχανημάτων σε ώρες που δεν χρησιμοποιούνται και ο υπολογιστικός εθελοντισμός (η παραχώρηση Η/Υ από χρήστες για την διεξαγωγή ενός έργου/προγράμματος) διαδόθηκαν το 1997 από το distributed.net, το πρώτο ερευνητικό πρόγραμμα γενικής χρήσης παράλληλων κατανεμημένων συστημάτων στο διαδίκτυο και στα τέλη του 1999 από την SETI@home, ένα πρόγραμμα υπολογιστικού εθελοντισμού βασισμένο στο διαδίκτυο. [3]

1.2. Εξελισσόμενες Τεχνολογίες GRID

Στη δεκαετία του 1990, εξελίχθηκε μια σειρά σημαντικών τεχνολογιών, αλλάζοντας το περιβάλλον των τεχνολογιών πληροφορικής στο οποίο εξελίχτηκε το Grid.

1. Το World Wide Web, 1990: Ένας ερευνητής στο Κέντρο Πυρηνικών Μελετών και Ερευνών (CERN), ο Tim Berners Lee, ξεκίνησε τις εργασίες για το World Wide Web, κάτι που οδήγησε στην ανακοίνωση του διαδικτυακού ιστού στον κόσμο το 1993. Ο Ιστός διαδραματίζει έναν κρίσιμο ρόλο στην απλούστευση των επικοινωνιών σχεδόν σε όλες τις πτυχές της ζωής.

2. Το λειτουργικό σύστημα Linux, το 1991: Ο Linus Torvalds στη συνέχεια στο Πανεπιστήμιο του Ελσίνκι, άρχισε τις εργασίες πάνω σε αυτό το λειτουργικό σύστημα ανοιχτού κώδικα. Σήμερα, το Linux αποτελεί πόλο έλξης χιλιάδων προγραμματιστών που διασφαλίζουν ότι λειτουργεί με κάθε νέο κομμάτι υλικού (hardware) που συμπεριλαμβάνει. Αυτός είναι ένας λόγος για τον οποίο το Linux είναι το προτιμώμενο λειτουργικό σύστημα για πολλές δοκιμές στο πλέγμα.

3. Beowulf PC clusters, 1994: Η πρώτη συστάδα φθηνών υπολογιστών τέθηκε από τον Donald Becker και τον Thomas Sterling στη NASA. Χρησιμοποίησαν κάρτες Ethernet για την παροχή σύνδεσης υψηλής ταχύτητας μεταξύ των υπολογιστών, μιμούμενοι το είδος της υπολογιστικής δύναμης που ήταν διαθέσιμη μόνο από τους ακριβούς υπερυπολογιστές. Τώρα, τέτοιες συστάδες είναι εμπορικά διαθέσιμες, προσυναρμολογημένες, από τους περισσότερους κατασκευαστές προσωπικών υπολογιστών. Το Grid Computing βασίζεται σε τέτοιες συστάδες.

4. Java, 1995: Μηχανικοί λογισμικού της Sun Microsystems σχεδίασαν αυτή τη γλώσσα προγραμματισμού ώστε να είναι ανεξάρτητη από τον υπολογιστή στον οποίο εκτελείται. Η Java σχεδιάστηκε αρχικά για τις συσκευές των καταναλωτών, και όχι για

υπολογιστές. Η Java είναι μια εξαιρετικά δημοφιλής γλώσσα προγραμματισμού του Grid και καλά προσαρμοσμένη στη φιλοσοφία του. [4]

1.3. Από τους Υπερυπολογιστές στο Grid

- Ισχύς: Ένα τυπικό PC σήμερα είναι τόσο ισχυρό όσο ήταν ένας γιγάντιος υπερυπολογιστής πριν από δέκα χρόνια. Οι υπολογιστές που κυκλοφορούν σήμερα στην αγορά έχουν πάνω από 100 Gb μνήμη, όσο θα μπορούσαν να συγκεντρώσουν ολόκληρα κέντρα υπολογιστών στις αρχές της δεκαετίας του 1990.

- Ταχύτητα: Οι ADSL συνδέσεις από τα σπίτια των χρηστών στο διαδίκτυο έχουν συνήθως ταχύτητες πολύ πάνω από τα 56kbits/sec που ήταν ότι ταχύτερο στη συνδεσιμότητα μεταξύ των μεγάλων κέντρων υπολογισμού υψηλών επιδόσεων, το 1985.

- Συνδυασμός ισχύος και ταχύτητας: Οι συστοιχίες Η/Υ στα σημερινά πλέγματα υπολογιστών ήταν αδιανόητες μόλις πριν από μία δεκαετία. Πλέον, τα δίκτυα ευρείας περιοχής με ταχύτητες Gigabit / sec που συνδέονται με αυτές τις ομάδες είναι κάτι περισσότερο από χίλιες φορές ταχύτερα από ό,τι ήταν διαθέσιμο το 1990. Οι εν λόγω εφαρμογές πλέγματος παρέχουν στους επιστήμονες ταχύτερους και πιο αποτελεσματικούς τρόπους για αυτούς που εργάζονται παγκόσμιες ομάδες. [5]

Το GRID ήδη χρησιμοποιείται από μεγάλες εταιρίες, εκπαιδευτικά ιδρύματα, επιστημονικές κοινότητες κ.α., όπως στο CERN (Conseil Européen pour la Recherche Nucléaire) όπου τα τεράστια δεδομένα που παράγονται από τα πειράματα που διεξάγονται, αποθηκεύονται και αναλύονται με την βοήθεια του πλέγματος. Το World Community Grid επίσης είναι μία παγκόσμια ανθρωπιστική προσπάθεια που έχει ως σκοπό να αξιοποιήσει την αδρανή και αχρησιμοποίητη υπολογιστική ενέργεια από εταιρικούς και προσωπικούς Η/Υ σε όλο τον κόσμο και να την κατευθύνει στην έρευνα για τους σκοπούς αποκωδικοποίησης γενετικών κωδίκων που βρίσκονται πίσω από ασθένειες όπως το AIDS, το Αλτσχάιμερ και ο καρκίνος. Το GRID εφαρμόζεται

ακόμα και στην μετεωρολογία για την βελτίωση των προβλέψεων των φυσικών καταστροφών, αλλά και στον τραπεζικό κλάδο.

1.4. Ευρωπαϊκά έργα με εφαρμογές Grid

Το πρόγραμμα EUROGRID ήταν ένα project έρευνας και τεχνολογικής ανάπτυξης μοιρασμένου κόστους που επιχορηγήθηκε από την Ευρωπαϊκή Επιτροπή. Είναι μέρος του προγράμματος Information Society Technologies (IST). Η περίοδος επιχορήγησης ήταν από 1 Νοεμβρίου 2000 μέχρι και 31 Ιανουαρίου 2004. Το πρόγραμμα EUROGRID κατέδειξε τη χρήση των πλεγμάτων σε επιλεγμένες επιστημονικές και βιομηχανικές κοινότητες, εξέτασε τις συγκεκριμένες απαιτήσεις αυτών των κοινοτήτων, και έδωσε έμφαση στα οφέλη της χρήσης του Grid.

Στόχοι του προγράμματος αυτού ήταν:

- Να καθιερώσουν ένα ευρωπαϊκό δίκτυο GRID με κορυφαίων υπολογιστικά κέντρα υψηλής επίδοσης από διάφορες ευρωπαϊκές χώρες.

- Να λειτουργήσει και να υποστηρίξει την υποδομή λογισμικού EUROGRID. Το λογισμικό EUROGRID χρησιμοποιώντας το υπάρχον δίκτυο Ίντερνετ θα προσφέρει τη συνεχή και ασφαλή πρόσβαση στους χρήστες του EUROGRID.

- Την ανάπτυξη σημαντικών τμημάτων λογισμικού GRID και την ενσωμάτωσή τους στο EUROGRID (γρήγορη μεταφορά αρχείων, εύρεση υπολογιστικών πόρων, διεπαφή για τις συνδεμένες εφαρμογές και διαλογική πρόσβαση).

- Την επίδειξη διανεμημένων κωδίκων προσομοίωσης από διαφορετικούς τομείς εφαρμογής (βιομοριακές προσομοιώσεις, μετεωρολογική πρόβλεψη, εξομοιωτές πτήσεων, δομική ανάλυση, επεξεργασία δεδομένων σε πραγματικό χρόνο).

- Την συμβολή στη διεθνή ανάπτυξη του πλέγματος και να έρχεται σε επαφή με τα κορυφαία διεθνή προγράμματα GRID.

- Να βγάλει στην παραγωγή τα τμήματα λογισμικού του EUROGRID. Μετά την λήξη του έργου το λογισμικό του EUROGRID θα είναι διαθέσιμο, ως υποστηριγμένο προϊόν. [6]

Όλοι οι παραπάνω στόχοι επιτεύχθηκαν κατά την διάρκεια διεξαγωγής του έργου.

Το DataGrid ήταν ένα έργο που χρηματοδοτήθηκε από την Ευρωπαϊκή Ένωση. Ο στόχος ήταν να χτιστεί η επόμενης γενιάς υπολογιστική υποδομή παρέχοντας εντατικό υπολογισμό και ανάλυση μοιρασμένων βάσεων δεδομένων μεγάλης κλίμακας, από εκατοντάδες terabytes σε petabytes, διανεμημένες ευρέως σε όλη την επιστημονική κοινότητα. [7]

Το SEE-GRID-SCI (SEE-GRID eInfrastructure for regional eScience) ήταν ένα πρόγραμμα που ξεκίνησε την 01/05/2008, διήρκησε δύο χρόνια και χρηματοδοτήθηκε από την Ευρωπαϊκή Επιτροπή. Οι ηλεκτρονικές υποδομές στην Ευρώπη έχουν φτάσει σε προχωρημένο στάδιο, όπου το δίκτυο GEANT (το πανευρωπαϊκό δίκτυο δεδομένων που αφιερώνεται στην ερευνητική και εκπαιδευτική κοινότητα) αποτελεί τον κορμό των επικοινωνιών πάνω στο οποίο μία κατανεμημένη υποδομή πληροφορικής - το Grid - παρέχει υπηρεσίες επεξεργασίας και αποθήκευσης για την έρευνα στις ηλεκτρονικές επιστήμες. Οι πρωτοβουλίες των ηλεκτρονικών υποδομών στις χώρες της Νοτιοανατολικής Ευρώπης δεσμεύεται να εξασφαλίσει ισότιμη συμμετοχή των χωρών με τους λιγότερους πόρους. [8]

1.5. Hellas Grid

Το HellasGrid είναι η ελληνική υποδομή υπολογιστικού πλέγματος (Grid), η μεγαλύτερη στην νοτιανατολική Ευρώπη, και μια από τις σταθερότερες υποδομές σε Ευρωπαϊκό επίπεδο. Πόροι της υποδομής χρησιμοποιούνται από έλληνες ερευνητές και από ευρωπαϊκά προγράμματα. Τα τελευταία χρόνια σημαντικός και αυξανόμενος αριθμός χρηστών από διάφορα επιστημονικά πεδία (φυσική, βιοτεχνολογία, υπολογιστική χημεία, πληροφορική, μετεωρολογία, κ.ά) χρησιμοποιεί την υποδομή

HellasGrid για τις υπολογιστικές τους ανάγκες. Η πρόσβαση είναι ελεύθερη στην ελληνική ακαδημαϊκή και ερευνητική κοινότητα με μία απλή διαδικασία εγγραφής.

Έχει ως στόχο την παροχή υπολογιστικών υπηρεσιών υψηλής απόδοσης (HighPerformanceComputing, HighThroughputComputing) στην ελληνική ακαδημαϊκή και ερευνητική κοινότητα. Την εγκατάσταση, λειτουργία και υποστήριξη έξι υπολογιστικών και αποθηκευτικών κόμβων ανά την Ελλάδα – HellasGrid (Αθήνα (3), Θεσσαλονίκη, Πάτρα, Ηράκλειο). Την λειτουργία Αρχής Πιστοποίησης HellasGrid (CertificationAuthority) για έκδοση πιστοποιητικών χρηστών. Την υποστήριξη χρηστών στη μεταφορά των εφαρμογών τους στην υποδομή. Την υποστήριξη εφαρμογών και παροχή βιβλιοθηκών και λογισμικού υποστήριξης.

Η υποδομή HellasGrid προσαρμόζεται και ανταποκρίνεται συνεχώς στις απαιτήσεις των χρηστών της και έχει δρομολογηθεί επέκταση των προσφερόμενων πόρων της ώστε να καλυφθούν επιπλέον υπολογιστικές ανάγκες που θα προκύψουν στα προσεχή χρόνια.

Το HellasGrid, έχοντας εθνική εμβέλεια, έρχεται να οργανώσει και να δέσει τις πρώτες αυτές προσπάθειες που αναπτύσσονται από την ερευνητική κοινότητα κυρίως σε θέματα eScience, αλλά και να δώσει κατευθύνσεις τόσο προς τον Ελληνικό εμπορικό κόσμο σε θέματα eBusiness, όσο και στη Δημόσια Διοίκηση σε θέματα εξυπηρέτησης του πολίτη και ηλεκτρονικής διακυβέρνησης (eGovernment). Στη προσπάθεια αυτή συναντώνται το αντίστοιχο ανθρωποδίκτυο και οι υποδομές που διαθέτει η χώρα μας και οργανώνονται έτσι σε έναν κόμβο άμεσης επικοινωνίας με αντίστοιχες προσπάθειες που γίνονται στο εξωτερικό. [9]

1.6. Διεθνείς Ερευνητικές Πρωτοβουλίες με Ελληνική Συμμετοχή

Αναμφισβήτητα στη χώρα μας τον τελευταίο καιρό έχουν ενταθεί οι προσπάθειες παρακολούθησης των ευρωπαϊκών και διεθνών εξελίξεων σε τεχνολογίες πλέγματος Grid. Αν και υπάρχουν αντικειμενικές δυσκολίες, το τεχνολογικό χάσμα με περισσότερο προηγμένες χώρες σε αυτά τα θέματα δεν καθίσταται πλέον απαγορευτικό για τη συνεργασία μαζί τους και τη συμμετοχή μας σε πρωτοβουλίες που αυτές οργανώνουν. Στην ερευνητική και επιστημονική κοινότητα ιδίως, υπάρχει έντονο ενδιαφέρον για τις υπηρεσίες Grid. Αντιπροσωπευτική του ενδιαφέροντος αυτού είναι η συμμετοχή ερευνητικών ομάδων (αλλά και εταιρειών) από τον Ελλαδικό χώρο σε διεθνείς πρωτοβουλίες με άμεση (Crossgrid, GridLab, Gria, EGEE) ή και έμμεση στόχευση σε τεχνολογίες Grid (Open Source).

Συγκεκριμένα στα πλαίσια του Crossgrid (http://www.crossgrid.org) υπάρχει ενεργός εμπλοκή του Αριστοτέλειου Πανεπιστημίου Θεσσαλονίκης (ΑΠΘ) και του ΕΚΕΦΕ Δημόκριτος σε εφαρμογές Φυσικής Υψηλών Ενεργειών αλλά και της εταιρείας Algosystems που δραστηριοποιείται στον ευρύτερο χώρο της Πληροφορικής και Τηλεπικοινωνιών.

Το GridLab (http://www.gridlab.org) είναι μια πρωτοβουλία που εστιάζει στην υλοποίηση ενός γενικού πλαισίου ανάπτυξης εφαρμογών σε περιβάλλον Grid. Ανέκυψε από τη συνειδητοποίηση πως το Grid πρέπει να απευθύνεται στον τελικό χρήστη που αναπτύσσει εφαρμογές χωρίς απαραίτητα να τον εμπλέκει στην πολυπλοκότητα που εμπεριέχει η ίδια η Grid-υποδομή. Το Εθνικό Μετσόβιο Πολυτεχνείο (ΕΜΠ) μετέχει σε αυτό το πρόγραμμα.

Το GRIA (http://www.gria.org) δίνει έμφαση στη χρήση του Grid σε βιομηχανικό περιβάλλον σε επίπεδο εφαρμογών όπως παραγωγή και επεξεργασία ψηφιακών ταινιών ή δομική ανάλυση. Ως εκ τούτου εστιάζει στη διασφάλιση της Ποιότητας των Υπηρεσιών (QoS), την υιοθέτηση προτύπων και την ασφάλεια, ενσωματώνοντας την εμπειρία του GridLab. Το ΕΜΠ και η εταιρεία ΚΙΝΟ, με

δραστηριότητα κυρίως στο χώρο της παραγωγής διαφημιστικών ταινιών, είναι οι ελληνικοί εταίροι σε αυτήν την προσπάθεια.

Το EGEE, (http://egee-ei.web.cern.ch) είναι το ευρωπαϊκό πλαίσιο ανάπτυξης τεχνολογιών πλέγματος υπολογιστικών συστημάτων Grid. Στο EGEE εντάσσονται οι εθνικές ή ευρύτερης περιοχής (regional) προσπάθειες για ανάπτυξη τεχνολογιών Πλέγματος. Το Hellasgrid εκπροσωπείται μέσω του ΕΔΕΤ (Εθνικό Δίκτυο Έρευνας και Τεχνολογίας) που συμμετέχει ως εταίρος αλλά ουσιαστικά θα υπάρξει συμμετοχή από το σύνολο της ομάδας εργασίας και έχει επιδείξει μια αυξημένη κινητικότητα σε αυτό το χώρο.

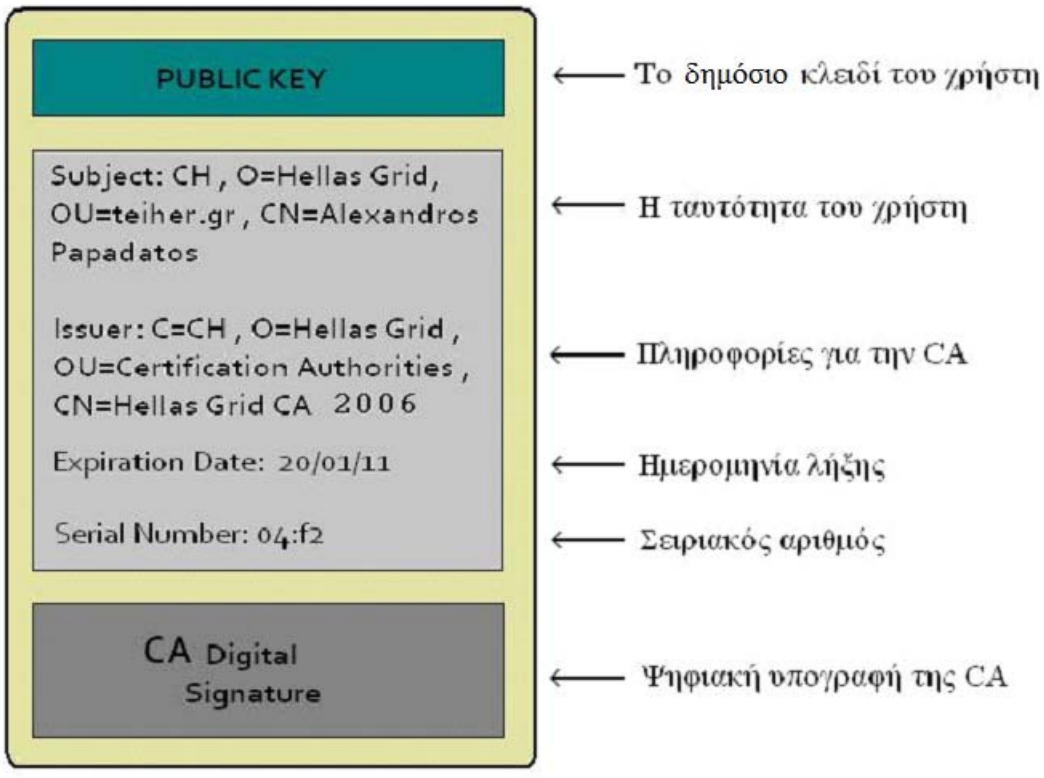
Το SEE-GRID (http://www.see-grid.org) με συντονιστή το ΕΔΕΤ / GRNET έχει στόχο τη δημιουργία ανθρώπινου δικτύου στην περιοχή της Νοτιοανατολικής Ευρώπης για την προώθηση των τεχνολογιών Πλέγματος. Το SEE-GRID θα υλοποιήσει κλίνες δοκιμών Πλέγματος σε όλες τις χώρες της περιοχής με έμφαση στα Δυτικά Βαλκάνια αποβλέποντας στην σταδιακή ένταξή τους στις Πανευρωπαϊκές Υποδομές σε στενή συνεργασία με το πρόγραμμα EGEE. [10]

2. Πρόσβαση στην υποδομή του Grid

2.1. Απόκτηση Ψηφιακού Πιστοποιητικού

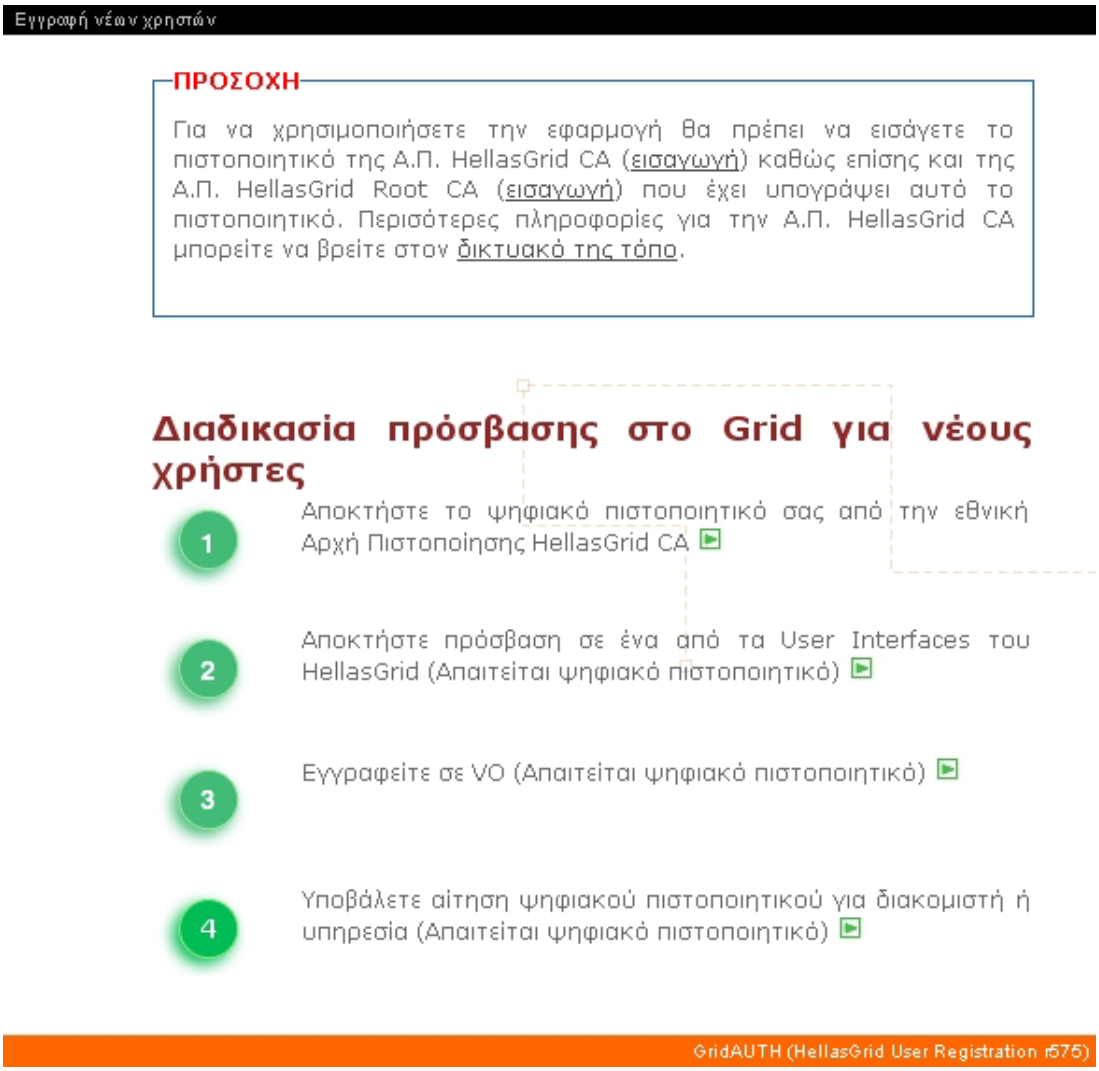
Η πιστοποίηση ενός χρήστη ή διακομιστή στην υποδομή γίνεται από μία κεντρική υπηρεσία, την Αρχή Πιστοποίησης Α.Π. (Certificate Authority CA) η οποία παρέχει ψηφιακά πιστοποιητικά υπογεγραμμένα από την ίδια. Για να ταυτοποιηθεί η κάθε οντότητα και να υπογραφεί έτσι ένα ψηφιακό πιστοποιητικό, απαραίτητη είναι η παρεμβολή μίας Αρχής Ταυτοποίησης Α.Τ. (Registration Authority RA) η οποία αναλαμβάνει την διαδικασία αυτή.

Το Ψηφιακό πιστοποιητικό είναι ένα ηλεκτρονικό έγγραφο που χρησιμοποιείται για την αναγνώριση μίας οντότητας (φυσικό πρόσωπο, εξυπηρετητής, οργανισμός κοκ) στο διαδίκτυο και την ανάκτηση του δημοσίου κλειδιού (public key) αυτής, το οποίο χρησιμοποιείται για την κρυπτογράφηση δεδομένων.

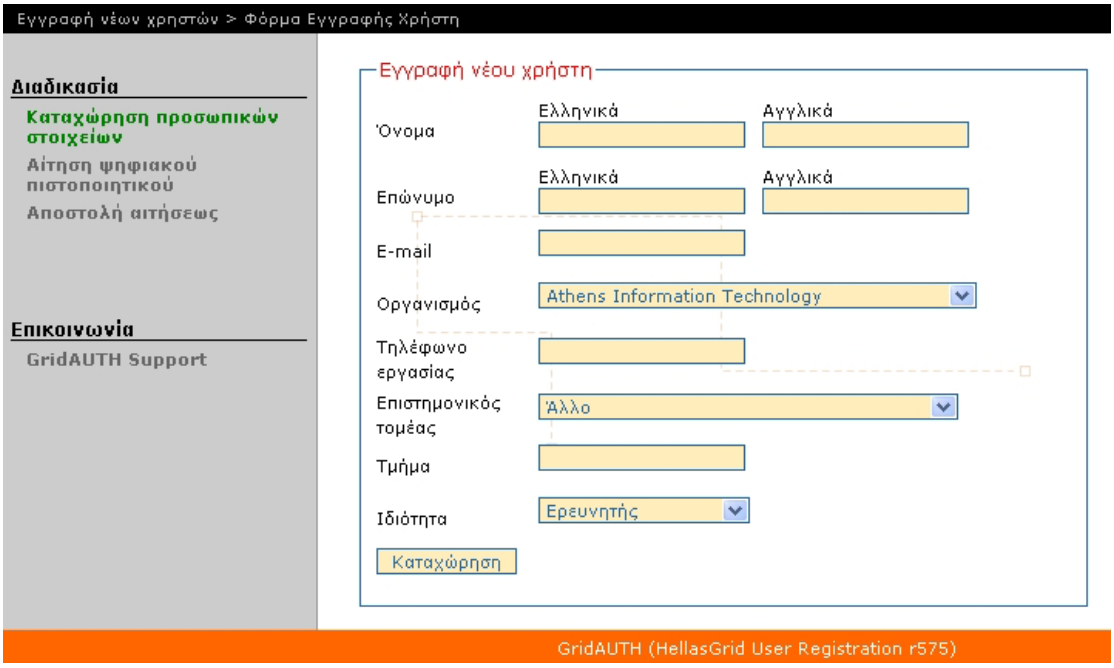


Εικόνα 1 : Το ψηφιακό πιστοποιητικό

Στην Ελλάδα η διαδικασία αυτή μπορεί να γίνει μέσω της ομάδας εργασίας HellasGrid που έχει ως σκοπό την διαμόρφωση Εθνικής Στρατηγικής και τον συντονισμό των δράσεων των τεχνολογιών Grid για την Ελλάδα.Ο χρήστης μπορεί να υποβάλλει αίτηση απόκτησης ψηφιακού πιστοποιητικού στην εθνική Αρχή Πιστοποίησης HellasGrid CA. Στο site www.hellasgrid.gr/ και ακολουθώντας τα links ο χρήστης θα πρέπει να καταχωρήσει τα προσωπικά του στοιχεία και να αποστείλει την αίτηση.

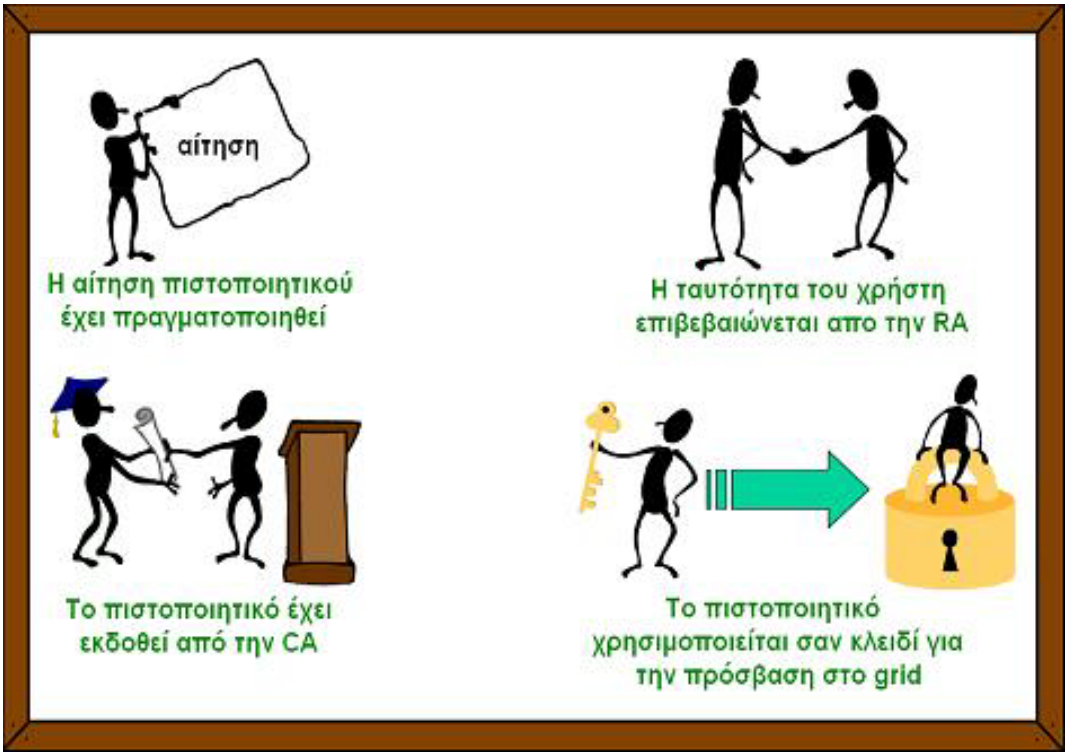


Εικόνα 2 : Αίτημα ψηφιακού πιστοποιητικού στο HellasGrid (https:// access.hellasgrid.gr/)



Εικόνα 3 : Φόρμα εγγραφής χρήστη στο HellasGrid

Έπειτα θα λάβει στο ηλεκτρονικό του ταχυδρομείο ένα e-mail όπου θα του ζητείται να προσκομίσει ο ίδιος τα προσωπικά του στοιχεία εντός μίας εβδομάδος, στην Αρχή Ταυτοποίησης η οποία εξυπηρετεί την περιοχή του, ώστε να επιβεβαιωθεί ότι η αίτηση για πιστοποιητικό έγινε από τον ίδιο.



Εικόνα 4 : Η διαδικασία απόκτησης ψηφιακού πιστοποιητικού

2.2. Εγκατάσταση Πιστοποιητικού

Ο χρήστης ύστερα από μερικές ημέρες θα λάβει ένα δεύτερο e-mail όπου θα βεβαιώνεται η απόκτηση του ψηφιακού πιστοποιητικού του από την HellasGrid CA. Το πιστοποιητικό θα πρέπει να εγκατασταθεί στον browser που χρησιμοποιήθηκε από τον χρήστη για την αίτηση καθώς και τον ίδιο υπολογιστή. Αυτό γίνεται ακολουθώντας το link που δίνεται στο e-mail. Στην συνέχεια ο χρήστης θα πρέπει να παράγει το δημόσιο και το ιδιωτικό κλειδί του όπως αυτό εξηγείται στην ηλεκτρονική σελίδα όπου θα οδηγηθεί. Τα ιδιωτικά και δημόσια κλειδιά (αντίστοιχα private key, public key) είναι αλληλουχίες χαρακτήρων (strings) συνήθως σε δεκαεξαδική μορφή HEX, που χρησιμοποιούνται για την κρυπτογράφηση και αποκρυπτογράφηση δεδομένων. Το ιδιωτικό κλειδί μπορεί να το διαβάσει μόνο ο κάτοχος του, ενώ το δημόσιο όλοι οι χρήστες στο πλέγμα.

Το ψηφιακό πιστοποιητικό και το private κλειδί βρίσκονται πλέον αποθηκευμένα και προστατευμένα στο software security device του browser.

2.3. Εγγραφή σε εικονικό οργανισμό (Virtual Organization)

Απαραίτητη προϋπόθεση για κάθε χρήστη είναι η εγγραφή σε τουλάχιστον έναν εικονικό οργανισμό (Virtual Organization VO) ο οποίος έχει την επίβλεψη της σωστής λειτουργίας της υποδομής καθώς και της συνεργασίας μεταξύ των μελών του. Ένας εικονικός οργανισμός (VO) είναι ένα σύνολο ατόμων ή / και ιδρυμάτων με κοινό σκοπό να μοιράζονται τους πόρους τους. Δεν υπόκεινται σε κάποιον ενιαίο ιεραρχικό έλεγχο και ενώνουν τις δυνάμεις τους για να λύσουν ένα συγκεκριμένο πρόβλημα, φέρνοντας σε συνεργασία ένα υποσύνολο των πόρων τους, την κατανομή αυτών κατά την κρίση τους και υπό τις δικές τους προϋποθέσεις. Οι πόροι αυτοί είναι συνήθως υπολογιστές, δεδομένα, λογισμικό, τεχνογνωσία και τα μέσα (εργαλεία). Υπάρχουν διάφοροι εικονικοί οργανισμοί σε όλο τον κόσμο όπως π.χ. ο ALICE collaboration, High Energy Physics experiment at CERN LHC (ALICE) ή ο United States ATLAS Collaboration

(ATLAS). Για την Ελλάδα οι χρήστες μπορούν να εγγραφούν στον VO SEE (South Eastern Europe) ο οποίος αφορά τους χρήστες της νοτιοανατολικής Ευρώπης. [11]

2.4. User Interface

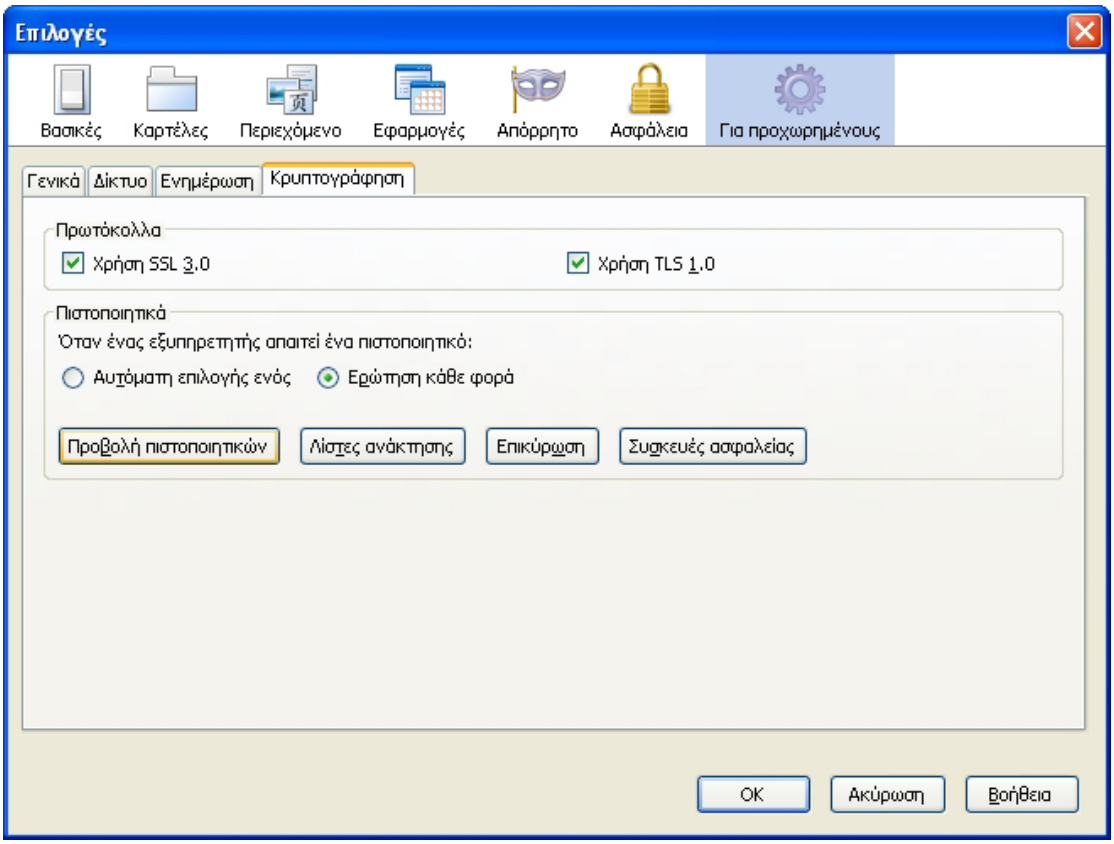
Κάθε χρήστης που επιθυμεί την πρόσβαση στην υποδομή του Grid πρέπει να έχει έναν λογαριασμό σε ένα User Interface UI (διεπαφή χρήστη). Το UI είναι ο σύνδεσμος ή το παράθυρο επικοινωνίας μεταξύ ενός χρήστη και ενός προγράμματος ηλεκτρονικού υπολογιστή. Είναι ένα σύνολο εντολών ή επιλογών μέσω του οποίου ένας χρήστης επικοινωνεί με ένα πρόγραμμα. [12]

Θα πρέπει λοιπόν να γίνει μία αίτηση απόκτησης λογαριασμού στο https://access.hellasgrid.gr/ σε κάποιο από τα UI ακολουθώντας το link. Ο χρήστης θα πρέπει να επιλέξει την πόλη στην οποία βρίσκεται (εάν η πόλη στην οποία εργάζεται ο χρήστης δεν βρίσκεται ανάμεσα στις διαθέσιμες επιλογές τότε θα πρέπει να επιλέξει την Αθήνα) και να στείλει την αίτηση. Αφού γίνει δεκτό το αίτημά του θα λάβει ένα e-mail με το username και ένα password με τα οποία θα έχει πρόσβαση στο Grid. Στην περίπτωση αυτή θα χρησιμοποιηθεί ως παράδειγμα διεπαφής χρήστη UI : ui01.isabella.grnet.gr .

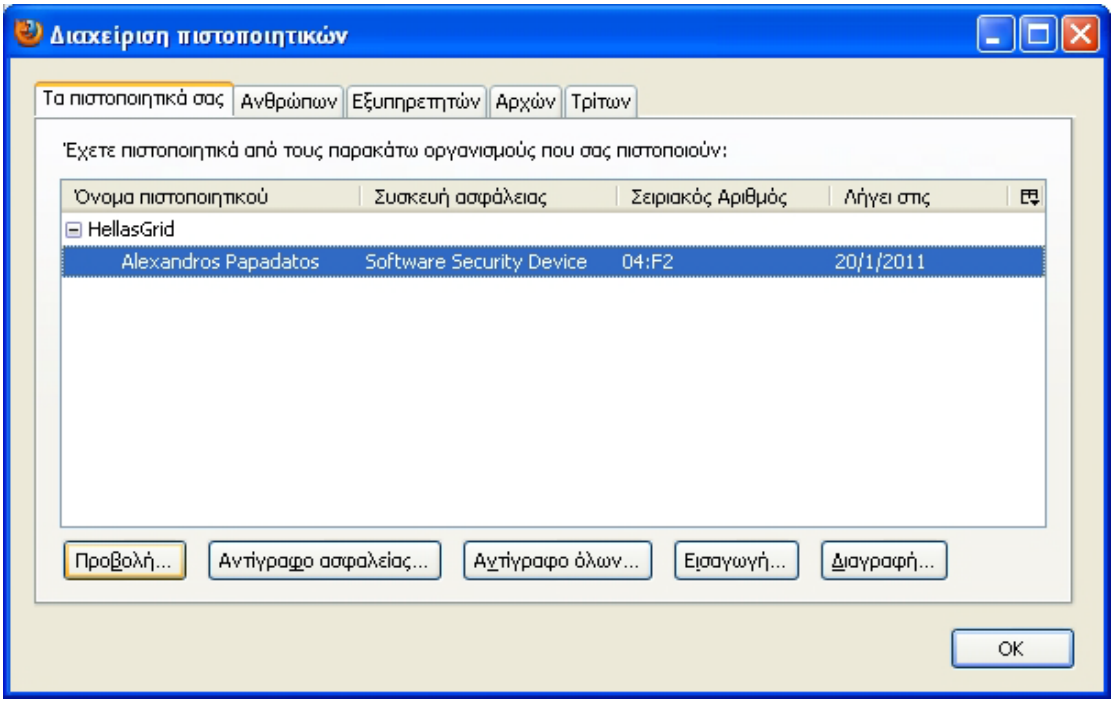
2.5. Εξαγωγή του Πιστοποιητικού

Όπως αναφέρεται παραπάνω, το πιστοποιητικό του χρήστη μαζί με το ιδιωτικό του κλειδί βρίσκεται στο software security device στον browser. Από εκεί ο χρήστης πρέπει να τα εξάγει σε p12 ή pfx format αναλόγως τι τύπου browser χρησιμοποιεί και να τα μεταφέρει στο User Interface ή σε άλλες εφαρμογές (π.χ. mail client). Η διαδικασία είναι η εξής : Για το Internet Explorer είναι: Tools → Internet Options.. → Content → Certificates → επιλέγουμε το πιστοποιητικό → Export , για το

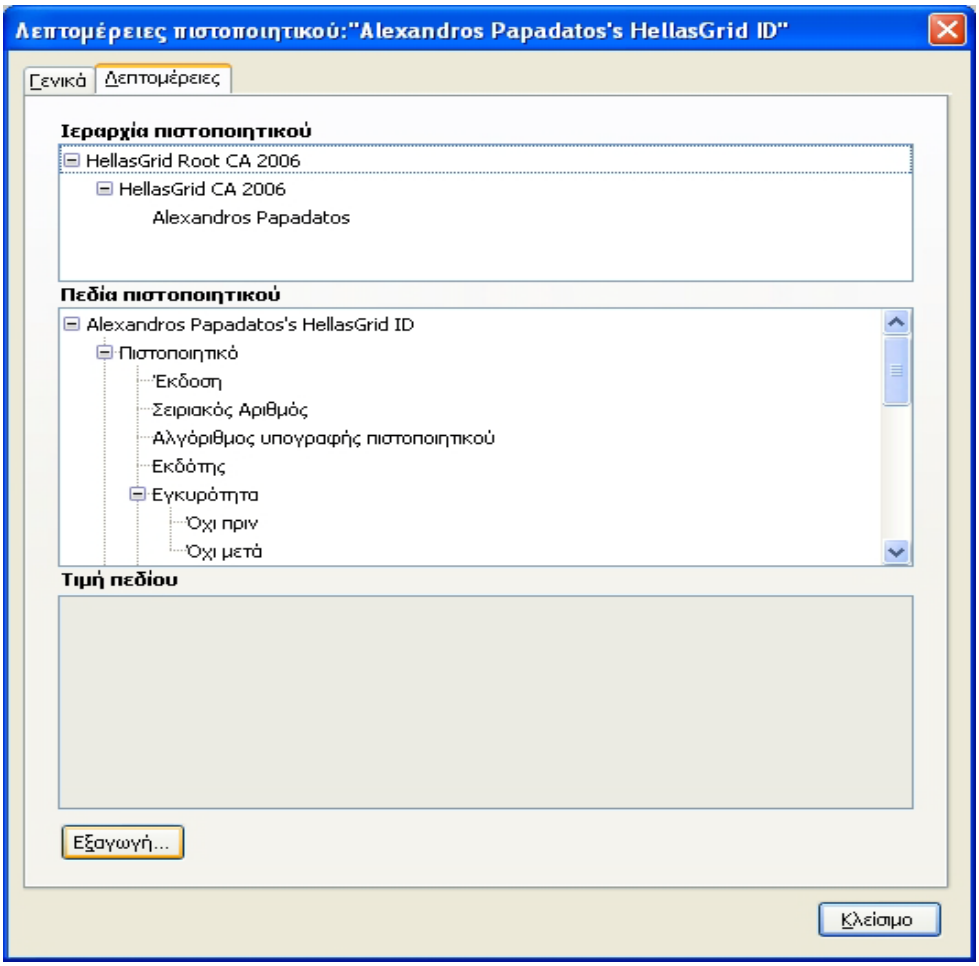
Mozilla Firefox: Εργαλεία→ Επιλογές → Για Προχωρημένους → Προβολή Πιστοποιητικών → Τα πιστοποιητικά μου → Λεπτομέρειες → Εξαγωγή. Μπορεί ακόμα να γίνει η επιλογή αντίγραφο ασφαλείας στο ίδιο μενού εάν δεν λειτουργήσει σωστά η παραπάνω διαδικασία και όπου θα ζητηθεί στον χρήστη να θέσει έναν κωδικό που θα πληκτρολογεί κάθε φορά που θα δημιουργεί ένα αντίγραφο του πιστοποιητικού του. [11]



Εικόνα 5 : Το πιστοποιητικό στον browser



Εικόνα 6 : Προβολή του πιστοποιητικού

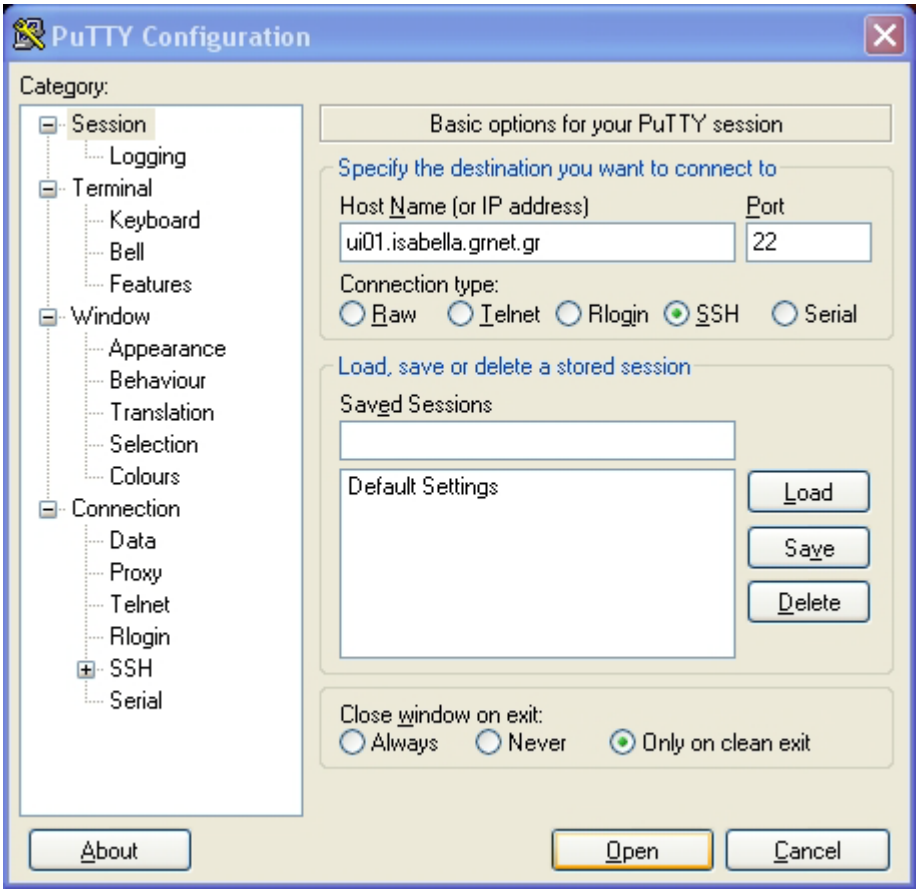


Εικόνα 7 : Λεπτομέρειες πιστοποιητικού και εξαγωγή

2.6. Αντιγραφή του πιστοποιητικού στο UI

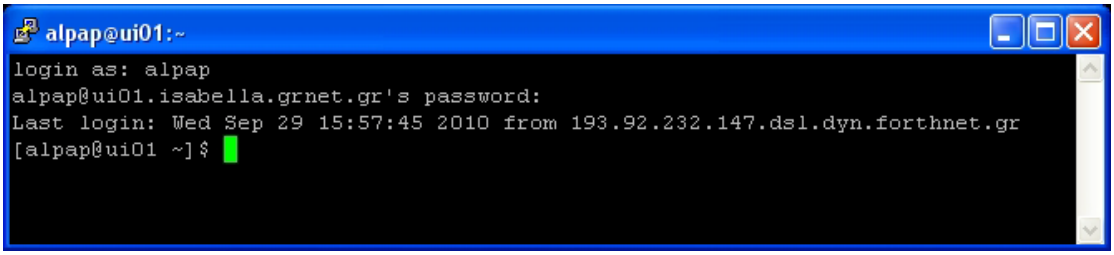
Για να εισέλθει κάποιος στην υποδομή πλέγματος θα πρέπει να λειτουργεί τον Η/Υ με λειτουργικό σύστημα Linux. Εάν ο χρήστης όμως χρησιμοποιεί Windows θα πρέπει να κατεβάσει το πρόγραμμα PuTTY (www.**putty**.org/) που είναι μία εφαρμογή SSH (ένα πρωτόκολλο δικτύου που επιτρέπει την ανταλλαγή δεδομένων μέσω ενός ασφαλούς καναλιού μεταξύ δύο συσκευών δικτύου) και μέσω της οποίας μπορεί ο χρήστης να εισέλθει από το τερματικό του στο UI. Μαζί με το PuTTY θα μεταφορτωθεί και το PSCP, ένα λογισμικό SCP (Secure copy) πρόγραμμα για το τερματικό των Windows. Μπορείτε να χρησιμοποιήσετε αυτό το πρόγραμμα αντί του FTP για την αντιγραφή αρχείων προς ή από τους διακομιστές Unix.

Όλα είναι πλέον έτοιμα για την είσοδο στην υποδομή και την υποβολή εργασιών. Ανοίγουμε το PuTTY και πληκτρολογούμε το Host Name, στην περίπτωση αυτή, το ui01.isabella.grnet.gr και κλικ στο open.



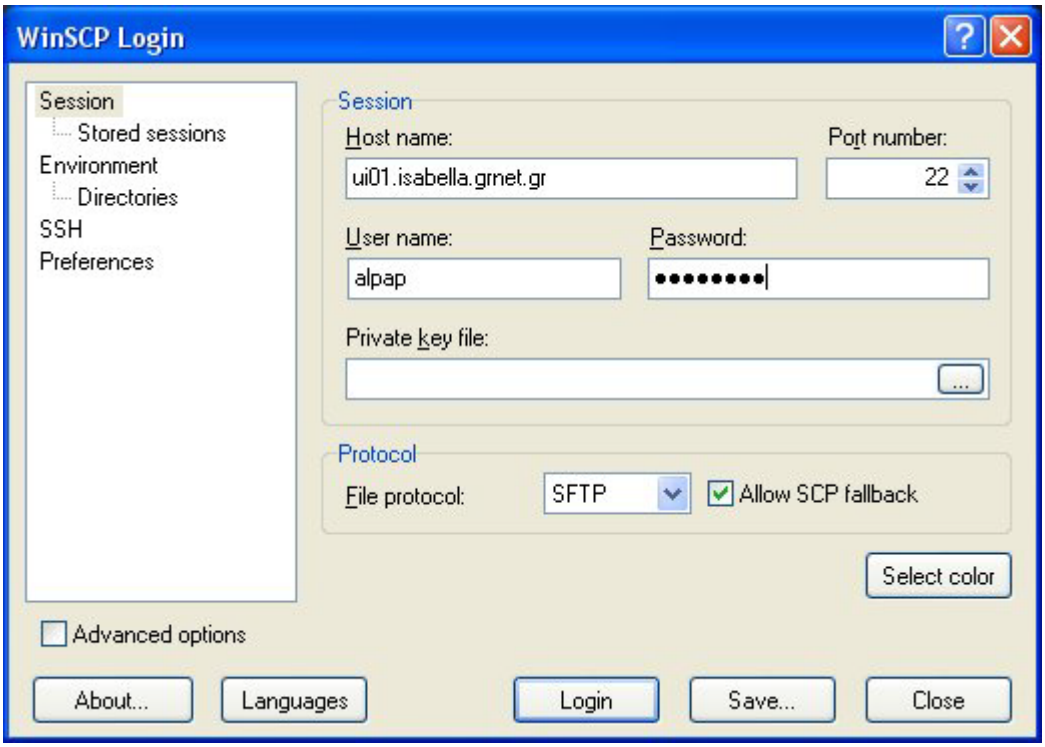
Εικόνα 8 : Το πρόγραμμα puTTY

Ο χρήστης βρίσκεται πλέον στο UI και για να εισέλθει στον λογαριασμό του θα πρέπει να δώσει το username και το password.

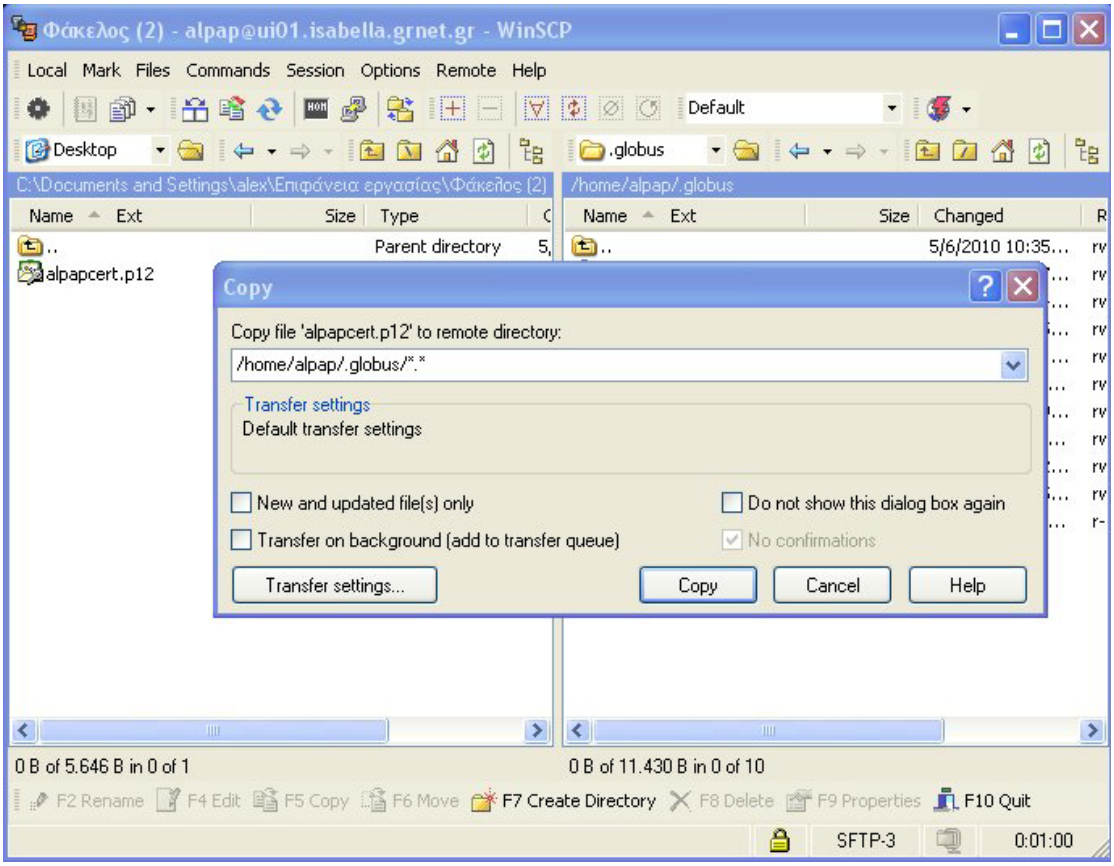


Εικόνα 9 : Εισαγωγή username και password στο UI

Αφού γίνει εγκατάσταση του προγράμματος PSCP μπορεί πλέον να γίνει αντιγραφή του πιστοποιητικού στον λογαριασμό του χρήστη στο UI. Το πιστοποιητικό έχει εξαχθεί από τον browser και βρίσκεται αποθηκευμένο στον Η/Υ του χρήστη σε μορφή .p12 ή .pfx. Γράφουμε το όνομα του UI στο Host Name καθώς και το user name με το password αντίστοιχα. Στην συνέχεια μπορούμε να κάνουμε drag and drop (δηλ. να σύρουμε) το πιστοποιητικό μας στον λογαριασμό μας. Το πιστοποιητικό θα πρέπει να το αντιγράψουμε στον φάκελο .globus που υπάρχει στον λογαριασμό μας.



Εικόνα 10 : Είσοδος στο πρόγραμμα WinSCP



Εικόνα 11 : Αντιγραφή του πιστοποιητικού στον UI λογαριασμό

2.7. Πρώτα βήματα και βασικές εντολές UNIX

Τα πρώτα βήματα είναι να εκτελεστούν οι δύο παρακάτω εντολές ώστε να παραχθούν από το πιστοποιητικό τα αρχεία usercert.pem το οποίο περιέχει το δημόσιο κλειδί του χρήστη και userkey.pem για το ιδιωτικό:

- **openssl pkcs12 -nocerts -in <alpapcert.p12> -out ~/.globus/userkey.pem**
- **openssl pkcs12 -clcerts -nokeys -in <alpapcert.p12> -out ~/.globus/usercert.pem**

όπου alpapcert.p12 το αποθηκευμένο όνομα του ψηφιακού πιστοποιητικού με την κατάληξη που έχει αποθηκευτεί στον Η/Υ, δηλ. .p12 ή .pfx .

[Αν δεν υπάρχει ο φάκελος .globus στον λογαριασμό του χρήστη θα πρέπει να τον δημιουργήσει εκτελώντας την εντολή **mkdir .globus** και ύστερα να μετακινήσει το πιστοποιητικό του στον φάκελο με την εντολή **mv alpapcert .globus** . Μετά ακολουθούν οι παραπάνω δύο εντολές.]

Εκτελώντας την εντολή για το userkey.pem θα ζητηθεί από τον χρήστη ο κωδικός που έθεσε όταν έκανε αντίγραφο ασφαλείας και να θέσει έναν νέο κωδικό, το PEM Pass Phrase για την προστασία του ιδιωτικού του κλειδιού. Για το usercert.pem χρειάζεται μόνο ο πρώτος κωδικός. [13]

Στη συνέχεια τα δύο κλειδιά πρέπει να ασφαλιστούν:

- **chmod 644 usercert.pem**

Με την εντολή αυτή το δημόσιο κλειδί μπορεί να διαβαστεί από όλους αλλά μόνο ο χρήστης μπορεί να το διαγράψει.

- **chmod 600 userkey.pem**

Μόνο ο χρήστης μπορεί να έχει πρόσβαση στο ιδιωτικό κλειδί του και να το διαβάσει.

Μπορείτε να δείτε το πιστοποιητικό σας με την εντολή :

- **grid-cert-info**

Για να δούμε τους φακέλους και τα αρχεία που περιέχονται σε έναν φάκελο πληκτρολογούμε :

- **ls –al**

Ενώ για να ανοίξουμε ένα φάκελο π.χ. το **.globus** :

- **cd .globus**

και για να πάμε πίσω :

- **cd ..**

Εικόνα 12 : Η εντολή cd

Για την δημιουργία ενός άδειου αρχείου :

- **touch αρχείο.xxx**

ενώ για την διαγραφή του

- **rm αρχείο.xxx**

Έξοδος από τον λογαριασμό :

- **exit**

[14]

2.8. Proxy Certificate

Πριν υποβάλλουμε οποιαδήποτε εργασία στην υποδομή, είναι απαραίτητη η δημιουργία ενός proxy certificate (πληρεξούσιο πιστοποιητικό). Με τον τρόπο αυτό εξουσιοδοτούμε το πιστοποιητικό ώστε να εγκρίνει ενέργειες για όλα τα στοιχεία του Grid (SE, CE, ...) και αποτρέπεται η ανάγκη για την αποστολή του κλειδιού του χρήστη στο δίκτυο. Έχοντας αυτό το πληρεξούσιο δεν είναι απαραίτητη η πληκτρολόγηση

κωδικών για την επιβεβαίωση της ταυτότητας του χρήστη κάθε φορά που υποβάλλει ένα αίτημα.

Η διαδικασία παραγωγής του γίνεται με την παρακάτω εντολή :

- **voms-proxy-init --voms=see**

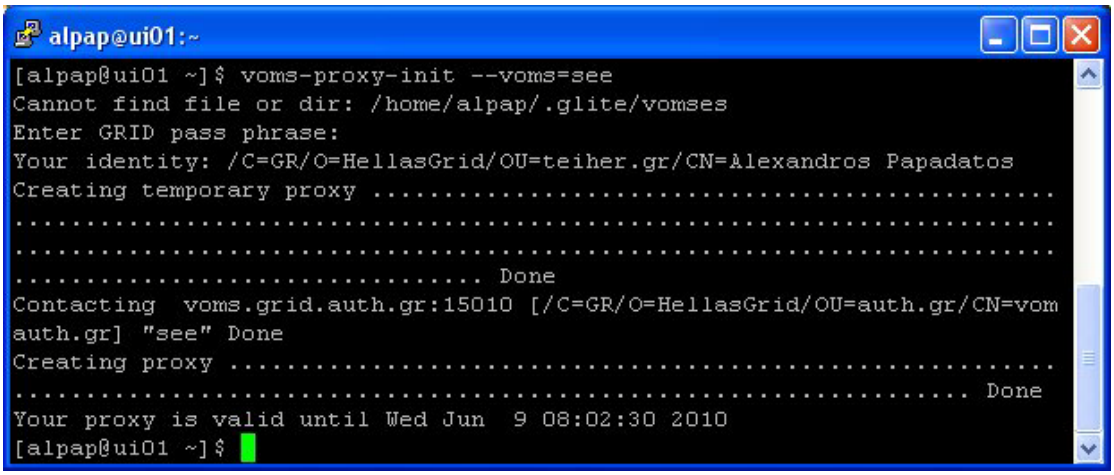
Θα ζητηθεί από το χρήστη ο κωδικός που είχε θέσει για την προστασία του ιδιωτικού του κλειδιού, το PEM pass phrase ή Grid pass phrase.

Όπου voms σημαίνει Virtual Organization Membership System, οπότε χρειάζεται να προσδιορίσουμε σε ποιον εικονικό οργανισμό ανήκουμε : voms=see.

Μπορούμε να δημιουργήσουμε πληρεξούσιο επίσης δίνοντας :

- **grid-proxy-init**

αλλά στην περίπτωση αυτή το πιστοποιητικό δεν θα βρίσκεται υπό την προστασία του εικονικού οργανισμού στον οποίο είμαστε εγγεγραμμένοι αλλά θα είναι ποιο γενικό.



Εικόνα 13 : Δημιουργία proxy πιστοποιητικού

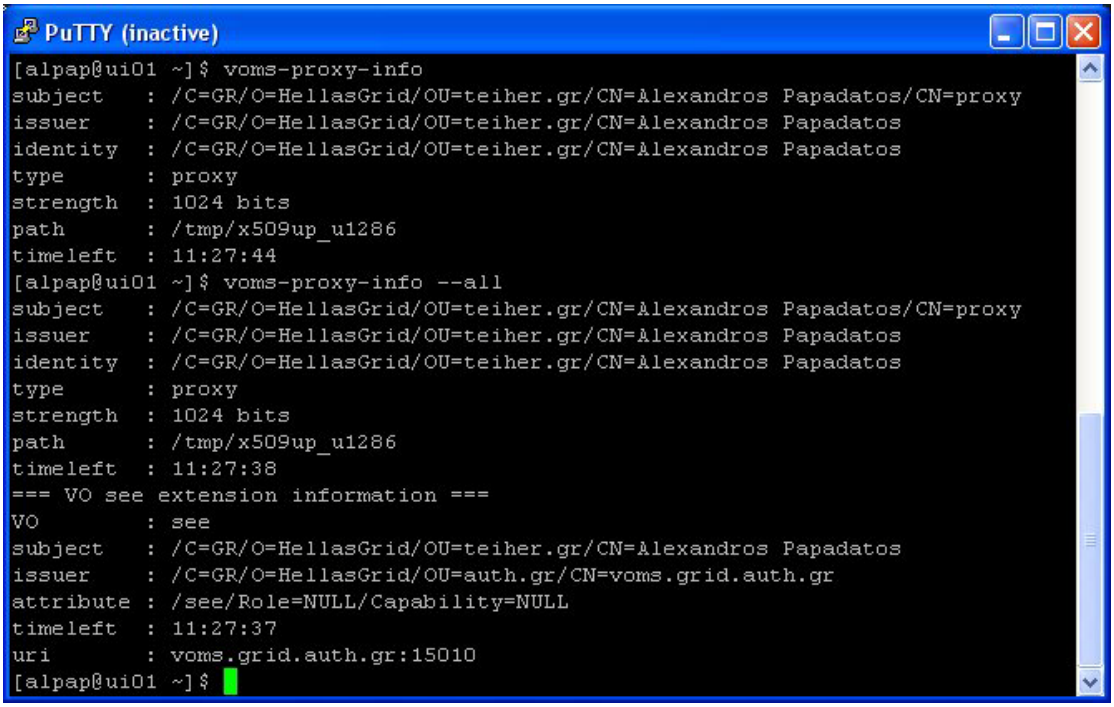
Το πληρεξούσιο αυτό πιστοποιητικό ισχύει για δώδεκα ώρες, αλλά ο χρήστης αν θέλει μπορεί να αλλάξει την διάρκεια ισχύος του εάν οι εργασίες που θα υποβάλλει διαρκέσουν περισσότερο. Αυτό γίνεται προσθέτοντας την επέκταση **–hours**, π.χ. :

- **voms-proxy-init --voms=see -hours 20** ,για 20 ώρες.

Πληροφορίες για το πιστοποιητικό δίνονται με την εντολή :

- **voms-proxy-info**

προσθέτοντας την επέκταση **--all** , δίνονται πιο αναλυτικές πληροφορίες.



Εικόνα 14 : Πληροφορίες για το proxy certificate

[15]

3. FTP και GridFTP

3.1. File Transfer Protocol

Το File Transfer Protocol FTP (Πρωτόκολλο Μεταφοράς Αρχείων), είναι ένα ευρέως χρησιμοποιούμενο πρωτόκολλο σε δίκτυα τα οποία υποστηρίζουν την στοίβα πρωτοκόλλων TCP/IP (δίκτυα όπως internet ή intranet). Ο υπολογιστής που τρέχει την εφαρμογή FTP client μόλις συνδεθεί με τον αντίστοιχο εξυπηρετητή, μπορεί να εκτελέσει ένα πλήθος διεργασιών όπως ανέβασμα αρχείων στον εξυπηρετητή (file upload), κατέβασμα αρχείων από τον εξυπηρετητή, μετονομασία ή διαγραφή αρχείων από τον εξυπηρετητή κ.ο.κ. Το πρωτόκολλο ακολουθεί ένα ανοιχτό πρότυπο. Είναι δυνατό κάθε υπολογιστής που είναι συνδεδεμένος σε ένα δίκτυο, να διαχειρίζεται αρχεία σε έναν άλλο υπολογιστή του δικτύου, ακόμη και εάν ο δεύτερος διαθέτει διαφορετικό λειτουργικό σύστημα. [16]

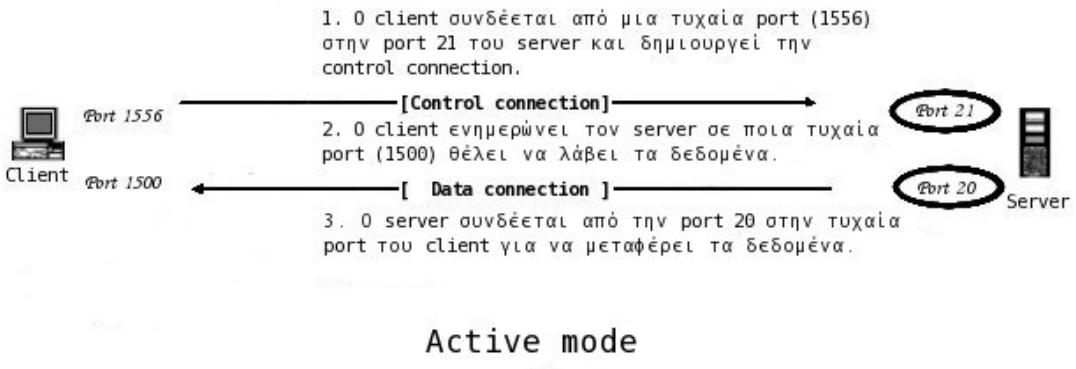
3.1.1. Τρόπος Λειτουργίας

Αρχικά ο FTP server ανοίγει την θύρα (port) 21 περιμένοντας την εφαρμογή ‘FTP client’ να συνδεθεί. Στη συνέχεια ο client ξεκινά μια νέα σύνδεση από μια τυχαία θύρα του υπολογιστή στον οποίο εκτελείται προς την θύρα 21 του εξυπηρετητή. Μόλις γίνει η σύνδεση, αυτή παραμένει ανοιχτή για όλη τη διάρκεια της συνόδου FTP. Η συγκεκριμένη σύνδεση ονομάζεται σύνδεση ελέγχου (control connection).

Έπειτα, δημιουργείται η σύνδεση δεδομένων (data connection), με την οποία μεταφέρονται τα δεδομένα. Υπάρχουν δύο τρόποι για να δημιουργηθεί, με χρήση της ενεργητικής λειτουργίας (active mode) ή με χρήση της παθητικής λειτουργίας (passive mode).

3.1.2. Ενεργητική Λειτουργία

Στην ενεργητική λειτουργία (active mode) ο FTP client διαλέγει μια τυχαία θύρα στην οποία δέχεται τα δεδομένα της σύνδεσης. Ο client στέλνει τον αριθμό της θύρας, στην οποία επιθυμεί να "ακούει" (listen) για εισερχόμενες συνδέσεις. Ο FTP server δημιουργεί μια σύνδεση από την θύρα 20 στην ανοιχτή θύρα του client για τη μεταφορά των δεδομένων. Οποιαδήποτε πληροφορία ζητήσει ο client, ανταλλάσσεται με βάση αυτή τη σύνδεση, που βασίζεται στο TCP. Όταν η μεταφορά ολοκληρωθεί ο εξυπηρετητής κλείνει τη σύνδεση αποστέλλοντας ένα πακέτο FIN (ένα bit που υποδηλώνει το τέλος της αποστολής δεδομένων από τον χρήστη), όπως σε κάθε σύνδεση βασισμένη στο TCP. Κάθε φορά που ο client ζητάει δεδομένα, δημιουργείται κατά παρόμοιο τρόπο μια σύνδεση δεδομένων και η διαδικασία επαναλαμβάνεται.

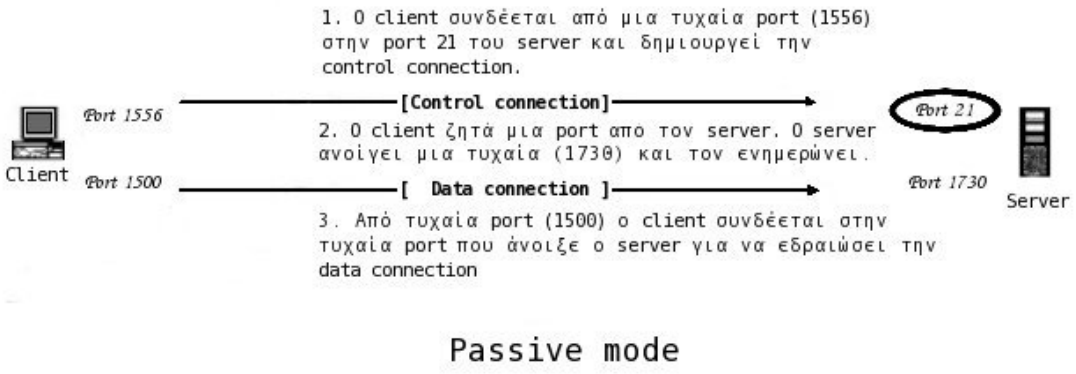


Εικόνα 15 : Ενεργητική λειτουργία σύνδεσης δεδομένων

3.1.3. Παθητική Λειτουργία

Στην παθητική λειτουργία (passive mode) ο client ζητά από τον εξυπηρετητή να διαλέξει μια τυχαία θύρα, στην οποία θα "ακούει" (listen) για την σύνδεση δεδομένων (data connection). Ο server ενημερώνει τον client για την θύρα την οποία έχει διαλέξει και ο client συνδέεται σε αυτή για τη μεταφορά των δεδομένων. Η μεταφορά

ολοκληρώνεται όπως και στην ενεργητική λειτουργία (active mode), αφού η σύνδεση δεδομένων βασίζεται στο TCP.



Εικόνα 16 : Παθητική λειτουργία σύνδεσης δεδομένων

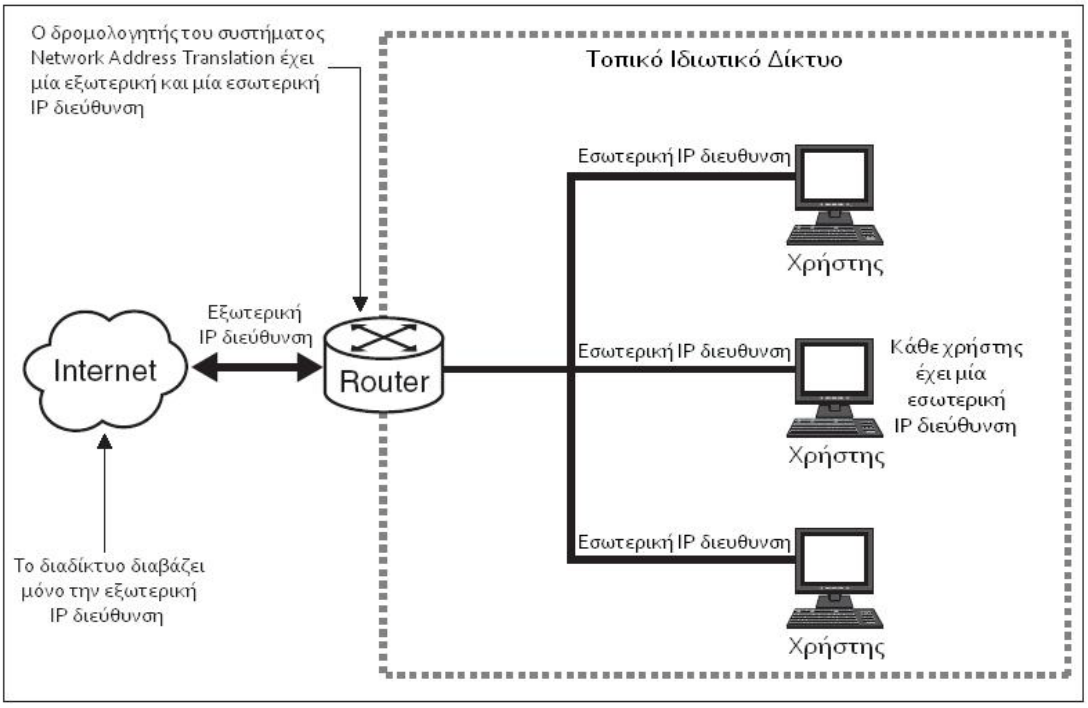
3.1.4. Χρήση

Το FTP είναι ένα πρωτόκολλο πελάτη-εξυπηρετητή 8-bit, ικανό να χειρίζεται οποιοδήποτε τύπο αρχείου χωρίς περαιτέρω επεξεργασία όπως δηλαδή κάνει το MIME (Multipurpose Internet Mail Extensions: ένα πρότυπο δικτύου για την ηλεκτρονική αλληλογραφία). Ωστόσο το FTP έχει εξαιρετικά υψηλή λανθάνουσα κατάσταση (latency). Αυτό σημαίνει ότι ο χρόνος μεταξύ του αιτήματος και της διαδικασίας παραλαβής του είναι αρκετά μεγάλος και γι'αυτό μερικές φορές απαιτείται μακρά διαδικασία σύνδεσης.

3.1.5. Ασφάλεια

Το FTP δεν σχεδιάστηκε με πρόνοια για ασφάλεια, με συνέπεια οι εφαρμογές να είναι ιδιαίτερα ευάλωτες και να εμφανίζονται ποικίλα προβλήματα κατά τη χρήση firewall ή NAT. Η Μετάφραση Διευθύνσεων Δικτύου NAT (Network Address Translation) σχεδιάστηκε για απλοποίηση και διατήρηση των IP διευθύνσεων αφού αυτό που κάνει είναι να επιτρέπει σε ιδιωτικά δίκτυα που χρησιμοποιούν εσωτερικές IP διευθύνσεις (internal IP adress) να έχουν σύνδεση με το Internet. Το σύστημα NAT

λειτουργεί σε κάποιον δρομολογητή (router), ο οποίος συνδέει συνήθως δύο δίκτυα και μεταφράζει τις ιδιωτικές (μη μοναδικές στον παγκόσμιο ιστό) διευθύνσεις του εσωτερικού δικτύου σε εξωτερικές IP διευθύνσεις (external IP adress) προτού τα πακέτα προωθηθούν σε άλλο δίκτυο.



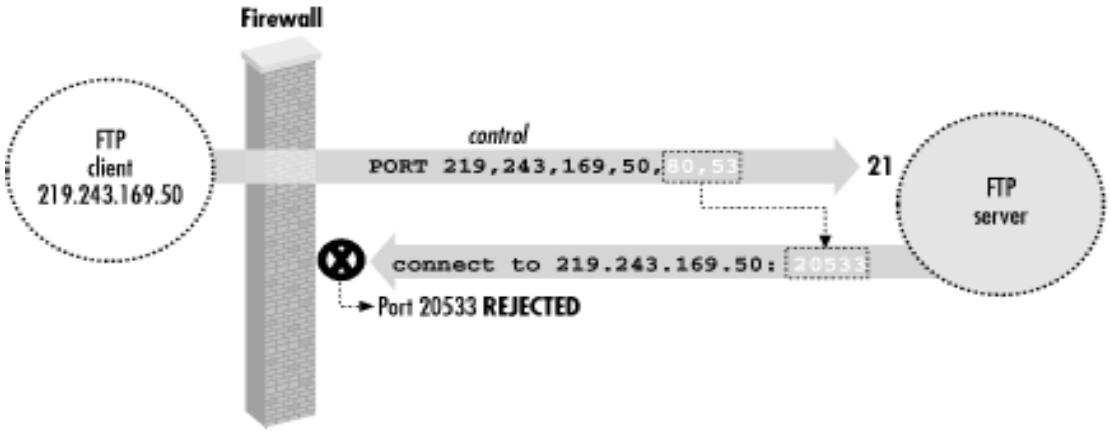
Εικόνα 17 : Network Address Translation (NAT)

3.1.6. Προβλήματα NAT

Στην ενεργητική λειτουργία ο FTP server ξεκινά μια σύνδεση δεδομένων συνδεόμενος στην εξωτερική διεύθυνση IP της πύλης (gateway) NAT. Στην άλλη πλευρά, το μηχάνημα το οποίο είναι υπεύθυνο για τη "μετάφραση" των εσωτερικών διευθύνσεων IP του δικτύου στην εξωτερική, θα πάρει το SYN πακέτο (το πακέτο αυτό υποδηλώνει την δημιουργία μίας νέας σύνδεσης). Όμως, στον πίνακα κατάστασης (state table) του NAT, στον οποίο διατηρείται το ιστορικό μεταφράσεων, δεν έχει καταγραφεί κανένα, με αποτέλεσμα το πακέτο να απορρίπτεται (γίνεται drop). Το

πακέτο δεν φτάνει ποτέ στον client, δεν σχηματίζεται σύνδεση δεδομένων και η μεταφορά δεδομένων είναι αδύνατη. [17]

Στην παθητική λειτουργία, επειδή η θύρα στην οποία συνδέεται ο server είναι τυχαία, είναι πιθανόν να μην επιτρέπεται σύνδεση προς τον αριθμό της από το λογισμικό - τείχος προστασίας (firewall). Σε αυτή την περίπτωση η σύνδεση δεδομένων δεν θα πραγματοποιηθεί και, επομένως, δεν θα μεταφέρονται δεδομένα.

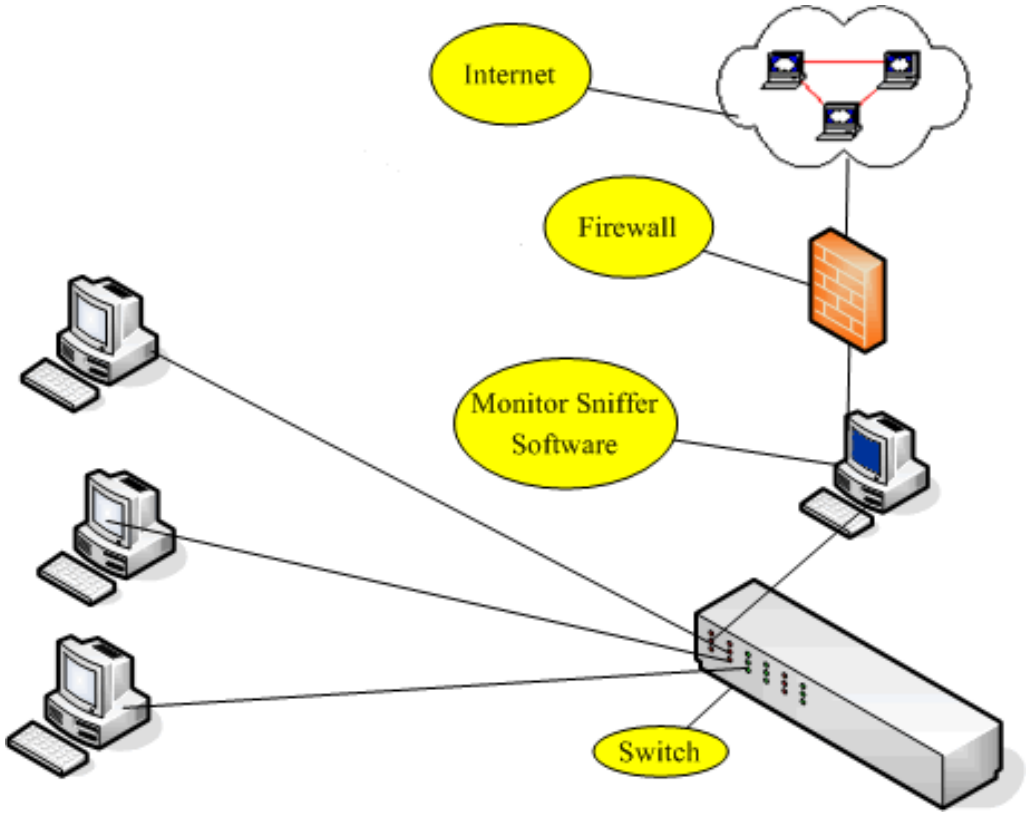


Εικόνα 18 : Πρόβλημα NAT με firewall

3.1.7. Έλλειψη κρυπτογράφησης

Τα δεδομένα που ανταλλάσσονται μέσω FTP δεν είναι κρυπτογραφημένα, με αποτέλεσμα οι εντολές που αποστέλλονται μέσω της control connection να είναι απλό κείμενο. Για το λόγο αυτό μπορούν εύκολα, με τη χρήση ενός sniffer, να καταγραφούν, να διαβασθούν και να ξανασταλθούν ανάλογα με τη βούληση του επιτιθέμενου. Packet sniffer ή απλώς sniffer, επίσης αποκαλούμενο network monitor ή network analyzer, είναι λογισμικό με δυνατότητα παρακολούθησης των πακέτων ενός δικτύου. Όταν γίνει αντιληπτό κάποιο πακέτο το οποίο ικανοποιεί συγκεκριμένα κριτήρια, καταγράφεται σε ένα αρχείο. Ανάμεσα στις εντολές που αποστέλλονται, αυτή που χρησιμοποιείται για να γίνει login σε ένα λογαριασμό FTP, με σύνταξη "PASS password", παρέχει στον επιτιθέμενο τον κωδικό του χρήστη. Αν συνδυαστεί με την

εντολή "USER", με την οποία αποστέλλεται το όνομα του χρήστη, ο επιτιθέμενος μπορεί να χρησιμοποιήσει τα στοιχεία για να εισέλθει στον ξένο λογαριασμό με τα ίδια δικαιώματα. [18]



Εικόνα 19 : Packet Sniffer

Επειδή οι περισσότεροι άνθρωποι τείνουν να επαναχρησιμοποιούν κωδικούς, ο επιτιθέμενος έχει αυξήσει τις πιθανότητες του με μια brute-force attack (επίθεση ωμής βίας) δηλαδή μια εξαντλητική δοκιμή πιθανών κλειδιών που παράγουν ένα κρυπτογράφημα, ώστε να αποκαλυφθεί το αρχικό μήνυμα. Με αυτό τον τρόπο, είναι πιθανό να αποκτήσει έλεγχο του συστήματος του χρήστη μόλις βρει τη διεύθυνση IP του, ανιχνεύοντας την έναρξη της συνόδου FTP (FTP session).
[19]

3.2. GridFTP

Το GridFTP είναι μια επέκταση του προτύπου File Transfer Protocol (FTP) για χρήση στις τεχνολογίες πλέγματος. Ορίζεται ως μέρος της εργαλειοθήκης Globus, υπό τη διοργάνωση του Global Grid Forum (συγκεκριμένα, από την ομάδα εργασίας GridFTP).

Στόχος του GridFTP είναι να παρέχει μια πιο αξιόπιστη και υψηλών επιδόσεων μεταφορά αρχείων για εφαρμογές στο Grid. Αυτό είναι αναγκαίο λόγω των αυξημένων απαιτήσεων για τη μετάδοση δεδομένων στα πλέγματα για συντόμευση χρόνου μετάδοσης και αξιοπιστία.

Το GridFTP είναι η απάντηση στο πρόβλημα της ασυμβατότητας μεταξύ των συστημάτων αποθήκευσης και πρόσβασης. Στο παρελθόν, κάθε πάροχος δεδομένων θα καταστούσε τα δεδομένα του διαθέσιμα, παρέχοντας μια βιβλιοθήκη των λειτουργιών πρόσβασης. Αυτό κατέστησε δύσκολη την συγκέντρωση δεδομένων από πολλαπλές πηγές, απαιτώντας μια διαφορετική μέθοδο πρόσβασης για την καθεμιά, και κατά συνέπεια τη διαίρεση του συνόλου των διαθέσιμων δεδομένων σε κομμάτια.

Το GridFTP παρέχει έναν ομοιόμορφο τρόπο πρόσβασης στα δεδομένα, που περιλαμβάνει τις λειτουργίες όλων των διαφορετικών τρόπων πρόσβασης, βασισμένο και σε επέκταση του παγκοσμίως αποδεκτού προτύπου FTP. Το FTP επιλέχθηκε ως βάση για αυτό, λόγω της εκτεταμένης χρήσης του, και επειδή έχει μια καλά καθορισμένη αρχιτεκτονική που επιτρέπει τις επεκτάσεις του πρωτοκόλλου.

Το GridFTP είναι χρήσιμο για πολλούς λόγους συμπεριλαμβανομένης της ταχύτερης μεταφοράς και της ενσωματωμένης ασφάλειας. Αυτό επιτυγχάνεται με τις ακόλουθες αλλαγές στο κανονικό FTP. [20]

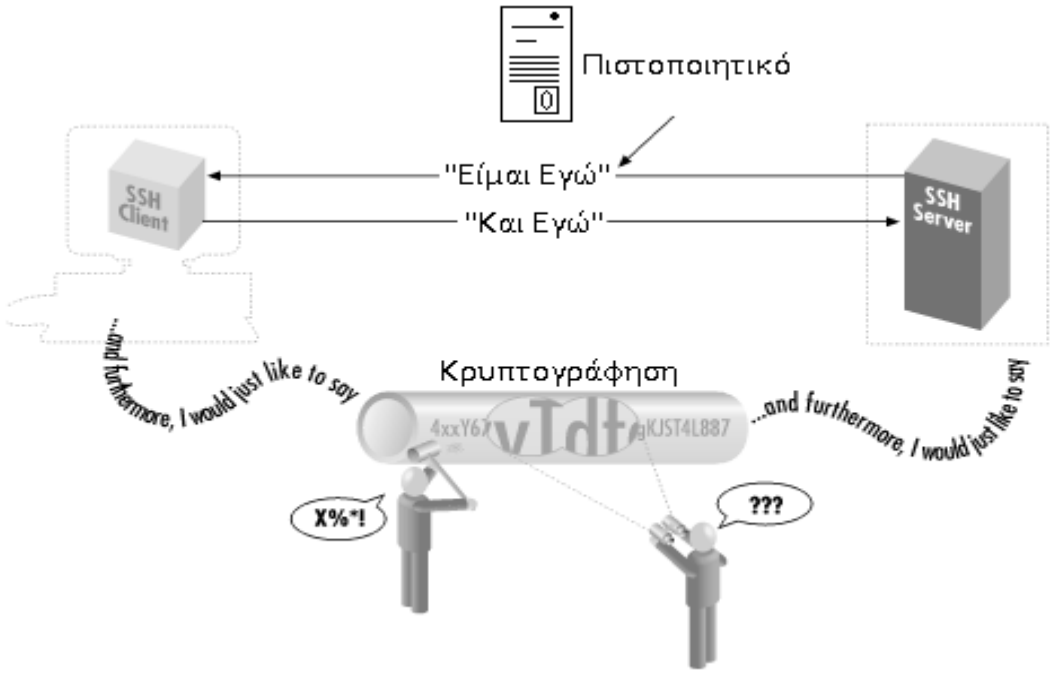
3.2.1. Ασφάλεια με GSI

Το Grid Security Infrastructure GSI είναι ένα άλλο εργαλείο του Globus το οποίο παρέχει τον έλεγχο ταυτότητας και την κρυπτογράφηση για μεταφορές αρχείων, με το χρήστη σε συγκεκριμένα επίπεδα εμπιστευτικότητας και ακεραιότητας των δεδομένων. Το FTP από μόνο του δεν είναι ασφαλές, συνεπώς είναι ευάλωτο σε υποκλοπές, και γι'αυτό επικαλείται τα πρωτόκολλα SSH και SSL για ασφάλεια.

Το SSL (Secure Sockets Layer) είναι ένα πρωτόκολλο ασφαλούς επικοινωνίας μεταξύ Η/Υ. Κάθε σύνδεση SSL ξεκινά πάντα με την ανταλλαγή μηνυμάτων και ψηφιακών πιστοποιητικών από τον εξυπηρετητή και τον πελάτη/χρήστη έως ότου επιτευχθεί η ασφαλής σύνδεση, πράγμα που ονομάζεται χειραψία (handshake). Η χειραψία επιτρέπει στον εξυπηρετητή να αποδείξει την ταυτότητά του στον πελάτη χρησιμοποιώντας τεχνικές κρυπτογράφησης δημοσίου κλειδιού και στην συνέχεια επιτρέπει στον πελάτη και τον εξυπηρετητή να συνεργαστούν για την δημιουργία ενός συμμετρικού κλειδιού ή κλειδιού μιας χρήσης που θα χρησιμοποιηθεί στην γρήγορη κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων που ανταλλάσσονται μεταξύ τους. [21]

Εικόνα 20 : Secure Sockets Layer (SSL)

Το SSH (Secure SHell) είναι και αυτό ένα πρωτόκολλο για ασφαλή επικοινωνία και ανταλλαγή δεδομένων με την χρήση ενός καναλιού. Χρησιμοποιείται ευρέως από τους διαχειριστές του δικτύου για τον έλεγχο του διαδικτύου και για την πρόσβαση σε απομακρυσμένους υπολογιστές και την εκτέλεση εντολών. Και τα δύο άκρα της σύνδεσης του πελάτη με τον εξυπηρετητή πιστοποιούνται χρησιμοποιώντας ένα ψηφιακό πιστοποιητικό, και τους κωδικούς πρόσβασης που είναι κρυπτογραφημένοι. [22]



Εικόνα 21 : Secure SHell (SSH)

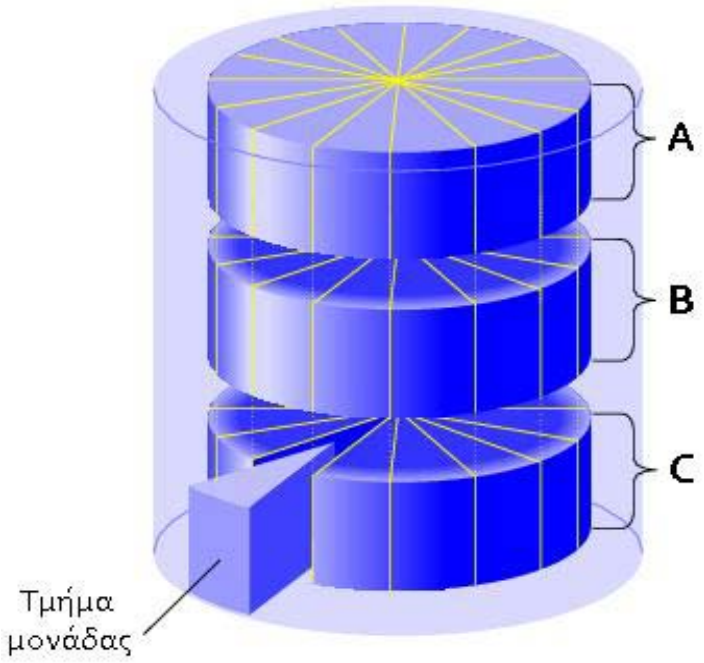
3.2.2. Μεταφορές Τρίτων Προσώπων

Ένα χρήσιμο χαρακτηριστικό του FTP είναι ότι επιτρέπει την απομακρυσμένη μεταφορά μεταξύ διακομιστών από ένα τοπικό χρήστη. Το GridFTP προσθέτει την ασφάλεια και την ταυτοποίηση. Αυτό το χαρακτηριστικό είναι παρόμοιο με το FXP (File eXchange Protocol) σε ορολογία FTP. Το FXP επιτρέπει την αντιγραφή αρχείων από έναν FTP-εξυπηρετητή σε έναν άλλο χρησιμοποιώντας έναν FXP-client. Κανονικά η

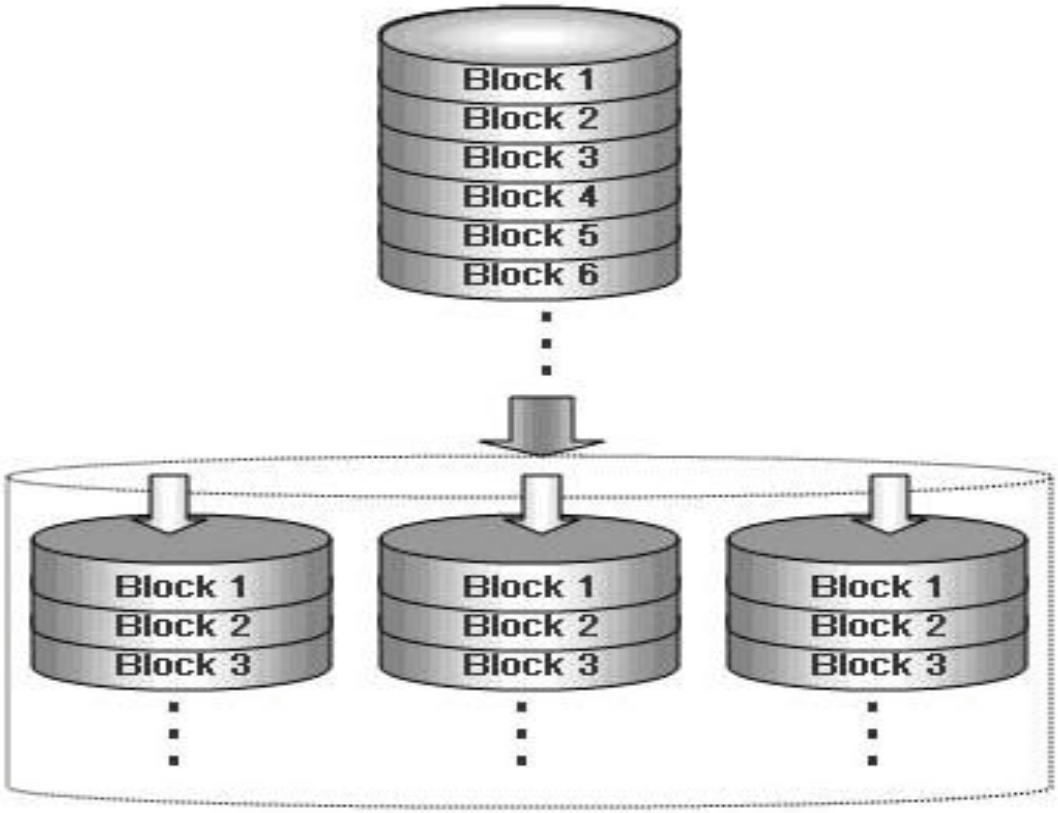
μεταφορά αρχείων γίνεται με την χρήση του πρωτοκόλλου FTP μεταξύ του υπολογιστή ενός χρήστη και ενός FTP-εξυπηρετητή, και η μέγιστη ταχύτητα μεταφοράς εξαρτάται από την ταχύτητα της σύνδεσης του χρήστη στο Internet (π.χ. 56k, καλωδιακή ή T1). Κατά τη μεταφορά αρχείων μεταξύ δύο απομακρυσμένων εξυπηρετητών χρησιμοποιώντας έναν FXP-client, η μέγιστη ταχύτητα μεταφοράς δεν εξαρτάται μόνο από την σύνδεση του ενός αλλά και από τις δύο κάτι που επιτυγχάνει γρηγορότερη μεταφορά αρχείων. [23]

3.2.3. Παράλληλη και Τμηματική μεταφορά

Το GridFTP επιτυγχάνει πολύ μεγαλύτερη χρήση του εύρους ζώνης, επιτρέποντας πολλαπλές ταυτόχρονες μεταφορές. Αρχεία μπορούν να μεταφορτωθούν ταυτόχρονα, τμηματικά, από πολλαπλές πηγές ή ακόμα και σε διακριτές παράλληλες ροές από την ίδια πηγή. Οι μεταφορές αυτές επιτρέπουν αύξηση της ταχύτητας μεταφοράς.



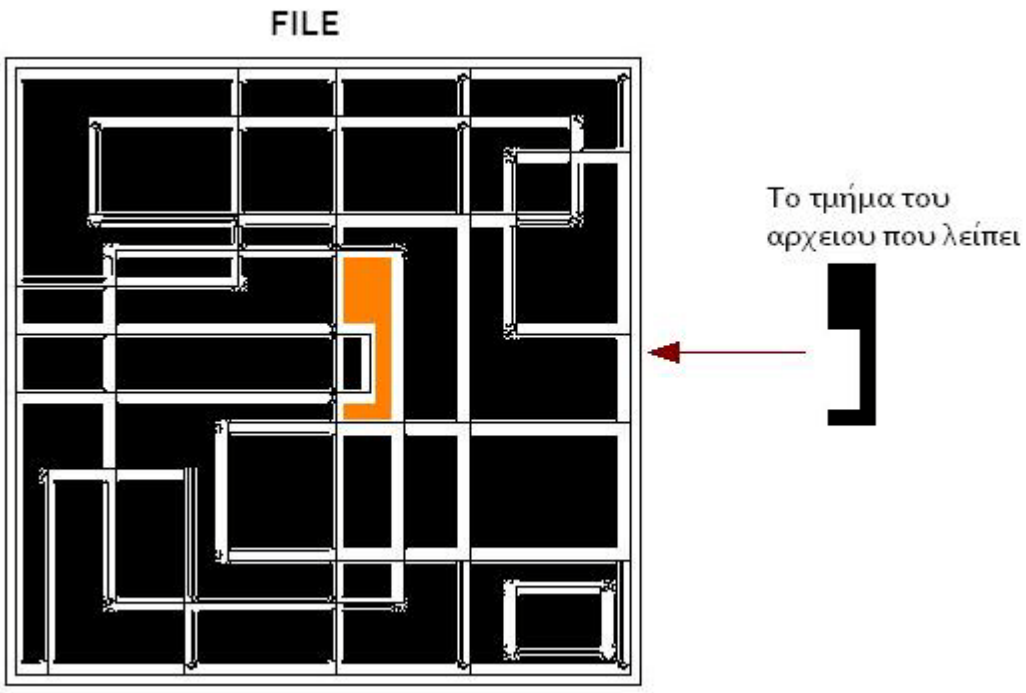
Εικόνα 22 : Εν μέρει μεταφορά δεδομένων με GridFTP



Εικόνα 23 : Παράλληλη μεταφορά δεδομένων με GridFTP

3.2.4. Τμηματική μεταφορά αρχείων

Το FTP έχει τη δυνατότητα να συνεχίσει τη μεταφορά αρχείων από ένα συγκεκριμένο σημείο εάν αυτή διακοπεί, αλλά μόνο για ένα τμήμα ενός αρχείου. Το GridFTP επιτρέπει να αποσταλεί ένα υποσύνολο ενός αρχείου. Μια τέτοια δυνατότητα είναι χρήσιμη σε εφαρμογές όπου απαιτούνται μόνο μικρά τμήματα ενός πολύ μεγάλου όγκου δεδομένων (ένα παράδειγμα είναι η επεξεργασία δεδομένων από ένα πείραμα φυσικής υψηλής ενέργειας, μια παραδοσιακή χρήση της τεχνολογίας Grid).



Εικόνα 24 : Μερική μεταφορά αρχείου

3.2.5. Ανοχή σφαλμάτων και επανεκκίνηση

Το GridFTP αποτελεί μία εφαρμογή του FTP ανεκτική στα σφάλματα που προκαλεί επανεκκίνηση για να χειριστεί την έλλειψη δικτυακής σύνδεσης και τα προβλήματα ενός διακομιστή. Μπορεί να γίνει αυτόματη επανεκκίνηση της μεταφοράς αν παρουσιαστεί κάποιο πρόβλημα.

3.2.6. Αυτόματη βελτιστοποίηση του TCP

Το TCP (Transmission Control Protocol - Πρωτόκολλο Ελέγχου Μεταφοράς) είναι ένα από τα κυριότερα πρωτόκολλα του διαδικτύου. Οι κύριοι στόχοι του είναι να επιβεβαιώνεται η αξιόπιστη αποστολή και λήψη δεδομένων καθώς επίσης να μεταφέρονται τα δεδομένα χωρίς λάθη και σε σωστή σειρά μεταξύ του στρώματος δικτύου (network layer) που είναι υπεύθυνο για τη δρομολόγηση των πακέτων προς παράδοση συμπεριλαμβανομένης και αυτής μέσω ενδιάμεσων δρομολογητών και του στρώματος εφαρμογής (application layer) το οποίο αλληλεπιδρά με τις εφαρμογές λογισμικού που συμπεριλαμβάνουν ένα επικοινωνιακό στοιχείο. Με το στρώμα αυτό

αλληλεπιδρά ο χρήστης και είναι υπεύθυνο για την παροχή υπηρεσιών, όπως είναι οι μεταφορές αρχείων, τα ηλεκτρονικά μηνύματα και τα e-mail.

Η υποκείμενη σύνδεση TCP στο FTP έχει πολλές παραμέτρους, όπως το μέγεθος των παραθύρων (window size) που καθορίζει πόσα bytes θα σταλούν μονομιάς και το μέγεθος του buffer, μιας περιοχής της μνήμης που χρησιμοποιείται για την προσωρινή αποθήκευση δεδομένων καθώς αυτά μεταφέρονται. Το GridFTP επιτρέπει την αυτόματη (ή χειροκίνητη) αλλαγή των εν λόγω ρυθμίσεων καθορίζοντας έτσι τις βέλτιστες ταχύτητες μεταφοράς και την αξιοπιστία (οι ρυθμίσεις θα πρέπει να είναι διαφορετικές κατά την μετάδοση μεγάλων αρχείων σε σχέση με μεγάλες ομάδες μικρών αρχείων, για καλύτερη απόδοση). [24]

4. Workload Management System (WMS)

Το σύστημα διαχείρισης φόρτου εργασίας WMS (Workload Management System) περιλαμβάνει ένα σύνολο τμημάτων μεσισμικού (middleware) του πλέγματος (Grid), ένα λογισμικό που αποτελείται από ένα σύνολο υπηρεσιών και επιτρέπει σε πολλαπλές διαδικασίες που εκτελούνται σε έναν ή περισσότερους Η/Υ να αλληλεπιδράσουν.

Τα τμήματα αυτά είναι αρμόδια για τη διανομή και τη διαχείριση των στόχων στους πόρους και στις πηγές του πλέγματος, κατά τέτοιο τρόπο ώστε οι εφαρμογές να εκτελούνται βολικά, αποδοτικά και αποτελεσματικά. [25]

4.1. WMProxy

Το WMProxy αποτελεί τμήμα του gLite Workload Management System (WMS) που είναι υπεύθυνο για την αποδοχή των εισερχόμενων αιτημάτων από το User Interface (UI-glite) (π.χ. την υποβολή μιας εργασίας, την απομάκρυνση εργασίας), τα οποία, εάν είναι έγκυρα (δηλαδή αν έχουν υποβληθεί σωστά), στη συνέχεια περνούν στα άλλα τμήματα/στοιχεία του WMS .

Παρέχει υποστήριξη για τις λειτουργίες ελέγχου της εργασίας μέσω Interfaces βασισμένα σε Web Services (λογισμικά σχεδιασμένα για την υποστήριξη διαλειτουργικής αλληλεπίδρασης μεταξύ μηχανημάτων/υπολογιστών σε ένα δίκτυο) και δυνατότητες όπως η διαχείρηση πολλών εργασιών και αιτημάτων προς υποβολή, υποστήριξη για σύνθετες εργασίες, καθώς και παρακολούθηση σε πραγματικό χρόνο της εξόδου της εργασίας.

Οι διάφορες εντολές αλληλεπίδρασης με το WMProxy είναι οι εξής:

- **glite-wms-job-submit**: υποβολή εργασίας σε μία WMProxy υπηρεσία. Απαιτεί σαν είσοδο ένα JDL αρχείο και επιστρέφει ένα WMS αναγνωριστικό της εργασίας.

- **glite-wms-delegate-proxy**: μπορεί ο χρήστης να αποστέλλει το proxy πιστοποιητικό του στην WMProxy υπηρεσία. Αυτό το πιστοποιητικό μπορεί μετά να χρησιμοποιηθεί για την υποβολή εργασιών στο πλέγμα.

- **glite-wms-job-list-match**: εμφάνιση των αναγνωριστικών των εργασιών που υποβλήθηκαν σε μία WMProxy υπηρεσία από τον χρήστη που εκτελεί την εντολή.

- **glite-wms-job-cancel**: αναβολή μίας ή περισσοτέρων εργασιών που είχαν υποβληθεί σε μία WMProxy υπηρεσία.

- **glite-wms-job-output**: ανάκτηση των αρχείων εξόδου μίας εργασίας, όταν η εκτέλεσή της ολοκληρωθεί.

- **glite-wms-job-perusal**: διαχείριση της λειτουργίας ανάγνωσης (perusal) μία συγκεκριμένης εργασίας:
  - Ενεργοποίηση ανάγνωσης για ένα ή περισσότερα αρχεία.
  - Ενεργοποίηση της λειτουργίας ανάγνωσης κατά την ανάκτηση ενός αρχείου.
  - Απενεργοποίηση της λειτουργίας ανάγνωσης.

- **glite-wms-job-info**: ανάκτηση πληροφοριών σχετικά με μία αποβληθείσα εργασία, το πιστοποιητικό και το JDL αρχείο που υποβάλλεται στην WMProxy υπηρεσία. [26]

4.1.1. Delegation

Το WMproxy είναι ένα σημαντικό στοιχείο του Grid υπεύθυνο για την αποδοχή των αιτημάτων που στέλνονται από το UI. Αφού δημιουργήσει ο χρήστης το πληρεξούσιο πιστοποιητικό του (proxy) θα πρέπει να το στείλει στο WMproxy. Η διαδικασία αυτή λέγεται delegation (αντιπροσωπεία) και είναι η πράξη της μεταβίβασης των δικαιωμάτων και των προνομίων σε ένα τρίτο πρόσωπο ή οντότητα. Η διαδικασία αυτή γίνεται με δύο τρόπους :

- **Ρητή αντιπροσωπεία**,

γίνεται μια φορά και στη συνέχεια, ως ξεχωριστές ενέργειες, εκτελούνται ενέργειες, όπως η υποβολή εργασίας, όπου το σύστημα αναγνωρίζει την ιδιότητα της αντιπροσωπείας όταν χρησιμοποιούμε την επιλογή/επέκταση **-d**. Μπορεί ο χρήστης να θέλει να χρησιμοποιήσει διαφορετικά πληρεξούσια για διαφορετικές εργασίες επιτρέποντας τους να έχουν διαφορετικούς ρόλους (π.χ. μέσω της χρήσης VOMS). Για να εκχωρήσει ένα τέτοιο πληρεξούσιο, θα πρέπει να δημιουργήσει ένα proxy με τις ομάδες και τους ρόλους που θέλει με την **voms-proxy-init** εντολή. Για να εξουσιοδοτήσει αυτό το πληρεξούσιο θα πρέπει να κάνει τα εξής :

glite-wms-job-delegate-proxy -d myproxy

Το "myproxy" μπορεί να αντικατασταθεί με ένα όνομα της επιλογής του χρήστη. Μπορεί να υπάρχουν πολλά διαφορετικά, πληρεξούσια με ονόματα για εξουσιοδότηση στο σύστημα WMS. Αυτό το πληρεξούσιο μπορεί να χρησιμοποιηθεί για μια συγκεκριμένη υποβολή εργασίας μέσω της επιλογή **-d**. Για παράδειγμα :

glite-wms-job-submit -d myproxy Myjob.jdl

Αυτό θα χρησιμοποιήσει το πληρεξούσιο που αναγνωρίζεται από το όνομα “myproxy” για την υποβαλλόμενη εργασία (το Myjob.jdl είναι το αρχείο που περιγράφει την εργασία και διευκρινίζεται στα επόμενα καφάλαια).

- **Αυτόματη αντιπροσωπεία,**

αντί της δημιουργίας μιας αντιπροσωπείας, θα μπορούσε να χρησιμοποιηθεί η επιλογή -a, η οποία εξουσιοδοτεί το πληρεξούσιο (proxy) αυτόματα. Με την χρήση της επιλογή -a, δεν χρειάζεται η εντολή glite-wms-job-delegate-proxy -d, αλλά πρέπει να καθοριστεί η επιλογή **-a** για κάθε χρήση των εντολών **gLite-wms-job-submit** και **gLite-wms-job-list-match**. Π.χ. :

gLite-wms-job-list-match -a Myjob.jdl

και

gLite-wms-job-submit -a Myjob.jid

Καθώς η διαδικασία της αντιπροσωπείας είναι αρκετά χρονοβόρα η πρώτη μέθοδος είναι πιο αποτελεσματική όταν εκτελούνται πολλαπλές εργασίες. [27]

4.2. Workload Manager WM

Ο πυρήνας του συστήματος διαχείρισης φόρτου εργασίας είναι ο διαχειριστής φόρτου εργασίας WM (Workload Manager), του οποίου σκοπός είναι να αποδεχθεί και να ικανοποιήσει τα αιτήματα για τη διαχείριση εργασίας που προέρχονται από τους πελάτες του.

Για μια εργασία υπολογισμού υπάρχουν δύο κύριοι τύποι αιτημάτων: υποβολή και ακύρωση. Συγκεκριμένα η έννοια του αιτήματος υποβολής είναι να περάσει η ευθύνη της εργασίας στον WM. Ο WM θα περάσει έπειτα την εργασία σε ένα κατάλληλο υπολογιστικό στοιχείο CE (computing element) για την εκτέλεση, λαμβάνοντας υπόψη τις απαιτήσεις και τις προτιμήσεις που αναφέρονται στην περιγραφή της εργασίας. [25]

4.3. Computing Element CE και Worker Node

Το Computing Element (CE) περιλαμβάνει δύο λογικά μέρη: τον gatekeeper/job-manager (διαχειριστής εργασιών και ελεγκτής εισόδου) και τους Worker Nodes WN (κόμβους εργασίας). Οι εργασίες διανέμονται στους WN μέσω λογισμικών όπως π.χ. το Portable Batch System (PBS) το οποίο προγραμματίζει την σειρά με την οποία θα εκτελεστούν οι εργασίες. Τεχνικά οι WN και το PBS εκτελούνται στο ίδιο μηχάνημα που λέγεται CE node και σε αυτό είναι συνδεδεμένοι άλλοι WN. Ένα CE παρέχει επίσης πληροφορίες για τους υπολογιστικούς του πόρους αλλά και την διαθεσιμότητα του. [28]

4.4. Match-maker

Για την επιλογή του κατάλληλου υπολογιστικού στοιχείου CE, ο WM (workload manager) χρησιμοποιεί ένα άλλο στοιχείο, το Match-maker, το οποίο συλλέγει πληροφορίες ανάλογα με την εργασία που έχει υποβληθεί και επιστρέφει στον WM την διεύθυνση του κατάλληλου CE. [29]

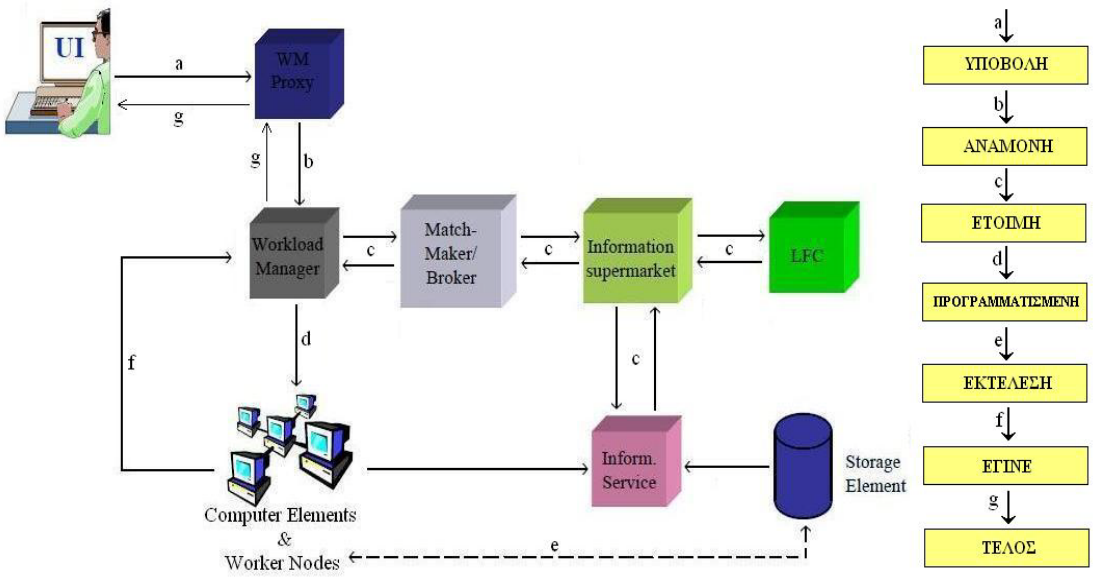
4.5. Information Supermarket

Το Information Supermarket (ISM) συλλέγει αντίστοιχα τα κατάλληλα δεδομένα από το LFC (LGC File Catalog, μία υπηρεσία που παρέχει αρχεία και διευθύνσεις αντιγράφων αρχείων) και την Υπηρεσία Πληροφοριών (IS, Information Service) η οποία παρέχει πληροφορίες σχετικά με πόρους του πλέγματος, όπως αποθηκευτικά στοιχεία (storage elements SE) και την κατάστασή τους. Οι πληροφορίες αυτές είναι απαραίτητες για τη λειτουργία του όλου πλέγματος, καθώς μέσω της IS ανακαλύπτονται οι πόροι.

4.6. Storage Elements SE

Ένα αποθηκευτικό στοιχείο SE προσφέρει ομοιόμορφη πρόσβαση σε αποθηκευμένους πόρους. Θα μπορούσε να είναι απλά ένας σκληρός δίσκος σε κάποιον διακομιστή, μεγάλες συστοιχίες δίσκων ή Mass Storage System (μαζικό σύστημα αποθήκευσης), όπως είναι το dCache και το Castor.

Η απόφαση για τον πόρο που πρέπει να χρησιμοποιηθεί είναι η έκβαση μιας διαδικασίας αντιστοίχισης μεταξύ των αιτημάτων υποβολής και των διαθέσιμων πόρων.



Εικόνα 25 : Workload Management System (WMS)

[32]

5. Command Line Interface (CLI)

Μια διεπαφή εντολών γραμμής (CLI) είναι ένας μηχανισμός για την αλληλεπίδραση με ένα λειτουργικό σύστημα υπολογιστή ή με κάποιο λογισμικό πληκτρολογώντας εντολές για την εκτέλεση ειδικών εργασιών (tasks). Στη συνέχεια ένας διερμηνέας της γραμμής εντολών λαμβάνει, αναλύει, και εκτελεί την εντολή που πληκτρολογήθηκε. Ο διερμηνέας της γραμμής εντολών μπορεί να εκτελεστεί σε ένα τερματικό κειμένου ή σε ένα παράθυρο ενός εξομοιωτή τερματικού όπως είναι το PuTTY. Μετά την ολοκλήρωση της εντολής, η έξοδος επιστρέφει συνήθως στο χρήστη, με τη μορφή γραμμών κειμένου στο CLI. Η έξοδος αυτή μπορεί να είναι μια απάντηση εάν η εντολή ήταν μια ερώτηση ή αλλιώς μια περίληψη της εργασίας.[33]

5.1. Ονομασίες αρχείων και ο εντοπισμός τους

Οι χρήστες και οι εφαρμογές μπορούν να εντοπίσουν τα αρχεία ή τα αντίγραφά τους στο Grid. Ένα αρχείο στο Grid μπορεί να εμφανίζεται με διαφορετικά ονόματα: **Grid Unique IDentifier (GUID)**, **Logical File Name (LFN)**, **Storage URL (SURL)** και **Transport URL (TURL)**. Οι αντιστοιχίες μεταξύ τους φυλάσσονται σε μια υπηρεσία που ονομάζεται File Catalog, ενώ τα αρχεία τους είναι αποθηκευμένα στα Storage Elements. Στην ιδανική περίπτωση, οι χρήστες δεν χρειάζεται να γνωρίζουν που βρίσκεται ένας φάκελος, καθώς χρησιμοποιούν λογικά ονόματα για τα αρχεία που η υπηρεσία Data Management Service θα χρησιμοποιήσει για να τα εντοπίσει και να αποκτήσει πρόσβαση σε αυτά.

- Το GUID αποδίδεται σε ένα αρχείο την πρώτη φορά που αυτό καταχωρείται στο Grid και είναι μοναδικό (ένα μοναδικό string). Είναι της μορφής:

  guid: 93bd772a-b282-4332-a0c5-c79e99fc2e9c

- Το LFN είναι ένας άλλος τρόπος αναφοράς σε ένα αρχείο και είναι της μορφής:
  lfn: oti_onoma_thelw

Σε περίπτωση που το LCG File Catalog χρησιμοποιείται, τα LFNs είναι οργανωμένα με ιεραρχική δομή που μοιάζει με κατάλογο, και θα έχουν την παρακάτω μορφή:

lfn: /grid/see/katalogos/arxeio

όπου **see** ο εικονικός οργανισμός στον οποίο είμαι εγγεγραμμένος, katalogos ο φάκελος μου και arxeio το αρχείο μου.

- Το SURL γνωστό και ως Physical File Name (PFN) χρησιμοποιείται για τον εντοπισμό του αντιγράφου ενός αρχείου σε ένα Storage Element και είναι της μορφής:

  sfn://se01.athena.hellasgrid.gr/3596e86f-c402-11d7-a6b0-978v3r76c8

όπου se01.athena.hellasgrid.gr είναι το storage element (01 στην περίπτωση αυτή) με το hostname (όνομα υποδοχέα υπολογιστή/μηχανήματος) και 3596e86f-c402-11d7-a6b0-978v3r76c8 το string που αντιστοιχεί στο αρχείο.

ή

srm://se01.athena.hellasgrid.gr/pnfs/athena.hellasgrid.gr/data/see/generated/201009-17/file4721796e-8606-4831-9386-1dd7b8f8bb62

όπου se01.athena.hellasgrid.gr το storage element με το hostname και /pnfs/athena.hellasgrid.gr/data/see/generated/201009-17/file4721796e-8606-4831-9386-1dd7b8f8bb62 η διεύθυνση/διαδρομή του αρχείου.

- Το TURL είναι μία έγκυρη διεύθυνση URI με τις κατάλληλες πληροφορίες για την πρόσβαση σε ένα αρχείο σε κάποιο αποθηκευτικό στοιχείο SE:
  <protocol>://<some_string>

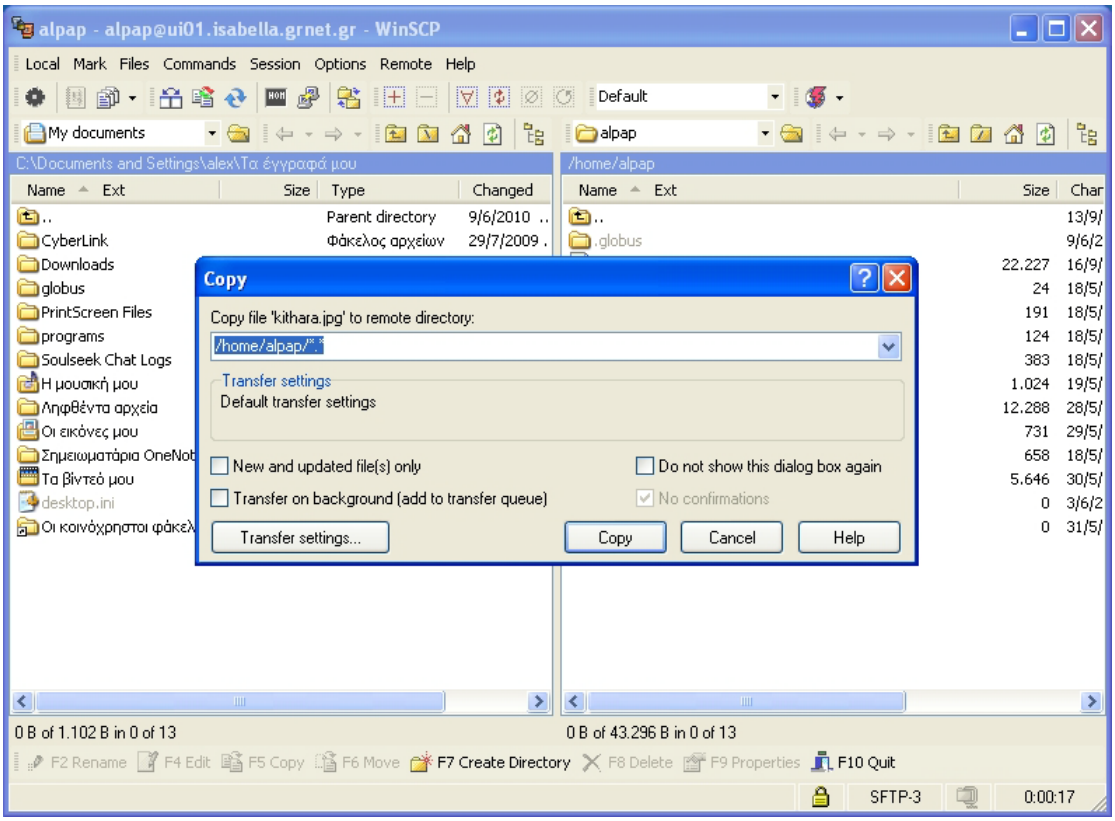
Π.χ. gsiftp: //tbed0101.cern.ch/flatfiles/SE00/dteam/93bd772a-b282-4332

όπου το gsiftp είναι ένα πρωτόκολλο μεταφοράς αρχείων που υποστηρίζεται από το storage element ,και το υπόλοιπο string μετά το // που αναγνωρίζει μοναδικά το αρχείο στο SE, αλλά συνήθως είναι της μορφής :

<protocol>://<se_hostname>:<port>/path [34]

5.2. Παράδειγμα μεταφοράς αρχείων

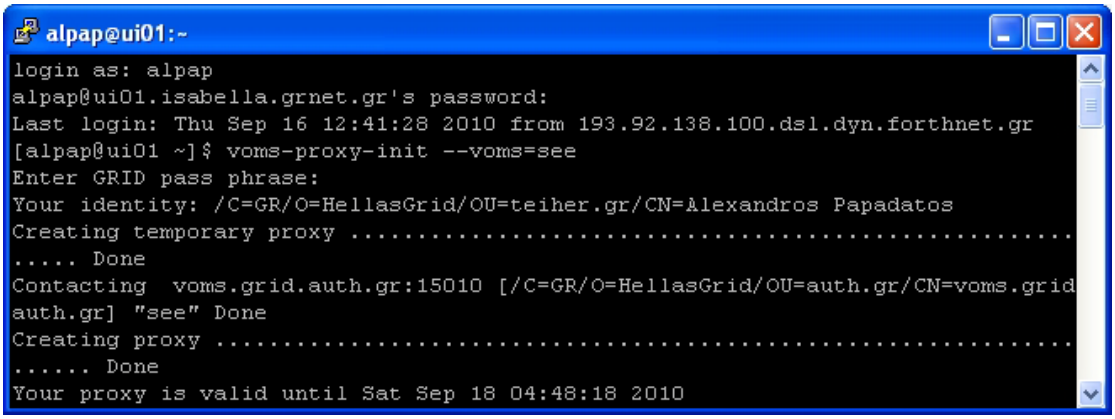
Πριν την χρήση εντολών CLI μέσω του προγράμματος puTTY, θα πρέπει να μεταφερθεί το αρχείο που ο χρήστης θέλει να αντιγράψει στο πλέγμα, στον λογαριασμό του στο UI. Αυτό γίνεται εύκολα με το πρόγραμμα WinSCP.



Εικόνα 26 : Αντιγραφή αρχείου στον UI λογαριασμό

Στην περίπτωση αυτή το αρχείο προς αντιγραφή είναι το " **kithara.jpg** " το οποίο μεταφέρεται στον λογαριασμό alpap όπως φαίνεται στην παραπάνω εικόνα.

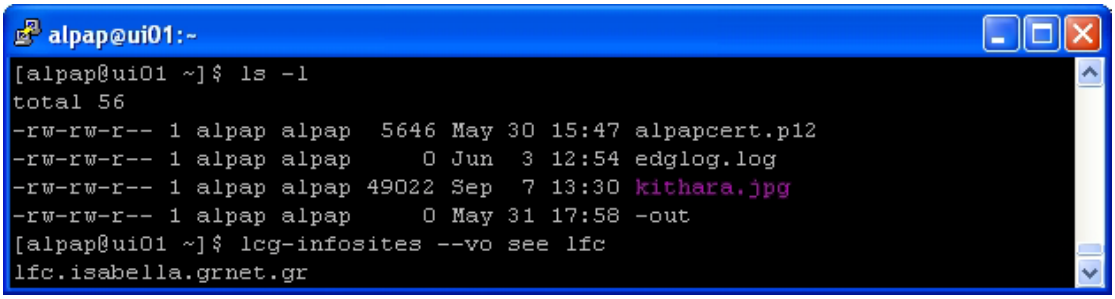
Ακολουθεί η δημιουργία ενός proxy certificate ώστε να μπορεί να υποβληθεί η εργασία της αντιγραφής.



Εικόνα 27 : Δημιουργία proxy πιστοποιητικού

Αφού επαληθευτεί η ύπαρξη του αρχείου στον κατάλογο του χρήστη με την εντολή " **ls -l** " θα πρέπει έπειτα να βρεθεί ο κατάλογος LFC που αντιστοιχεί στην VO του χρήστη και στην περίπτωση αυτή, στην **SEE**. Η εντολή είναι η εξής :

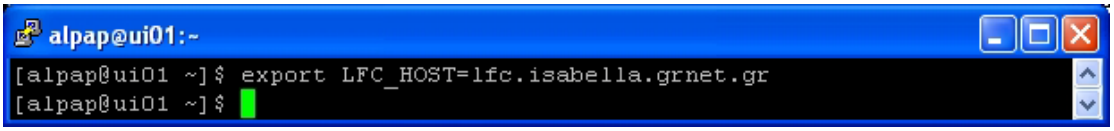
lcg-infosites --vo see lfc



Εικόνα 28 : Εμφάνιση αρχείων με την εντολή ls -l

και ο κατάλογος είναι ο **lfc.isabella.grnet.gr** τον οποίο πρέπει να καθορίσουμε ως Host, δηλαδή ως κατάλογο υποδοχής, πληκτρολογώντας :

export LFC_HOST=lfc.isabella.grnet.gr



Εικόνα 29 : Καθορισμός hostname

Η εντολή για την αντιγραφή του αρχείου στο πλέγμα είναι η ακόλουθη:

lcg-cr -v -l /grid/see/kithara file:/home/alpap/kithara.jpg

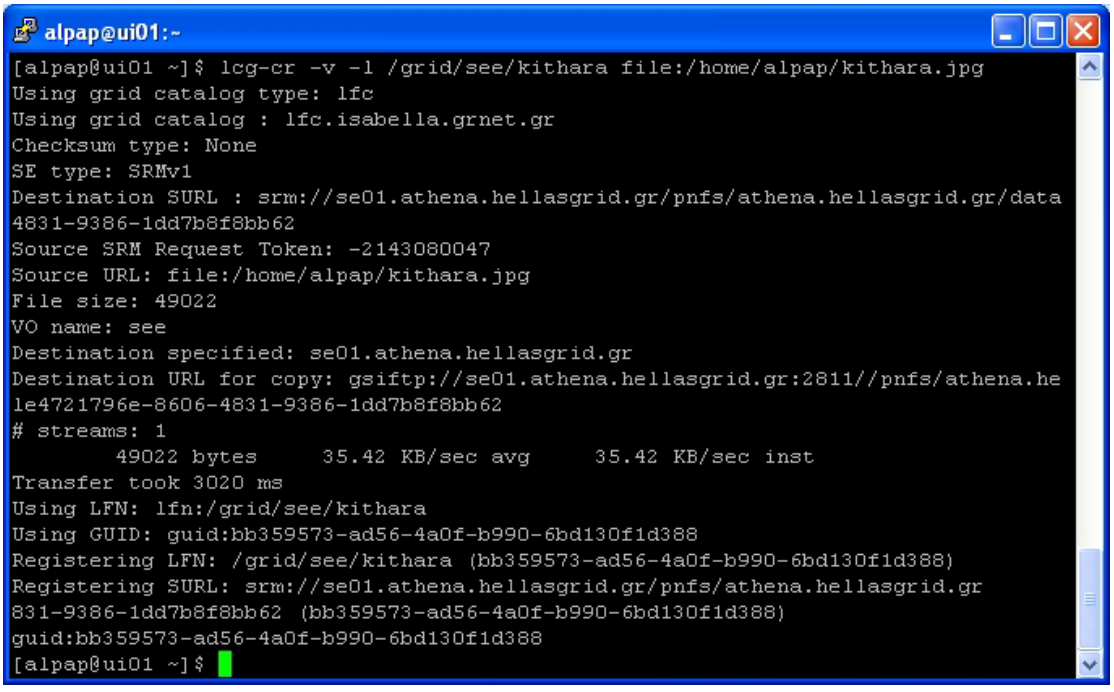
όπου:

-cr σημαίνει αντιγραφή και καταχώρηση (copy and register).

-v (verbose=φλύαρος), η έξοδος με αυτή την μεταβλητή θα είναι πιο αναλυτική.

-l (lfn), δείχνει πώς θα προσδιοριστεί το αρχείο.

Αντιγράφεται λοιπόν το αρχείο από το τοπικό σύστημα αρχείων του χρήστη (/home/alpap/kithara.jpg) στο πλέγμα (/grid/see/kithara).



Εικόνα 30 : Αντιγραφή ενός αρχείου στο πλέγμα

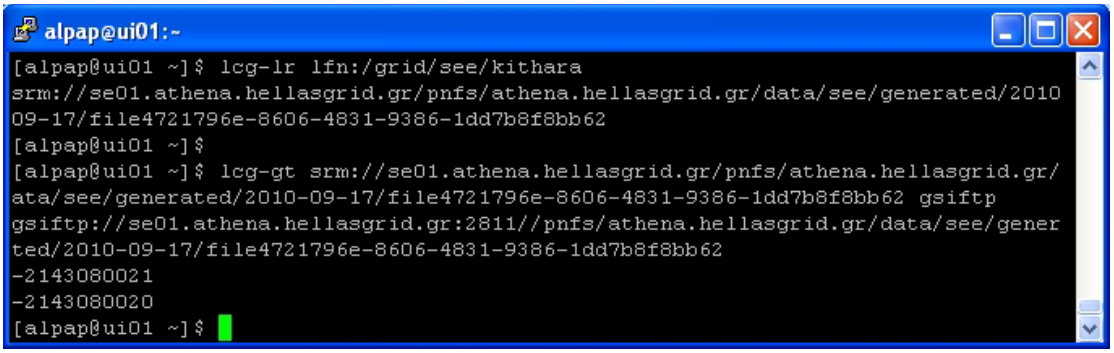
Εάν δεν προσδιορίζεται το Storage Element (SE) στο οποίο θέλουμε να αποθηκευτεί το αρχείο επιλέγεται το κοντινότερο. Ο όρος "κοντά" στο πλέγμα μπορεί να έχει διάφορες σημασίες συμπεριλαμβανομένης της φυσικής απόστασης και της ταχύτητας πρόσβασης στα δεδομένα (διαθέσιμο εύρος ζώνης). Το πιο κοντινό SE σε ένα CE ορίζεται από τον διαχειριστή του CE. Στην περίπτωση αυτή το αρχείο αποθηκεύτηκε στο se01.athena.hellasgrid.gr.

Το αρχείο έχει πλέον καταχωρηθεί στο Grid και παράλληλα έχει δημιουργηθεί το string **GUID: bb359573-ad56-4a0f-b990-6bd130f1d388** με το οποίο μπορεί εύκολα να εντοπισθεί μέσα στο πλέγμα.

Ξέροντας το lfn και το guid του αρχείου ο χρήστης μπορεί να βρει τα surl και turl του πληκτρολογώντας τις αντίστοιχες εντολές:

lcg-lr lfn:/grid/see/kithara για το surl (στο surl φαίνεται και σε ποιο SE έχει αποθηκευτεί το αρχείο, στην περίπτωση αυτή στο se01.athena.hellasgrid.gr)

και **lcg-gt srm** , για το turl (όπου srm η έξοδος από την παραπάνω εντολή)



Εικόνα 31 : Εύρεση surl και turl αρχείου στο πλέγμα

Ο χρήστης μπορεί ακόμα να ελέγξει πληροφορίες για κάποιο αρχείο (π.χ. τον χρήστη που το ανέβασε) που έχει καταχωρηθεί στο πλέγμα με την εντολή:

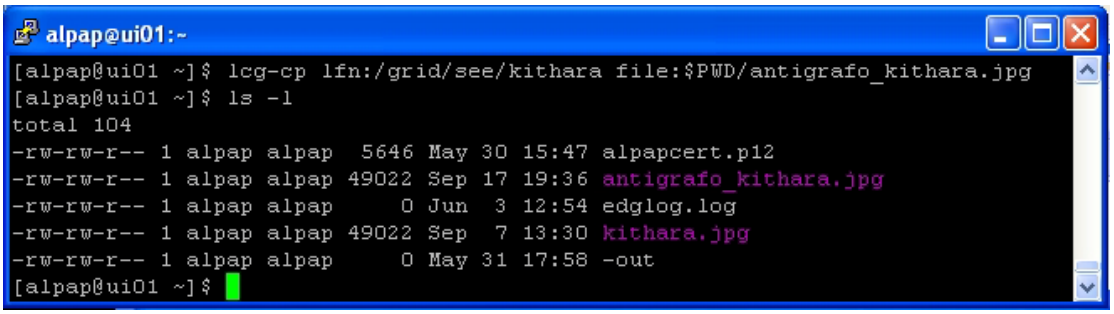
lfc-getacl /grid/see/kithara

Εικόνα 32 : Εμφάνιση πληροφοριών αρχείου

Η αντίστροφη διαδικασία για την αντιγραφή του αρχείου από το grid στον λογαριασμό UI του χρήστη είναι μία εύκολη διαδικασία και γίνεται με την εξής εντολή:

lcg-cp lfn:grid/see/kithara file:$PWD/antigrafo_kithara.jpg

Το αρχείο αντιγράφεται από το grid δίνοντας το lfn του, στο τοπικό σύστημα αρχείων του χρήστη με όνομα που θα καθορίσει αυτός, π.χ. antigrafo_kithara.jpg . Πληκτρολογώντας έπειτα την εντολή " **ls -l** " επαληθεύει ο χρήστης την αντιγραφή.



Εικόνα 33 : Αντιγραφή αρχείου από το πλέγμα στο τοπικό σύστημα του χρήστη

6. Job Description Language (JDL)

Η γλώσσα περιγραφής εργασίας JDL είναι μια υψηλού επιπέδου γλώσσα με την οποία ένας χρήστης μπορεί να περιγράψει μια εργασία ή ένα σύνολο εργασιών έτσι ώστε να μπορεί να τις υποβάλλει στο πλέγμα . Όντας η JDL μια επεκτάσιμη γλώσσα, ο χρήστης επιτρέπεται να χρησιμοποιήσει οποιαδήποτε ιδιότητα για την περιγραφή ενός αιτήματος χωρίς να υποστεί τα λάθη από τον parser (κατατμητής) JDL.

Στην τεχνολογία υπολογιστών, ένας κατατμητής είναι ένα πρόγραμμα, συνήθως μέρος ενός μεταγλωττιστή, που λαμβάνει στην είσοδό του υπό μορφή προγράμματος πηγής, διαδοχικές οδηγίες, διαδραστικές εντολές, ετικέτες σήμανσης, ή κάποια άλλη καθορισμένη ιδιότητα και τις διαμερίζει σε τμήματα (π.χ. τα ουσιαστικά (αντικείμενα), τα ρήματα (μέθοδοι), και τις ιδιότητες ή τις επιλογές τους) που μπορούν έπειτα να ρυθμιστούν με άλλο προγραμματισμό (π.χ. άλλα στοιχεία σε έναν μεταγλωττιστή). Ο κατατμητής ελέγχει ότι όλες οι είσοδοι έχουν τη σωστή σύνταξη και χρησιμοποιώντας έναν λεξιλογικό αναλυτή χτίζει μία δομή, χωρισμένη σε τμήματα. [37]

Εντούτοις, μόνο ένα ορισμένο σύνολο ιδιοτήτων, που θα τις αναφέρουμε ως «υποστηριζόμενες ιδιότητες» από τώρα και στο εξής, λαμβάνονται υπόψη από τα τμήματα συστημάτων διαχείρισης φόρτου εργασίας (WMS) προκειμένου να σχεδιαστεί και να υποβληθεί μία ή περισσότερες εργασίες ενός σύνθετου αιτήματος.

Οι ιδιότητες JDL αντιπροσωπεύουν τις συγκεκριμένες πληροφορίες του αιτήματος και διευκρινίζουν κατά κάποιο τρόπο τις ενέργειες που πρέπει να εκτελεσθούν από το WMS για να σχεδιάσουν την εργασία ή τις εργασίες ενός σύνθετου αιτήματος. Μερικές από τις ιδιότητες στο JDL είναι **υποχρεωτικές**. Εάν ο χρήστης δεν τις διευκρινίζει, το WMS δεν μπορεί να χειριστεί το αίτημα. Για τις άλλες ιδιότητες το σύστημα μπορεί να βρει μια προκαθορισμένη τιμή εάν είναι απαραίτητες για την επεξεργασία του αιτήματος. [38]

6.1. Attributes (Ιδιότητες)

Πριν αρχίσουμε τη λεπτομερή περιγραφή ιδιοτήτων να σημειωθεί ότι μια περιγραφή αιτήματος συντίθεται από τις καταχωρήσεις που είναι σειρές χαρακτήρων (strings) και έχουν τη μορφή :

Attributes = Expressions;
(Ιδιότητες = Εκφράσεις)

Ολοκληρώνεται από το χαρακτήρα του ερωτηματικού (;) και ολόκληρη η περιγραφή εργασίας πρέπει να συμπεριληφθεί σε αγκύλες, δηλ. [<job description>]. Η λήξη με το ερωτηματικό δεν είναι υποχρεωτική για την τελευταία ιδιότητα πριν από το κλείσιμο με την αγκύλη] .

Οι εκφράσεις ιδιοτήτων μπορούν να πιάσουν περισσότερες από μία γραμμές υπό τον όρο ότι το ερωτηματικό τίθεται μόνο στο τέλος ολόκληρης της έκφρασης. Τα σχόλια πρέπει να έχουν μία δίεση (#) ή μια διπλή κάθετο (//) στην αρχή κάθε γραμμής. Τα σχόλια που εκτείνονται σε πολλαπλές γραμμές ή που είναι τοποθετημένα μετά από την άνω τελεία που δείχνει το τέλος της προδιαγραφής ιδιοτήτων, μπορούν να διευκρινιστούν εσωκλείοντας το κείμενο μεταξύ ‘’/*’’ και ‘’*/’’.

Προσοχή: Η JDL είναι ευαίσθητη στους χαρακτήρες κενού και Tab και δεν θα πρέπει να τοποθετούνται μετά το ερωτηματικό στο τέλος κάθε γραμμής.

Επιπλέον αξίζει να σημειωθεί ότι οι απαιτήσεις και η κατηγορία των εκφράσεων των ιδιοτήτων που αξιολογούνται από το WM κατά τη διάρκεια της διαδικασίας αντιστοιχίσεων για την επιλογή των υπολογιστικών στοιχείων CEs όπου θα σταλεί η εργασία, χτίζονται χρησιμοποιώντας τις ιδιότητες των πόρων, που αντιπροσωπεύουν τα χαρακτηριστικά και την κατάστασή τους στο πλέγμα.

6.2. Περιγραφή Ιδιοτήτων των Εργασιών

Αυτό το τμήμα περιέχει τη λεπτομερή περιγραφή των σημαντικότερων ιδιοτήτων JDL που μπορούν να διευκρινιστούν για την περιγραφή των αιτημάτων μιας εργασίας.

6.2.1. Type

Η ιδιότητα Type είναι ένα string (μία ακολουθία χαρακτήρων) που αντιπροσωπεύει τον τύπο του αιτήματος που περιγράφεται από το JDL.

Type = “Job”; (Εργασία)

Οι πιθανές τιμές είναι:

− Job

− DAG

− Collection

- Job (εργασία)

απλή εργασία

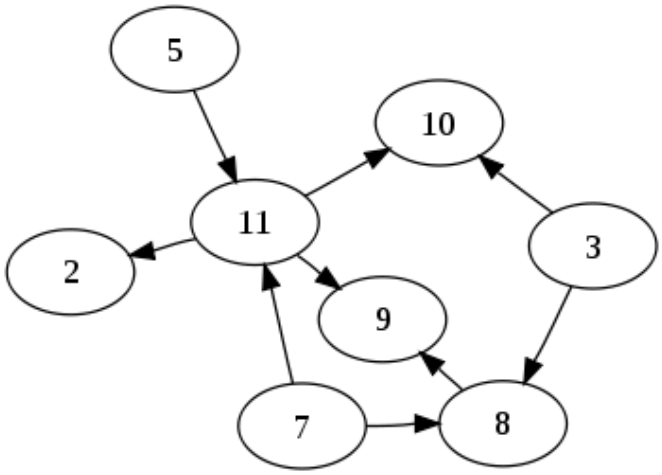
- DAG:

Directed Acyclic Graphs,

Κατευθυνόμενα Μη-Κυκλικά Γραφήματα:

Ένα γράφημα στο οποίο

καμία διαδρομή δεν ξεκινά και τελειώνει στον ίδιο κόμβο.



Εικόνα 34 : Οι διαδρομές μιας εργασίας DAG

- Collection (συλλογή), Ένα σύνολο ανεξάρτητων εργασιών.

Παρόλο που τα DAGs και οι συλλογές αντιπροσωπεύουν σύνολα εργασιών, περιγράφονται μέσω μιας ενιαίας περιγραφής JDL και μπορούν ως εκ τούτου να υποβληθούν μονομιάς στο WMS. Επιπλέον πάνω στην υποβολή τέτοιου είδους

αιτημάτων, το WMS θα παρέχει στο χρήστη ένα συλλογικό προσδιοριστή που θα του επιτρέψει να παρακολουθεί και να ελέγχει ολόκληρο το σύνολο των εργασιών μέσω ενός ενιαίου χειρισμού.

Οι δομές JDL για τα DAGs και τις συλλογές είναι κατά κάποιο τρόπο παρόμοιες, αλλά έχουν κρατηθεί ως ξεχωριστοί τύποι αιτήματος δεδομένου ότι αντιπροσωπεύουν σύνολα εργασιών με διαφορετικό είδος αμοιβαίας σχέσης.

Εάν αυτή η ιδιότητα δεν διευκρινίζεται σωστά (κάτι που βασίζεται στην χρήση των κεφαλαίων και πεζών γραμμάτων) ή ακόμα καθόλου στην περιγραφή JDL, το WMS θα την θέσει στο «Job».

– Default**:** “Job” (Προεπιλογή)

6.2.2. Job Type (Τύπος Εργασίας)

Η ιδιότητα JobType είναι ένα string (ακολουθία χαρακτήρων) που αντιπροσωπεύει τον τύπο της εργασίας που περιγράφεται από το JDL και έχει νόημα μόνο όταν η ιδιότητα του τύπου (Type) ισοδυναμεί με την «εργασία» (“Job”) και δεν ισχύει για τα DAGs και τις συλλογές:

π.χ.:

JobType = “Interactive”; (διαλογική)

Οι πιθανές τιμές είναι:

- Normal
- Interactive
- MPICH
- Checkpointable
- Partitionable

- Parametric

Να σημειωθεί ότι μια partitionable εργασία πρέπει να είναι επίσης checkpointable.

- Normal (κανονική)

Μία απλή εργασία batch: ένα σχεδιασμένο πρόγραμμα που εκτελείται χωρίς την παρέμβαση χρηστών.

- Interactive (διαλογική)

Μία εργασία της οποίας τα *standard streams* [καθιερωμένα ρεύματα τα οποία είναι τα προ-συνδεμένα κανάλια εισόδου και εξόδου μεταξύ ενός προγράμματος υπολογιστή και της διεπαφής χρήστη (όπως ένα τερματικό κειμένου) όταν ξεκινά να εκτελείται.] διαβιβάζονται στον υποβάλλοντα πελάτη.

- MPICH

Είναι μια διαθέσιμη, φορητή εφαρμογή MPI (Message Passing Interface), ένα πρότυπο για την διαβίβαση μηνυμάτων μεταξύ εφαρμογών διανομής μνήμης που χρησιμοποιούνται στον παράλληλο προγραμματισμό (parallel processing).

- Partitionable

Μία εργασία που μπορεί να θεωρηθεί ως ένα σύνολο ανεξάρτητων βημάτων ή επαναλήψεων, δηλ. ένα σύνολο ανεξάρτητων υπο-εργασιών, όπου η κάθε μία εκτελεί ένα βήμα ή ένα υποσύνολο των βημάτων, και που μπορεί να εκτελεσθεί παράλληλα.

- Checkpointable

Μία εργασία ικανή να αποθηκεύσει την κατάστασή της, έτσι ώστε η εκτέλεσή της να

μπορεί να ανασταλεί, και να επαναληφθεί αργότερα, ξεκινώντας από το ίδιο σημείο όπου σταμάτησε αρχικά.

- Parametric (Παραμετρική)

Μία εργασία της οποίας το JDL περιέχει παραμετρικές ιδιότητες (π.χ. μεταβλητές, StdInput κ.λπ.) των οποίων οι τιμές μπορούν να ποικίλουν προκειμένου να ληφθεί η υποβολή διαφορετικών περιπτώσεων παρόμοιων εργασιών που διαφέρουν μόνο στην τιμή των ιδιοτήτων με παραμέτρους.

Ανάλογα στα DAGs και τις συλλογές, για τους τύπους εργασίας των οποίων η υποβολή οδηγεί στην δημιουργία ενός συνόλου εργασιών (δηλ. Partitionable και Parametric εργασίες) το WMS θα παράσχει στο χρήστη ένα συλλογικό προσδιοριστή που θα επιτρέπει τη παρακολούθηση και τον έλεγχο ολόκληρου του συνόλου εργασιών μέσω ενός ενιαίου χειρισμού.

Η τιμή της ιδιότητας αυτής δεν διακρίνεται από κεφαλαίους ή πεζούς χαρακτήρες. Εάν η ιδιότητα δεν διευκρινίζεται στο JDL, το WMS θα την θέσει στο «κανονικό» (“normal”) ως προεπιλογή.

6.2.3. Executable (εκτελέσιμο)

Η ιδιότητα “εκτελέσιμου” είναι ένα string που αντιπροσωπεύει το όνομα της εκτελέσιμης εντολής ή του εκτελέσιμου προγράμματος. Ο χρήστης μπορεί να διευκρινίσει ένα εκτελέσιμο - executable που βρίσκεται ήδη σε έναν απομακρυσμένο CE WN (Computer Element Worker Node).

Σε αυτήν την περίπτωση ολόκληρη η ιδανική διαδρομή/διεύθυνση, ενδεχομένως συμπεριλαμβανομένων των μεταβλητών περιβάλλοντος που αναφέρονται σε αυτό το αρχείο πρέπει να διευκρινιστεί., π.χ.:

Executable = “/usr/local/ java/j2sdk1.4.0_01/bin/java”;

ή ισοδύναμα

Executable = “$JAVA_HOME/bin/java”; ,

εδώ ζητείται να εκτελεστεί η γλώσσα προγραμματισμού java ακολουθώντας την συγκεκριμένη διαδρομή η οποία έχει οριστεί από το Πρότυπο Ιεραρχίας Αρχείων (Filesystem Hierarchy Standard (FHS)) που ορίζει τους κύριους καταλόγους και το περιεχόμενό τους στα λειτουργικά συστήματα Linux (δηλ. την $JAVA_HOME ή /usr/local/ java/j2sdk1.4.0_01).

Η άλλη δυνατότητα είναι να παρασχεθεί είτε ένα διαθέσιμο εκτελέσιμο αρχείο - executable από το τοπικό σύστημα αρχείων είτε ένα executable που βρίσκεται σε έναν απομακρυσμένο εξυπηρετητή gridFTP προσιτό από το χρήστη. Και στις δύο περιπτώσεις το εκτελέσιμο αρχείο θα μεταφερθεί από την αρχική του θέση στο υπολογιστικό στοιχείο WN. Και στις δύο περιπτώσεις μόνο το όνομα του αρχείου πρέπει να διευκρινιστεί ως εκτελέσιμο. Το URI (Uniform Resource Identifier, μια σειρά χαρακτήρων που χρησιμοποιούνται για να προσδιορίσουν ένα όνομα ή έναν πόρο στο διαδίκτυο) του εκτελέσιμου πρέπει να γραφτεί έπειτα στην έκφραση ιδιοτήτων InputSandbox που θα δούμε παρακάτω για να το κάνει να μεταφερθεί. Π.χ. αντίστοιχα:

Executable = “script.sh”;

InputSandbox = {“file:///home/alpap/fakelos2/script.sh”};

(Στο InputSandbox γράφουμε την διαδρομή του αρχείου στο τοπικό σύστημα αρχείων.)

ή

Executable = “script.sh”;

InputSandbox = {“gsiftp://neo.datamat.it:5678/tmp/script.sh”};

(όπου gsiftp://neo.datamat.it:5678/tmp/script.sh είναι η διαδρομή για τον απομακρυ-

σμένο gridftp εξυπηρετητή)

Επίσης, για να υπάρξει συμβατότητα με τις προηγούμενες εκδόσεις της JDL γλώσσας, η παρακάτω περιγραφή :

Executable = “script.sh”;
InputSandbox = {“/home/alpap/φάκελος2/script.sh”};

είναι αποδεκτή και ερμηνεύεται όπως και στην πρώτη περίπτωση, δηλ. το εκτελέσιμο αρχείο είναι διαθέσιμο στο τοπικό σύστημα αρχείων.

Είναι σημαντικό να παρατηρηθεί ότι εάν η εργασία χρειάζεται για την εκτέλεση της μερικές γραμμές εντολών, πρέπει να διευκρινιστούν μέσω των ιδιοτήτων Arguments (επιχειρήματα). Αυτή η ιδιότητα είναι υποχρεωτική.

6.2.4. Arguments (Ορίσματα)

Η ιδιότητα arguments είναι μια ακολουθία χαρακτήρων (string) που περιέχει όλα τα ορίσματα για την γραμμή εντολής της εργασίας.

Π.χ. ένα εκτελέσιμο αρχείο (executable) άθροισης που πρέπει να αρχίσει ως εξής :

$ sum N1 N2 –out result.out

Καθορίζεται από :

Executable = “sum”;
Arguments = “N1 N2 –out result.out”;

Εάν χρειάζεται να χρησιμοποιηθεί ένα συγκεκριμένο string (quoted string π.χ. η διπλή

απόστροφος “) μέσα στα ορίσματα θα πρέπει το σχετικό απόσπασμα να καθοριστεί με τον χαρακτήρα ****. Π.χ. κατά την περιγραφή μιας εργασίας όπως :

$ grep –i “my name” *.txt

(ψάχνει όλα τα txt αρχεία και εμφανίζει όλες τις γραμμές που περιέχουν το “my name” ανεξαρτήτως πεζών ή κεφαλαίων (-i))

θα πρέπει να διευκρινίσετε:

Executable = “/bin/grep”;
Arguments = “-i \”my name\” *.txt”;

Ανάλογα, εάν η εργασία παίρνει ως όρισμα ένα string που περιέχει έναν ιδιαίτερο χαρακτήρα (π.χ. η εργασία είναι μια εντολή *tail* που εκδίδεται σε ένα αρχείο του οποίου το όνομα περιέχει το χαρακτήρα «&», για παράδειγμα file1&file2), δεδομένου ότι στη γραμμή εντολών *shell* (μία άλλη γλώσσα προγραμματισμού) θα έπρεπε να γράψετε:

$ tail –f file1\&file2

Η εντολή “ tail ” διαβάζει τις τελευταίες γραμμές οποιουδήποτε κειμένου που δίνεται ως input και τις γράφει στην έξοδο standard output (που, εξ ορισμού, είναι η οθόνη). [39]

Στην JDL γλώσσα θα πρέπει να γράψετε :

Executable = “/usr/bin/tail”;
Arguments = “-f file1\\\&file2”;

δηλαδή ένα **** για κάθε ιδιαίτερο χαρακτήρα.

Γενικά, πρόσθετοι χαρακτήρες όπως **& , |, > , <** επιτρέπονται μόνο όταν καθορίζονται μέσα σε ένα quoted string όπως το " ή όταν προηγούνται του τριπλού χαρακτήρα ****. Ο χαρακτήρας ` δεν μπορεί να καθοριστεί στο JDL.

6.2.5. StdInput

Η ιδιότητα StdInput είναι ένα string που αντιπροσωπεύει την τυποποιημένη /πρότυπη είσοδο της εργασίας δηλαδή το "standard input". Αυτό σημαίνει ότι η εργασία εκτελείται ως εξής:

$ job < <standard input file>

Μπορεί να είναι μια απόλυτη διαδρομή/διεύθυνση ενδεχομένως συμπεριλαμβανομένων των μεταβλητών περιβάλλοντος (τα λεγόμενα wild cards, τα οποία είναι ειδικοί χαρακτήρες που αντιπροσωπεύουν έναν ή περισσότερους χαρακτήρες όπως το * και το ?, δεν επιτρέπονται), δηλ. είναι ήδη διαθέσιμο στο CE, π.χ. :

StdInput = "/var/tmp/jobInput";

ή απλά ένα όνομα αρχείου δηλαδή :

StdInput = "myjobInput";

και αυτό σημαίνει ότι το αρχείο πρέπει να γίνει διαθέσιμο στο WN όπου η εργασία εκτελείται. Επομένως το standard input file πρέπει να προστεθεί στον φάκελο αρχείων InputSandbox έτσι ώστε να αντιγραφεί στον κόμβο WMS και να μεταφορτωθεί (download) έπειτα στο WN από το χειρόγραφο WMS JobWrapper. Πριν αρχίσουν να εκτελούνται οι υποβληθείσες εργασίες, ένα script που ονομάζεται JobWrapper προετοιμάζει το περιβάλλον για την εργασία, μεταφέροντας στο WN

πρόσθετα αρχεία με το InputSandbox. Οι ίδιοι κανόνες που περιγράφονται για τις εκτελέσιμες ιδιότητες ισχύουν και για την StdInput.

6.2.6. StdOutput

Η ιδιότητα StdOutput είναι ένα string που αντιπροσωπεύει το όνομα του αρχείου στο οποίο αποθηκεύεται η πρότυπη έξοδος (standard output) στο Worker Node, WN.

Ο χρήστης μπορεί είτε να ορίσει ένα όνομα αρχείου ή μια σχετική διαδρομή (σε σχέση με τον φάκελο στον WN, των εργασιών που εκτελούνται), π.χ.:

StdOutput = “myjobOutput”;
StdOutput = “event1/myjobOutput”;

Τα wild cards δεν επιτρέπονται. Η τιμή που προβλέπεται για το StdError μπορεί να είναι η ίδια με εκείνη της ιδιότητας StdOutput: αυτό σημαίνει ότι οι δύο τυποποιημένες ροές/streams της εργασίας, αποθηκεύονται στο ίδιο αρχείο.

Για να πάει πίσω αυτό το αρχείο στο μηχάνημα που υπέβαλε την εργασία θα πρέπει ο χρήστης να συμπεριλάβει το όνομα του αρχείου στο OutputSandbox και να κάνει χρήση, π.χ. της εντολής : gLite-wms-job-output <όνομα αρχείου> .

Διαφορετικά ο χρήστης μπορεί να επιλέξει να βάλει το αρχείο σε έναν GridFTP server (εξυπηρετητή), στην ιδιότητα OutputSandboxDestURI, προσδιορίζοντας το με ένα Uniform Resource Identifier (URI) δηλ. ένα string το οποίο χρησιμοποιείται για τον προσδιορισμό ενός ονόματος ή μιας πηγής στο Internet. Π.χ.:

StdOutput = “myjobOutput”;
OutputSandbox = { “myjobOutput”, … };

OutputSandboxDestURI =

{“gsiftp://fox.infn.it:5678/home/gftp/myjobOutput”,…};

σημαίνει ότι το αποτέλεσμα/έξοδος myjobOutput, όταν η εργασία εκτελεσθεί θα πρέπει να μεταφερθεί στο gsiftp://fox.infn.it:5678 στον φάκελο /home/gftp.

6.2.7. StdError

Η ιδιότητα StdError είναι ένα string που αντιπροσωπεύει το όνομα του αρχείου στο WN όπου αποθηκεύονται τα μηνύματα λάθους.

Ο χρήστης μπορεί είτε να ορίσει ένα όνομα αρχείου ή μια σχετική διαδρομή (σε σχέση με τον κατάλογο των εργασιών που εκτελούνται στον WN), π.χ.:

StdError = “myjobError”;

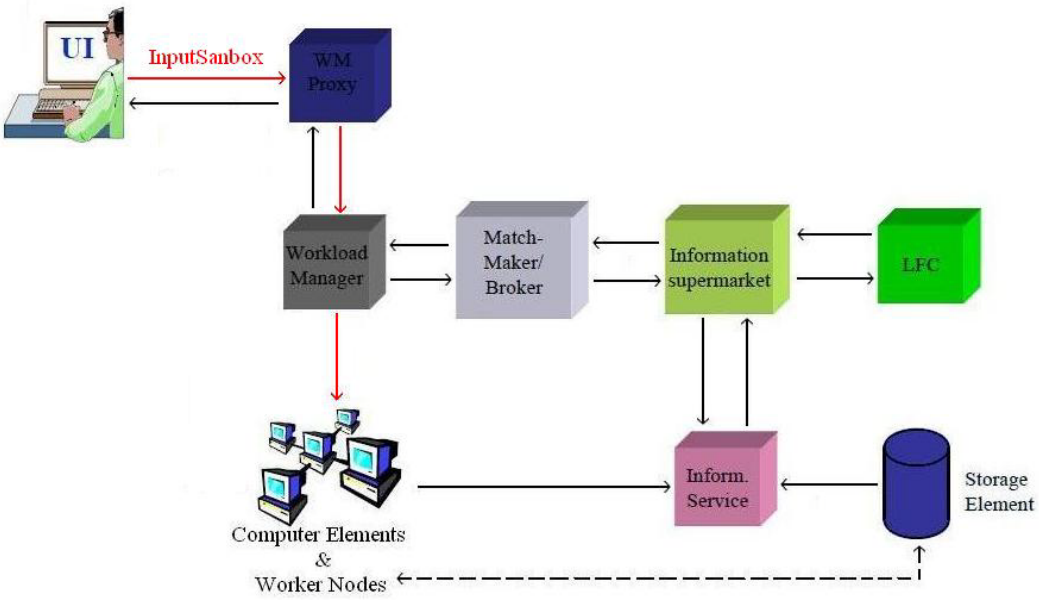
StdError = “event1/myjobError”;

Οι κανόνες για το StdError και το StdOutput είναι ίδιοι.

6.2.8. InputSandbox

Η ιδιότητα InputSandbox είναι ένα string ή μία λίστα από strings που προσδιορίζουν τον φάκελο των αρχείων στο τοπικό σύστημα αρχείων στο UI (user interface) ή σε έναν προσβάσιμο server του Gridftp (η ιδιότητα υποστηρίζεται και από HTTPS servers αρκεί ο χρήστης να έχει εγκαταστήσει την εντολή Gridsite htcp στο WN, μία εντολή UNIX για την διενέργεια εργασιών με αρχεία σε απομακρυσμένους HTTP και HTTPS fileservers) και απαιτείται ώστε να εκτελεστεί η εργασία. Αυτά τα αρχεία μεταφέρονται από το περιβάλλον του χρήστη (UI) στον κόμβο WMS και έπειτα μεταφορτώνονται στον CE WN όπου η εργασία εκτελείται από τον WMS JobWrapper

κώδικα. Αν βρίσκονται ήδη σε έναν Gridftp server τότε μεταφορτώνονται απευθείας στον CE WN όπου η εργασία εκτελείται από τον WMS JobWrapper κώδικα.



Εικόνα 35 : Η διαδρομή του InputSandbox στο πλέγμα

Wildcards και μεταβλητές περιβάλλοντος εισάγονται στον προσδιορισμό της ιδιότητας μόνο εάν η υποβολή της εργασίας γίνεται από χρήστη που είναι σε θέση να τα διευκρινίσει προτού μεταβιβάσει το JDL στην υπηρεσία WMProxy. Αποδεκτά wildcards είναι αυτά που υποστηρίζονται από το λειτουργικό Linux glob. Το globbing είναι μία λειτουργία που επεκτείνει το όνομα ενός αρχείου που περιέχει ένα wildcard σε ένα πρότυπο όνομα που ήδη υπάρχει αποθηκευμένο σε κάποιον άλλο Η/Υ ή server [40]. Το ειδικό νόημα των χαρακτήρων " ? ", " * " και " [" μπορεί να αφαιρεθεί τοποθετώντας πριν από αυτά τον χαρακτήρα " \ ".

Τα ονόματα των αρχείων μπορούν να παρέχονται ως URI για ένα gridFTP/HTTPS server, απλά ονόματα αρχείων, άμεσες ή έμμεσες διευθύνσεις και διαδρομές (absolute and relative paths) : μία άμεση διεύθυνση είναι π.χ. http://public.web.cern.ch/public/ όπου ο χρήστης πηγαίνει απευθείας στο site, ενώ

έμμεση θα ήταν το ./public/ σε σχέση με τον τρέχοντα UI φάκελο εργασίας. Ο φάκελος αρχείων InputSandbox δεν μπορεί να περιέχει δύο ή περισσότερα αρχεία με το ίδιο όνομα (έστω και με διαφορετικές διευθύνσεις), καθώς όταν θα μεταφέρονται στον φάκελο εργασίας στο WN θα αντικατασταθεί το ένα με το άλλο.

Το χαρακτηριστικό αυτό χρησιμοποιείται επίσης για να ολοκληρώσει την εκτελέσιμη και την τυπική είσοδο που μεταφέρονται στο WN του CE όπου εκτελείται η εργασία. Το InputSandbox είναι απολύτως συνδεδεμένο με την τιμή της ιδιότητας InputSandboxBaseURI που εξηγείται παρακάτω και καθορίζει μια θέση σε έναν gridFTP / HTTPS server, όπου βρίσκονται τα αρχεία του InputSandbox που δεν έχουν ήδη οριστεί ως URI.

Παρακάτω ακολουθεί ένα παράδειγμα ρύθμισης InputSandbox :

```
InputSandbox = {
                    "/tmp/ns.log",
                    "mytest.exe",
                    "myscript.sh",
                    "data/event1.txt",
"gsiftp://neo.datamat.it:5678/home/fpacini/cms_sim.exe ",
                    "file:///tmp/myconf"
};

InputSandboxBaseURI = "gsiftp://matrix.datamat.it:5432/tmp";
```

Αυτό σημαίνει ότι :

- tmp/log βρίσκεται στο τοπικό σύστημα αρχείων του χρήστη.
- mytest.exe,myscript.sh και τα δεδομένα /event1.txt είναι διαθέσιμα στο gsiftp://matrix.datamat.it:5432 στον φάκελο /tmp.
- /tmp/myconf βρίσκεται στο τοπικό σύστημα αρχείων του χρήστη (διευκρινίζε-

ται χρησιμοποιώντας το πρόθεμα file://).

Εάν δεν διευκρινίζεται η ιδιότητα InputSandboxBaseURI θα θεωρηθεί ότι το mytest.exe, myscript.sh καθώς επίσης και το data/event1.txt βρίσκονται στο τοπικό σύστημα αρχείων του χρήστη.

6.2.9. InputSandboxBaseURI

Η ιδιότητα InputSandboxBaseURI είναι ένα string που αντιπροσωπεύει το URI για έναν gridFTP διακομιστή (Οι HTTPS servers υποστηρίζονται, αλλά αυτό απαιτεί ο χρήστης να έχει την εντολή htcp GridSite εγκατεστημένη στον WN) και όπου τα αρχεία InputSandbox που έχουν οριστεί ως απλά ονόματα αρχείων, άμεσες και έμμεσες διευθύνσεις ή διαδρομές, είναι διαθέσιμα για την μεταφορά τους στο WN πριν αρχίσει να εκτελείται η εργασία. π.χ.

```
InputSandbox = {
                    …
                    "data/event1.txt",
                    …
                     };
InputSandboxBaseURI= "gsiftp://matrix.datamat.it:5432/tmp";
```

Κάνει το WMS να θεωρεί το

"gsiftp://matrix.datamat.it:5432/tmp/data/event1.txt"

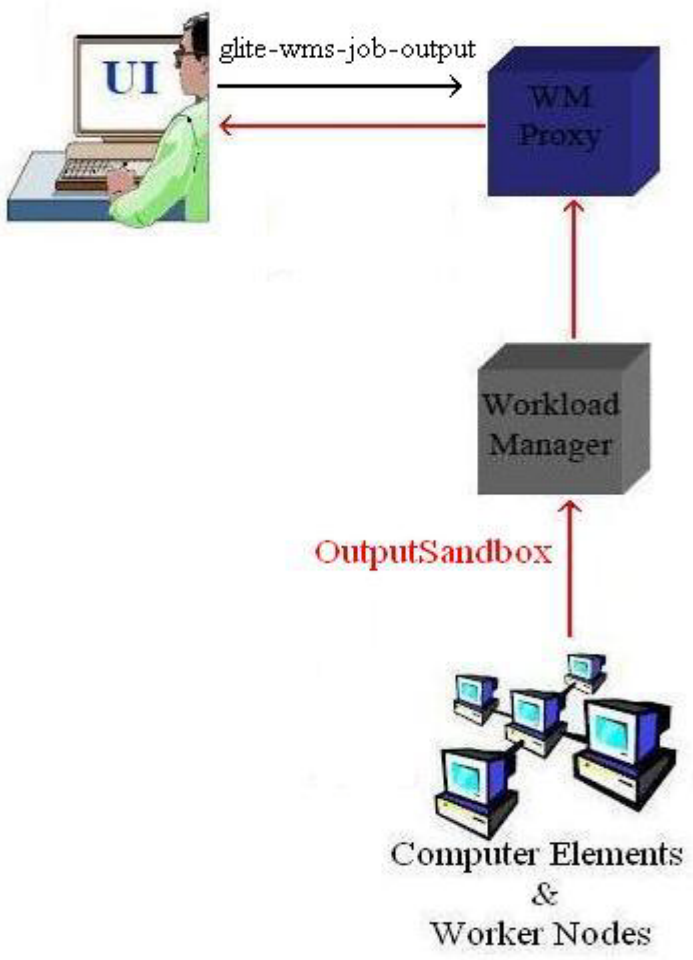
για την μεταφορά στο WN.

6.2.10. OutputSandbox

Η ιδιότητα OutputSandbox είναι ένα string ή μια λίστα strings για τον προσδιορισμό του φακέλου των αρχείων που δημιουργούνται από την εργασία στο WN κατά το χρόνο εκτέλεσης, την οποία ο χρήστης επιθυμεί να ανακτήσει. Τα αρχεία που παρατίθενται, μεταφέρονται στον κόμβο WMS και μπορούν να μεταφορτωθούν στο UI τοπικό σύστημα αρχείων με χρήση της εντολής *glite-wms-job-output*. Wildcards δεν έχουν εισαχθεί στη συγγραφή αυτής της ιδιότητας.

Αυτή η ιδιότητα μπορεί να συνδυαστεί με την OutputSandboxDestURI (που περιγράφεται στην παράγραφο 6.2.11) ή την OutputSandboxBaseDestURI (που περιγράφεται στην παράγραφο 6.2.12) για να έχουν, μετά την ολοκλήρωση της εργασίας, την αντιγραφή της εξόδου σε συγκεκριμένες τοποθεσίες που χρησιμοποιούν έναν gridFTP server. Σημειώνεται ότι τα αρχεία εξόδου που αντιγράφονται με τον τρόπο αυτό δεν χρειάζεται να ανακτηθούν από την εντολή gLite-wms-job-output .

Τα ονόματα αρχείων μπορούν να είναι είτε απλά ονόματα ή έμμεσες διευθύνσεις σχετικά με τον τρέχοντα φάκελο εργασίας, στον WN που εκτελείται. Ο φάκελος αρχείων δεν πρέπει να περιέχει δύο ή περισσότερα αρχεία με το ίδιο όνομα, καθώς όταν θα μεταφέρονται στο WMS θα αντικαταστήσουν το ένα το άλλο, εκτός αν στην ιδιότητα OutputSandBoxDestURI έχει προσδιοριστεί διαφορετικός προορισμός URI.



Εικόνα 36 : Το OutputSandbox στην ανάκτηση της εξόδου

Παρακάτω φαίνεται ένα παράδειγμα OutputSandBox:

```
OutputSandbox = {
                "myjobOutput",
                "myjobError",
                "run1/event1",
                "run1/event2",
        };
```

Αυτό δείχνει ότι όλα τα αρχεία που παρατίθενται (αν όντως προκύπτουν από την εργασία) θα είναι διαθέσιμα, μετά την ολοκλήρωση της εργασίας, στον

κόμβο WMS (αν δεν έχουν προσδιοριστεί τα OutputSandboxDestURI και OutputSandboxBaseDestURI) και θα είναι ανακτήσιμα με την χρήση της εντολής:

glite-wms-job-output.

6.2.11. OutputSandboxDestURI

Η ιδιότητα OutputSandboxDestURI περιέχει το URI (συμπεριλαμβανομένου του ονόματος αρχείου), όπου θα πρέπει να μεταφερθεί μετά την ολοκλήρωση της εργασίας κάθε ένα από τα αρχεία που αναφέρονται στον φάκελο OutputSandbox.

```
OutputSandbox = {
        "myjobOutput",
        "run1/event1",
        "myjobError"
        };
OutputSandboxDestURI = {
        "gsiftp://matrix.datamat.it:5432/tmp/myjobOutput ",
        "gsiftp://grid003.ct.infn.it:6789/home/cms/event1",
        "myjobError"
        };
```

Το WMS θα κάνει την αντίστοιχη μεταφορά :

- Το myjobOutput θα πάει στο matrix.datamat στον φάκελο /tmp
- Το event1 στο grid003.ct.infn.it στον φάκελο /home/cms
- Το myjobError στον κόμβο WMS

Η ιδιότητα OutputSandboxDestURI πρέπει να έχει το ίδιο πλήθος αρχείων με τον φάκελο OutputSandbox, διαφορετικά το JDL θα θεωρηθεί ως άκυρο. Να σημειωθεί ότι το όνομα του αρχείου που καθορίζεται στο OutputSandbox μπορεί να

είναι διαφορετικό από το αντίστοιχο όνομα του αρχείου προορισμού που ορίζεται στην OutputSandboxBaseDestURI.

Οι ιδιότητες *OutputSandboxDestURI* και *OutputSandboxBaseDestURI* δεν μπορούν να προσδιοριστούν και οι δύο στο ίδιο JDL.

Αν καμία από τις δύο παραπάνω ιδιότητες δεν καθορίζεται στο JDL, τότε όλα τα αρχεία που βρίσκονται στον φάκελο OutputSandbox θα είναι διαθέσιμα, μετά την ολοκλήρωση της εργασίας, στον κόμβο WMS και θα είναι διαθέσιμα για ανάκτηση χρησιμοποιώντας την εντολή gLite-wms-job-output. Σημειώστε ότι αν τα URI κάποιων αρχείων δεν κατευθύνονται στον κόμβο WMS θα είναι ανακτήσιμα με την εντολή glite-wms-job-output.

6.2.12. OutputSandBoxBaseDestURI

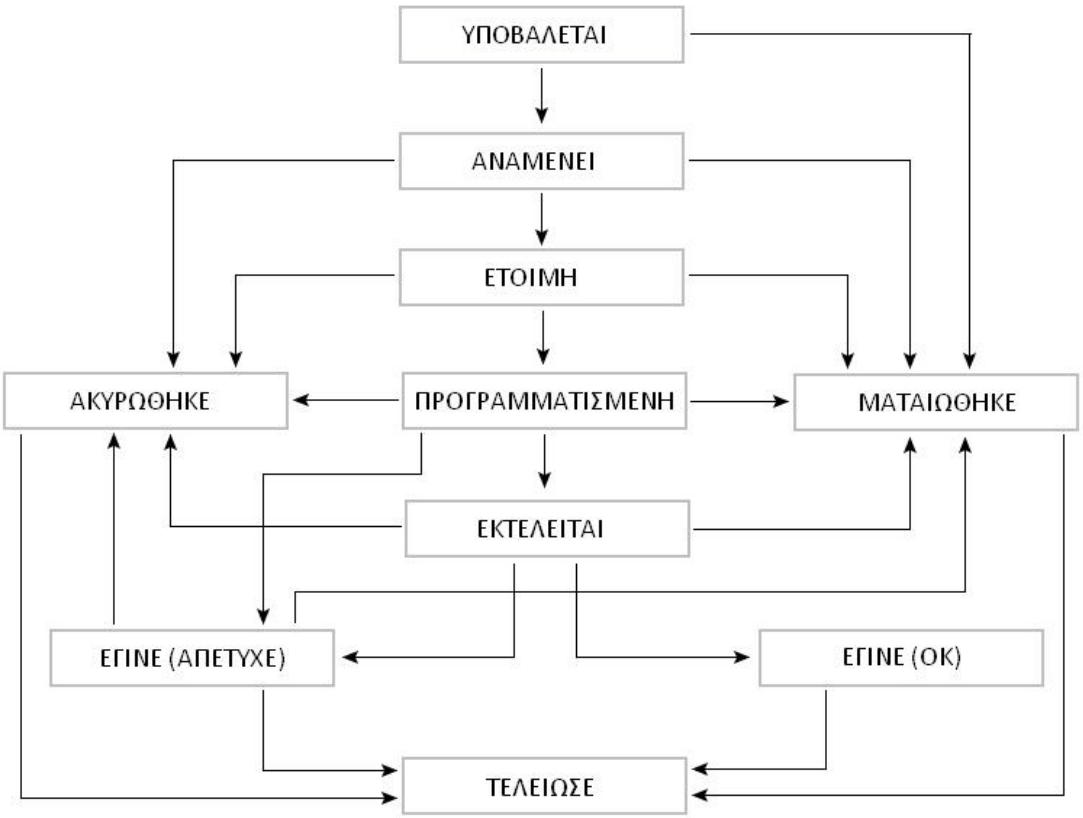
Η ιδιότητα OutputSandboxBaseDestURI είναι ένα string που αντιπροσωπεύει το βασικό (base) URI σε έναν gridFTP /HTTPS server, δηλαδή, ενός φακέλου στον εξυπηρετητή, όπου τα αρχεία που περιλαμβάνονται στο χαρακτηριστικό OutputSandbox που πρόκειται να μεταφερθούν με την ολοκλήρωση της εργασίας. Όλα τα αρχεία του OutputSandbox μεταφέρονται στη θέση που καθορίζεται από το URI με τα ίδια ονόματα, όπως αυτά έχουν καθοριστεί στο OutputSandbox.

Π.χ.

```
OutputSandbox = {
                "myjobOutput",
                "run1/event1",
                };
OutputSandboxBaseDestURI= "gsiftp://matrix.datamat.it:5432/tmp";
```

Το WMS θα μεταφέρει και τα δύο αρχεία στον φάκελο /tmp στον gridFtp server matrix.datamat.it (να σημειωθεί ότι το event1 θα πάει στο /tmp και όχι στο /tmp/run1).

7. Εργασίες



Εικόνα 37 : Οι πιθανές καταστάσεις μιας εργασίας

7.1. Παράδειγμα 1^{ο} : Normal Job

Για κάθε εργασία χρειαζόμαστε ένα αρχείο JDL. Σε ένα αρχείο WordPad λοιπόν, γράφουμε τα εξής :

```
executable="/bin/echo";
arguments="Geia Sou";
stdout="gs.out";
stderror="gs.err";
outputsandbox={"gs.out","gs.err"};
virtualorganisation="see";
```

και στη συνέχεια αλλάζουμε την κατάληξη του αρχείου σε .jdl, στην περίπτωση αυτή θα το αποθηκεύσουμε ως geiasou.jdl .

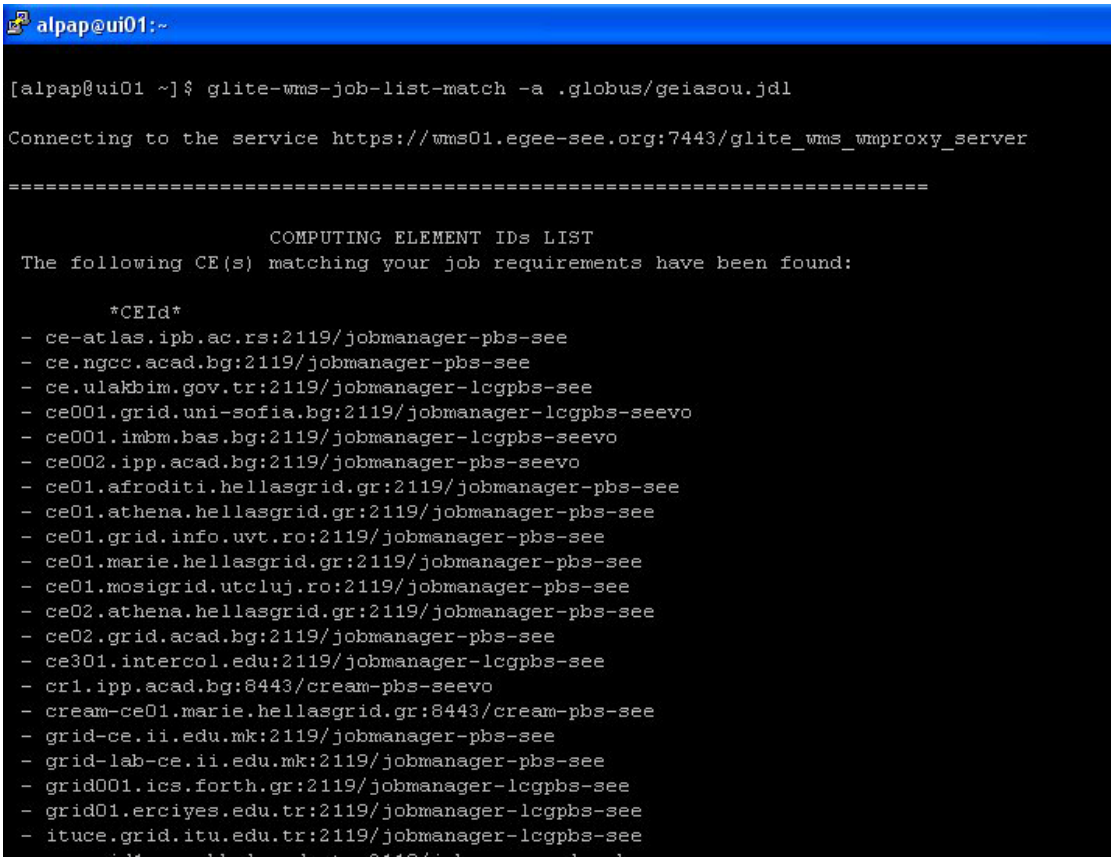
Αυτό που γίνεται είναι να εμφανίζεται το string "Geia Sou" στην έξοδο stdoutput. Τα stdout και stderr ανακατευθύνονται στα αρχεία που ονομάζονται gs.out και gs.err, και στο τέλος της εργασίας τα αρχεία αυτά θα επιστραφούν στο "outputsandbox", μια λίστα αρχείων η οποία μπορεί να ανακτηθεί όταν ολοκληρωθεί η εργασία.

Το αρχείο αυτό θα πρέπει να αντιγραφεί στον λογαριασμό του χρήστη στο UI χρησιμοποιώντας το πρόγραμμα WinSCP που χρησιμοποιήθηκε και για την αντιγραφή του πιστοποιητικού.

Για να ελέγξουμε ότι όλα λειτουργούν σωστά πληκτρολογούμε την εντολή :

- **glite-wms-job-list-match –a geiasou.jdl**

όπου λαμβάνουμε την λίστα με τα διαθέσιμα CE (computer elements) που μπορούν να αναλάβουν την εκτέλεση της εργασίας. Η επέκταση –a είναι για την αυτόματη αντιπροσωπεία.

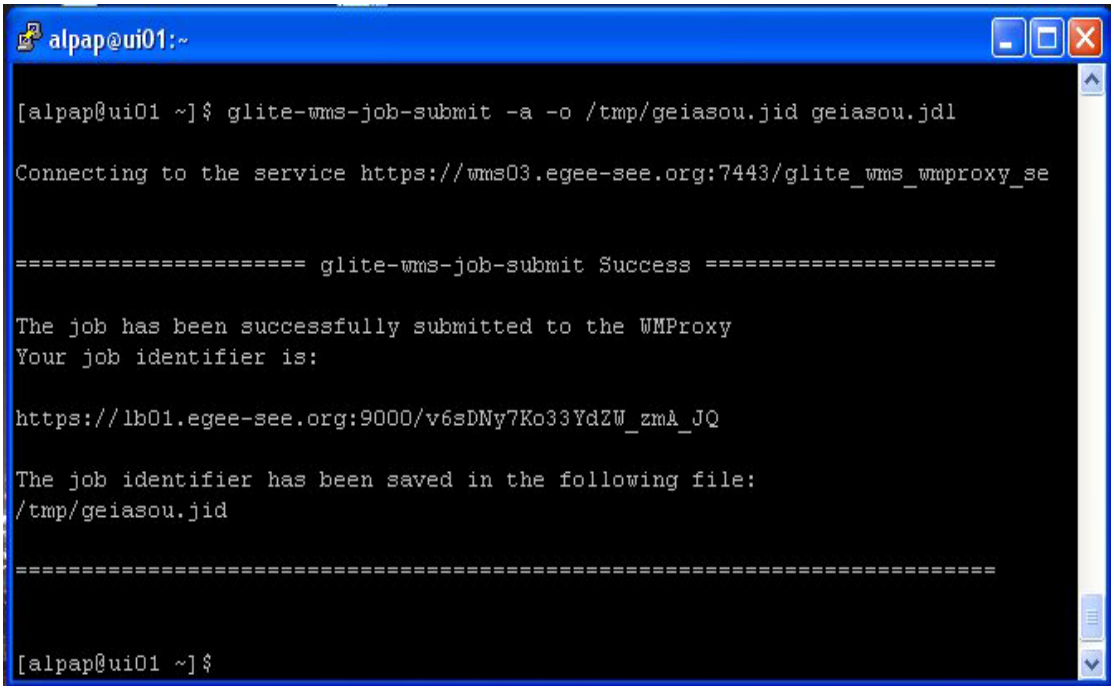


Εικόνα 38 : Η εντολή glite-wms-job-list-match

Δεν έμεινε παρά να υποβάλλουμε την εργασία :

- **glite-wms-job submit –a –o /tmp/geiasou.jid geiasou.jdl**

Το αρχείο geiasou.jid είναι ένα προσωρινό αρχείο που περιέχει την ταυτότητα της εργασίας, ενώ το χαρακτηριστικό –o (output) επισημαίνει ότι η ταυτότητα ID της εργασίας θα αποθηκευτεί στο αρχείο ή διαδρομή που ακολουθεί, στην περίπτωση αυτή στο geiasou.jid αρχείο.

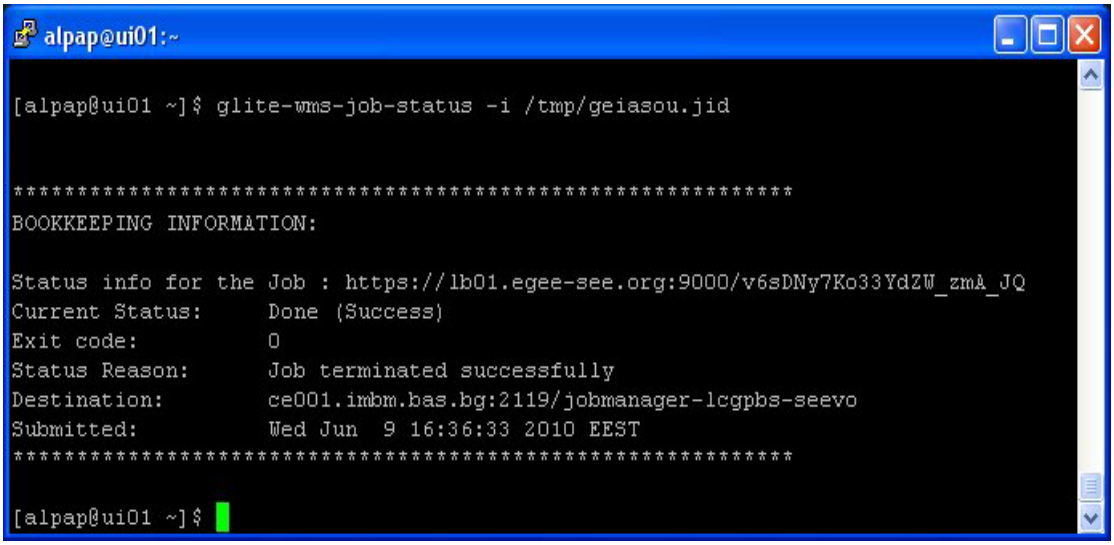


Εικόνα 39 : Υποβολή μιας εργασίας

Ακόμη και μία σύντομη εργασία όπως αυτή, θα χρειαστεί μερικά λεπτά και ενδεχομένως πολύ περισσότερο όταν δεν υπάρχουν ελεύθεροι πόροι και η εργασία πρέπει να παραμείνει στην αναμονή. Αν όλα πάνε καλά στο status θα πρέπει τελικά να γράφει “Done”. Το status, η κατάσταση δηλαδή της εργασίας δίνεται με την εντολή :

- **glite-wms-job-status -i /tmp/geiasou.jid**

όπου με το χαρακτηριστικό -i ζητάμε πληροφορίες για την κατάσταση της εργασίας που υποδεικνύει η διαδρομή που ακολουθεί, /tmp/geiasou.jid . Το status μπορεί να είναι σε αναμονή (scheduled), να εκτελείται (in process) ή έγινε (done).

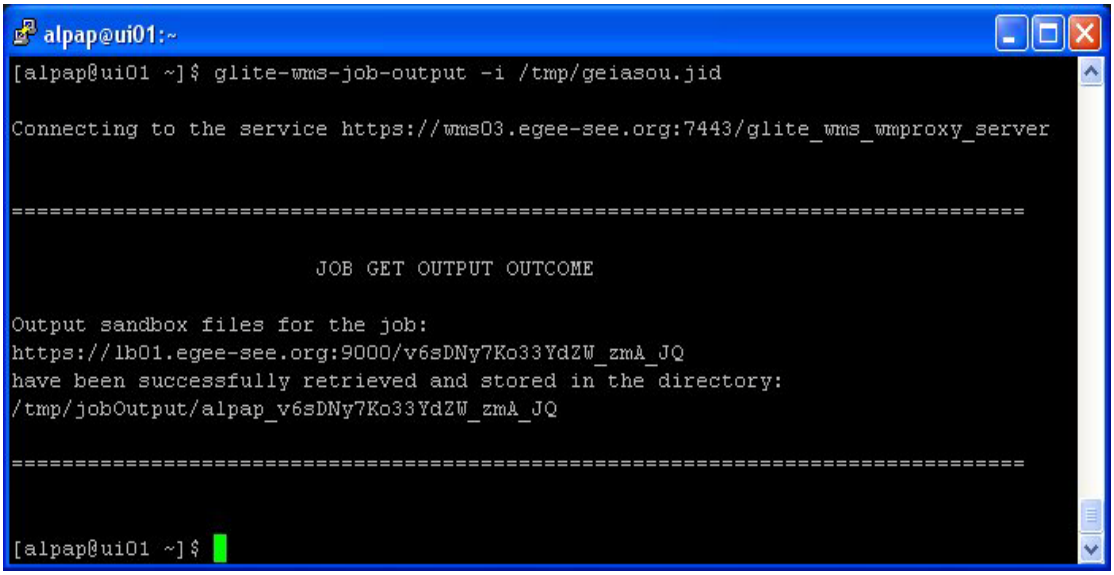


Εικόνα 40 : Έλεγχος κατάστασης μιας εργασίας

Σε αυτό το σημείο μπορούμε να ανακτήσουμε την παραγωγή της εργασίας (το stdout και stderr αρχεία σε αυτή την περίπτωση) δηλαδή τα gs.out gs.err :

- **glite-wms-job-output –i /tmp/geiasou.jid**

Η έξοδος αποθηκεύεται σε έναν κατάλογο που ρυθμίζεται τοπικά και συνήθως στον κατάλογο / tmp, ή μπορεί ο χρήστης να δώσει το δικό του κατάλογο με το χαρακτηριστικό --dir ακολουθούμενο από τον κατάλογο. Σε κάθε περίπτωση, η παραπάνω εντολή θα εκτυπώσει την τοποθεσία.



Εικόνα 41 : Ανάκτηση εξόδου εργασίας

Μπορούμε να επιβεβαιώσουμε ότι ανακτήθηκαν τα αρχεία με την εντολή **cd** στον φάκελο που μας υποδεικνύει το σύστημα:

/tmp/jobOutput/alpap_v6sDNy7Ko33YdZW_zmA_JQ .

Όπως φαίνεται και στην παραπάνω εικόνα το gs.err είναι μηδενικό, κάτι που σημαίνει πως όλα δούλεψαν σωστά και χωρίς λάθη. Δεν μένει παρά να ελέγξουμε το περιεχόμενο της εξόδου gs.out δηλαδή το μήνυμα " Geia Sou ". Αυτό γίνεται με την εντολή : cat gs.out . [79]

Εικόνα 42 : Εμφάνιση εξόδου εργασίας

7.2. Παράδειγμα 2^{ο} : Normal Job με InputSandbox

Η παραπάνω εργασία εκτελεί μόνο την προ-εγκατεστημένη εντολή (/ bin / echo), αλλά στις περισσότερες περιπτώσεις ο χρήστης θα θέλει να στείλει ένα κείμενο και ενδεχομένως άλλα αρχεία με την εργασία. Αυτά αποστέλλονται με το λεγόμενο inputsandbox.

Ας ξεκινήσουμε λοιπόν δημιουργώντας ένα αρχείο txt που θα ονομάσουμε geiaxara.txt και το οποίο θα περιέχει το string "Geia Xara!!".

Έπειτα ένα άλλο αρχείο που θα περιέχει τα εξής :

```
#!/bin/bash
cat $1
```

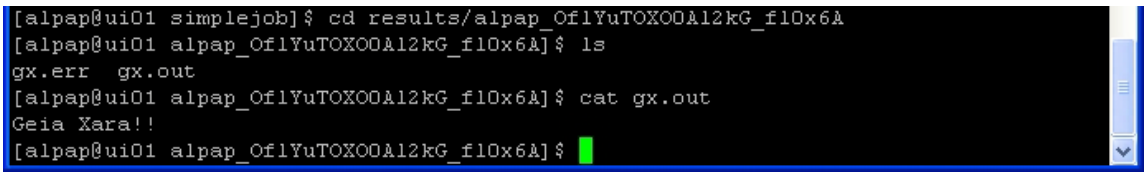
και του οποίου θα αλλάξουμε την κατάληξη σε .sh και θα το ονομάσουμε geiaxara.sh. Αυτό που κάνει το παραπάνω αρχείο είναι να εμφανίζει το περιεχόμενο του πρώτου ($1) argument (δηλ. το geiaxara.txt) που θα βρει στο jdl αρχείο όπως φαίνεται

παρακάτω, δηλαδή το μήνυμα “Geia Xara”.Τέλος θα δημιουργήσουμε το αρχείο με κατάληξη .jdl και θα το ονομάσουμε geiaxara.jdl . Θα περιέχει τα παρακάτω :

```
executable = "geiaxara.sh";
arguments = "geiaxara.txt";
inputsandbox = {"geiaxara.sh", "geiaxara.txt"};
stdoutput = "gx.out";
stderror = "gx.err";
outputsandbox = {"gx.out", "gx.err"};
virtualorganisation = "see";
```

Όπως φαίνεται, υπάρχει μία νέα παράμετρος, το inputsandbox με το οποίο θα σταλούν τα geiaxara.sh και geiaxara.txt.

Στη συνέχεια αντιγράφουμε τα αρχεία αυτά από το τοπικό σύστημα αρχείων στο λογαριασμό μας στο UI όπως στο παραπάνω παράδειγμα. Υποβάλουμε την εργασία και όταν αυτή ολοκληρωθεί και ανακτηθεί η έξοδος ακολουθώντας τις εντολές που δίνονται και στο 1^{ο} παράδειγμα, μπορούμε να εμφανίσουμε το αποτέλεσμα πληκτρολογώντας **cat gx.out** (όπου gx.out=std.out δηλ. η έξοδος).

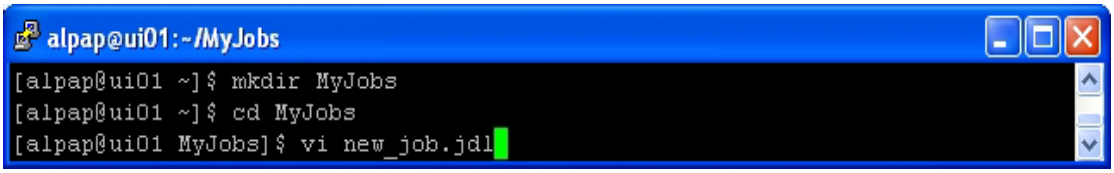


Εικόνα 43 : Εμφάνιση της εξόδου

7.3. Παράδειγμα 3^{ο} : Δημιουργία αρχείου στο UI

Ένα αρχείο jdl μπορεί εύκολα να γραφτεί και στην γραμμή εντολών μέσω του UI γλιτώνοντας έτσι την αντιγραφή του από το τοπικό σύστημα αρχείων του χρήστη στο UI με την χρήση του WinSPC. Στο παράδειγμα που ακολουθεί εξηγείται η διαδικασία αυτή.

Αφού μπούμε στον λογαριασμό μας στο UI φτιάχνουμε έναν φάκελο με το όνομα MyJobs με την εντολή **mkdir MyJobs** και ύστερα πληκτρολογώντας **cd MyJobs** μπαίνουμε στον φάκελο αυτόν. Για την δημιουργία του αρχείου jdl χρησιμοποιούμε την εντολή **vi** (variable information), δηλαδή εμείς δίνουμε τις πληροφορίες των μεταβλητών για το αρχείο αυτό. Θα ονομάσουμε το αρχείο new_job.jdl οπότε η εντολή που θα δώσουμε είναι η : **vi new_job.jdl** .



Εικόνα 44 : Δημιουργία φακέλου και αρχείου στο UI

Όταν πατήσουμε ENTER για την εκτέλεση της τελευταίας εντολής θα δημιουργηθεί το αρχείο new_job.jdl και θα ανοίξει ένα κενό παράθυρο μέσα στο οποίο θα γράψουμε τις μεταβλητές του.

Εικόνα 45 : Δημιουργία ενός αρχείου

Για να μπορέσουμε να ξεκινήσουμε να γράφουμε πατάμε μία φορά τον χαρακτήρα « I » (INSERT). Έπειτα δίνουμε τις εξής μεταβλητές:

Executable = “/bin/hostname”;

Arguments = “-f”;

StdOutput = “std.out”;

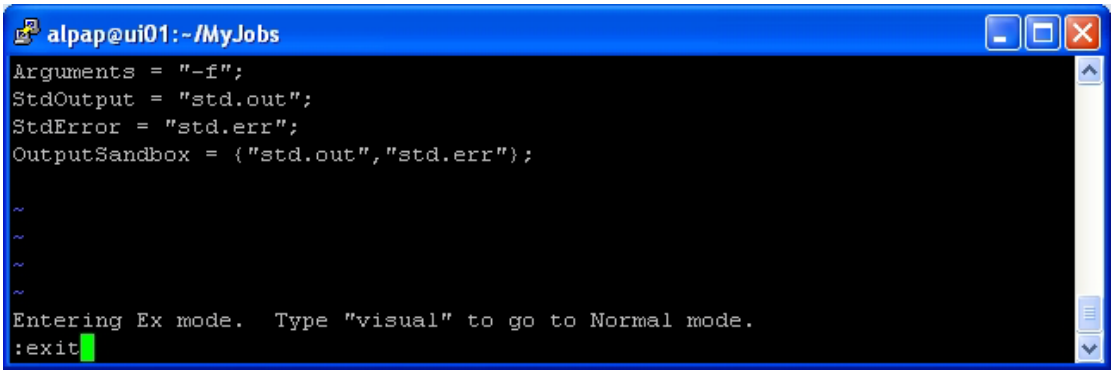
StdError = “std.err”;

OutputSandbox = {“std.out”,”std.err”};

Εικόνα 46 : Το jdl αρχείο

Το εκτελέσιμο **/bin/hostname** εκτυπώνει στην οθόνη το όνομα του υποδοχέα Host του Worker Node (WN) και η μεταβλητή **-f** το fqdn (fully qualified domain name) του.

Για να επιστρέψουμε στον φάκελο MyJobs που θα περιέχει και το καινούργιο jdl αρχείο, πατάμε το πλήκτρο **ESC**, μετά **shift+Q** (από το quit) και πληκτρολογούμε **exit** (ή **visual** αν θέλουμε να επιστρέψουμε και να διορθώσουμε κάτι).



Εικόνα 47 : Έξοδος από την δημιουργία αρχείου

Τώρα που είναι έτοιμο το jdl αρχείο, το μόνο πράγμα που έμεινε πριν υποβάλλουμε την εργασία είναι να φτιάξουμε ένα proxy πιστοποιητικό.

Πληκτρολογούμε λοιπόν για την υποβολή την εντολή :

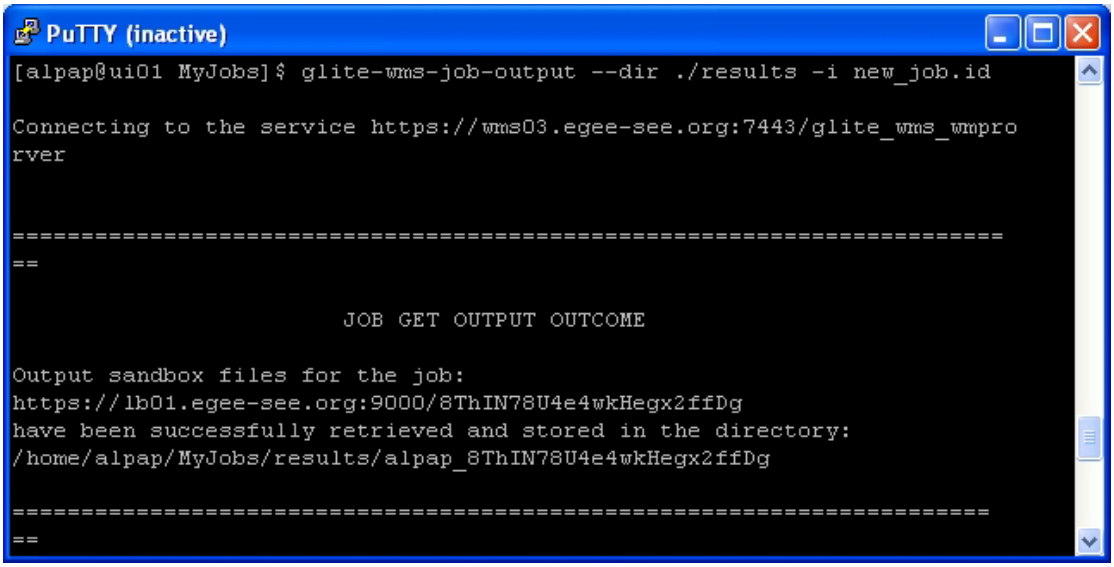
glite-wms-job-submit -a -o new_job.id new_job.jdl

και περιμένουμε μέχρι να εκτελεστεί (Done), ελέγχοντας το status με την εντολή :

glite-wms-job-status new_job.id

Όταν εκτελεστεί η εργασία, η ανάκτηση των εξόδων γίνεται με την εντολή :

glite-wms-job-output --dir ./results -i new_job.id



Εικόνα 48 : Ανάκτηση εξόδου και αποθήκευση στον φάκελο results

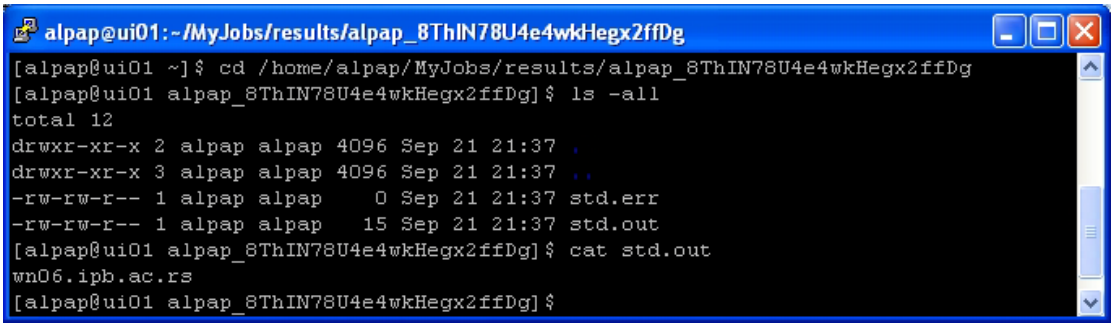
Με την μεταβλητή --dir δηλώνουμε την δημιουργία του φακέλου **results** μέσα στον οποίο θα αποθηκευτεί ο φάκελος με τις εξόδους ο οποίος στην περίπτωση αυτή όπως φαίνεται και στην παραπάνω εικόνα είναι ο **alpap_8ThIN78U4e4wkHegx2ffDg**. Για να δούμε λοιπόν το αποτέλεσμα και να μπούμε στον παραπάνω φάκελο εκτελούμε την εντολή:

cd /home/alpap/MyJobs/results/alpap_8ThIN78U4e4wkHegx2ffDg ,

ls -all για να ελέγξουμε ότι ο φάκελος περιέχει τα **std.out** και **std.err**

και

cat std.out για να δούμε το περιεχόμενο της εξόδου (το host name του WN). [42]



Εικόνα 49 : Εμφάνιση αποτελεσμάτων εργασίας

7.4. Παράδειγμα 4^{ο} : Collection Job

Το παράδειγμα που ακολουθεί είναι η υποβολή μίας Collection, δηλαδή μίας συλλογής ανεξάρτητων εργασιών που υποβάλλονται όλες μαζί σαν μία.

Δημιουργούμε έναν φάκελο με το όνομα **JobCollection** και μέσα σε αυτόν έναν άλλο με το όνομα **jdl-collection** που θα περιέχει τρία αρχεία jdl που θα φτιάξουμε με την ίδια διαδικασία που χρησιμοποιήσαμε στο 3^{ο} παράδειγμα. Τα αρχεία αυτά θα είναι τα **job1.jdl**, **job2.jdl** και **job3.jdl** και τα οποία θα περιέχουν αντίστοιχα:

```
Executable = "/bin/hostname";
Arguments = "-f";
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err"};
```

Το οποίο είναι το ίδιο αρχείο με αυτό στο 3^{ο} παράδειγμα,

```
Executable = "/bin/echo";
Arguments = "Hello World";
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err"};
```

το οποίο τυπώνει το “Hello World” όπως δείξαμε στο παράδειγμα 1,

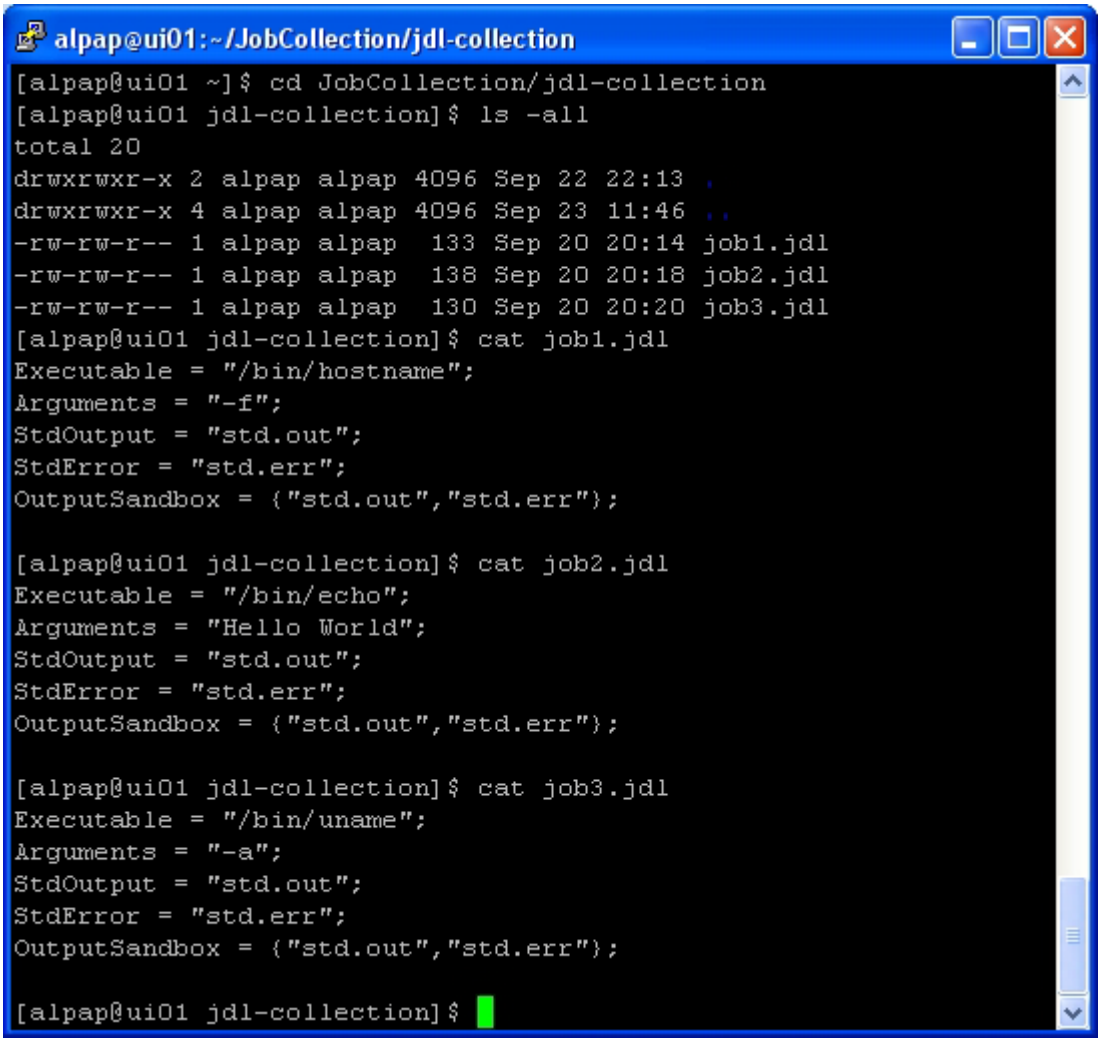
και τέλος

```
Executable = "/bin/uname";
Arguments = "-a";
StdOutput = "std.out";
StdError = "std.err";
```

OutputSandbox = {"std.out","std.err"};

που τυπώνει πληροφορίες για το λειτουργικό σύστημα του WN.

Αν όλα πάνε καλά τα περιεχόμενα των φακέλων θα πρέπει να είναι κάπως έτσι:



Εικόνα 50 : Τα jdl αρχεία μιας εργασίας collection

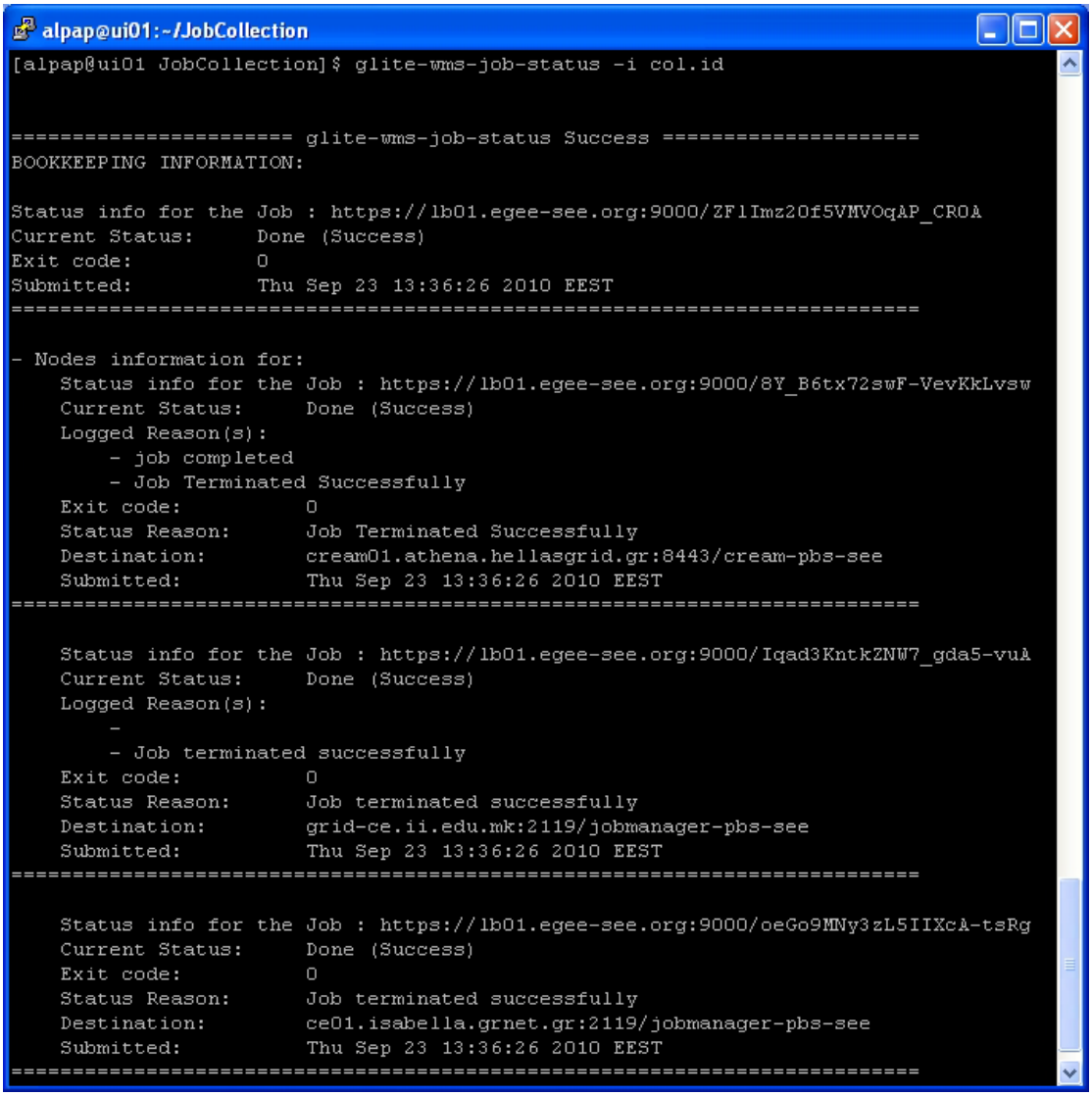
Αφού γυρίσουμε στον προηγούμενο φάκελο με την εντολή **cd ..** (ο χρήστης θα πρέπει να δίνει μεγάλη προσοχή και να γνωρίζει σε ποιον φάκελο/κατάλογο βρίσκεται όταν πληκτρολογεί εντολές) υποβάλουμε την εργασία πληκτρολογώντας:

glite-wms-job-submit -a -o col.id --collection jdl-collection/

Ονομάζουμε **col.id** την ταυτότητα της εργασίας και με την μεταβλητή --collection δηλώνουμε την υποβολή μιας συλλογής εργασιών με τα αρχεία jdl να βρίσκονται στον φάκελο **jdl-collection**.

Ελέγχουμε την κατάσταση της εργασίας:

glite-wms-job-status -i col.id



Εικόνα 51 : Έλεγχος κατάστασης μιας εργασίας collection

Όπως φαίνεται στην παραπάνω εικόνα δίνοντας την εντολή, παίρνουμε το αποτέλεσμα της κατάστασης για όλη την συλλογή αλλά και για κάθε μία εργασία χωριστά.

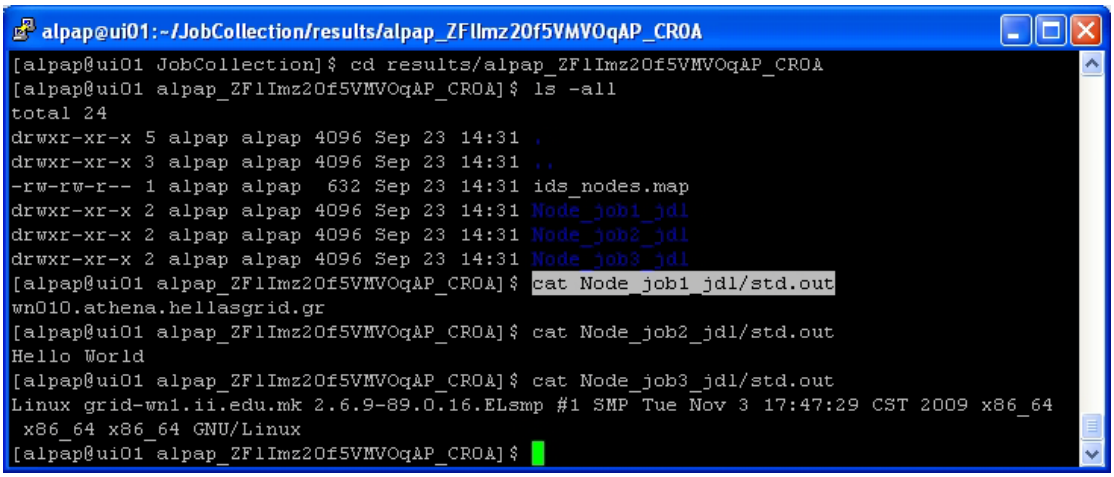
Τέλος για την ανάκτηση των εξόδων χρησιμοποιούμε την εντολή:

glite-wms-job-output --dir ./results -i col.id

Θα δημιουργηθεί ένας φάκελος με το όνομα **results** στον οποίο θα αποθηκευτεί ο φάκελος με τα δεδομένα. Στην προκειμένη το σύστημα τον ονόμασε alpap_ZFlImz20f5VMVOqAP_CR0A, άρα για να έχουμε πρόσβαση σε αυτόν δίνουμε την εντολή :

cd results/alpap_ZFlImz20f5VMVOqAP_CR0A

Στον φάκελο βρίσκονται τα αποτελέσματα χωρισμένα σε φακέλους οι οποίοι περιέχουν τα **std.err std.out** για κάθε εργασία. Με την εντολή **cat** για κάθε αρχείο **std.out** βλέπουμε το περιεχόμενο όπως παρακάτω. [43]



Εικόνα 52 : Εμφάνιση αποτελεσμάτων μιας εργασίας collection

Παράδειγμα 5^{ο} : Parametric Job 1

Μία Parametric job είναι μια εργασία Collection (συλλογή) όπου οι εργασίες που εκτελούνται είναι πανομοιότυπες, με μόνη διαφορά μία παράμετρο. Χαρακτηρίζεται από ένα ενιαίο jdl αρχείο, όπου οι τιμές των ιδιοτήτων μπορεί να

περιλαμβάνουν διάφορες τιμές ή την παράμετρο. Παρακάτω ακολουθεί ένα παράδειγμα μιας τέτοιας εργασίας.

Δημιουργούμε λοιπόν ένα αρχείο jdl (στην περίπτωση αυτή είναι το parametric3.jdl) με την εντολή **vi** και γράφουμε τα εξής:

```
JobType = "Parametric";
Executable = "/bin/cat";
Arguments = "input_PARAM_.txt";
InputSandbox = "input_PARAM_.txt";
StdOutput = "myoutput_PARAM_.txt";
StdError = "myerror_PARAM_.txt";
Parameters = {EARTH,MARS,MOON};
OutputSandbox = {"myoutput_PARAM_.txt"};
```

Όπως φαίνεται η ιδιότητα **JobType** τίθεται ως **Parametric**. Το ειδικό string **_PARAM_** δείχνει την παραμετρική ιδιότητα και θα αντικατασταθεί με την σωστή τιμή κατά την υποβολή εργασιών. Η ιδιότητα Parameters μπορεί να είναι είτε ένας αριθμός, ή μια λίστα στοιχείων. Στην περίπτωση αυτή έχουμε μία λίστα στοιχείων και αυτά είναι τα EARTH,MARS,MOON.

Η υποβολή ενός τέτοιου αρχείου jdl αντιστοιχεί στην υποβολή τριών jdl σαν αυτό που ακολουθεί.

```
[
    Type = "job";
    JobType = "normal";
    Executable = "/bin/cat";
    StdInput = "inputvalue.txt";
    StdOutput = "myoutputvalue.txt";
    StdError = "myerrorvalue.txt";
    InputSandbox = "inputvalue.txt";
```

```
    OutputSandbox = {"myoutputvalue.txt","myerrorvalue.txt"};
]
```

Όπου value = "EARTH", "MARS", "MOON"

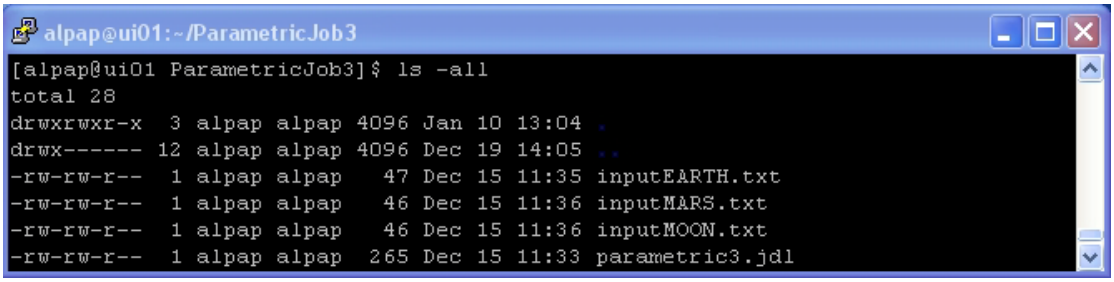
Με τον τρόπο αυτό ουσιαστικά γλυτώνουμε δύο αρχεία jdl.

Επιστρέφοντας στο αρχικό αρχείο βλέπουμε ότι στις ιδιότητες Attribute και InputSandbox υπάρχει το **input_PARAM_.txt** ,δηλαδή το αρχείο που θέλουμε να εμφανίσουμε για κάθε παράμετρο. Οπότε θα χρειαστεί να φτιάξουμε τρία αρχεία txt, ένα για κάθε παράμετρο. Το πρώτο θα είναι το **inputEARTH.txt** :

Testing of a parametric job.
Hello from Earth!

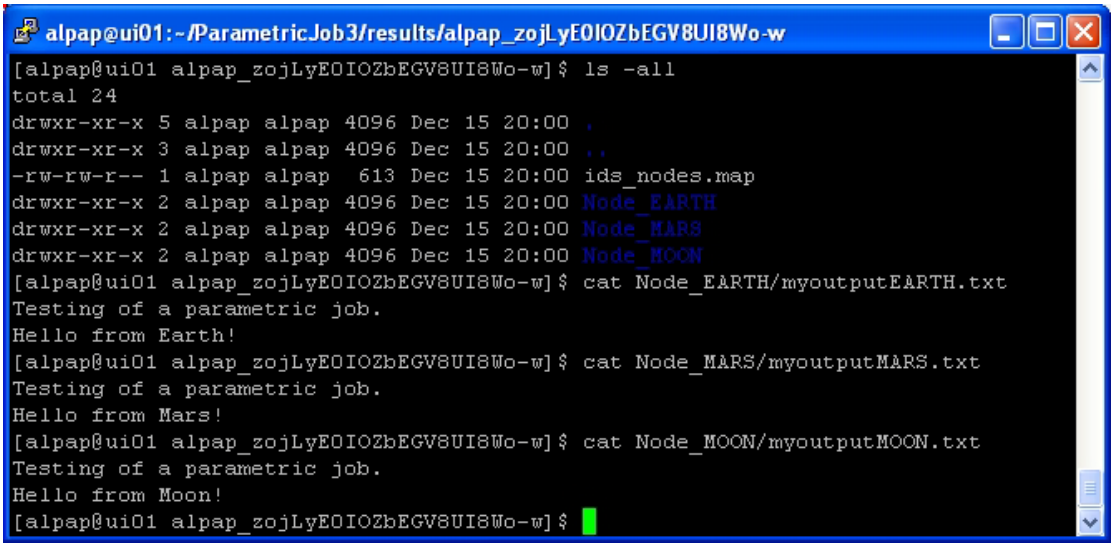
Αντίστοιχα για τα άλλα δύο **inputMARS.txt**, **inputMOON.txt** .

Άρα ο φάκελος με τα απαραίτητα αρχεία για την εργασία θα πρέπει να είναι κάπως έτσι:



Εικόνα 53 : Τα αρχεία μιας Parametric εργασίας

Υποβάλουμε την εργασία και αφού ολοκληρωθεί και ανακτήσουμε την έξοδο, βλέπουμε πως έχουμε αποτελέσματα από τρείς διαφορετικούς κόμβους (όσοι και οι παράμετροι). Τέλος χρησιμοποιούμε την εντολή **cat** για να δούμε το αποτέλεσμα της εξόδου σε κάθε κόμβο. [44]



Εικόνα 54 : Τα αποτελέσματα μιας Parametric εργασίας

Παράδειγμα 6^{ο} : Parametric Job 2

Ένας άλλος τρόπος υποβολής μιας Parametric εργασίας είναι χρησιμοποιώντας τις ιδιότητες **Parameters** , **ParameterStart** και **ParameterStep** τα οποία δηλώνουν αντίστοιχα, την μεγαλύτερη τιμή που μπορεί να πάρει η μεταβλητή, το βήμα με το οποίο αυξάνεται η τιμή της μεταβλητής και τέλος την αρχική της τιμή. Το πλήθος των εργασιών που θα εκτελεστεί δίνεται σύμφωνα με την παρακάτω σχέση.

Πλήθος εργασιών = (Parameters - ParameterStart) / ParameterStep

Η εργασία αυτή χρησιμοποιεί έναν κώδικα C++ που υπολογίζει το “π” χρησιμοποιώντας τη μέθοδο ολοκλήρωσης Monte Carlo. Με λίγα λόγια, παράγουμε ένα σύνολο τυχαίων σημείων N σε ένα εμβαδόν του καρτεσιανού επιπέδου. Στη συνέχεια μετράει και κρατάει στη μνήμη τον αριθμό των σημείων που ανήκουν στο εσωτερικό της ακτίνας του κύκλου και εκχωρεί αυτό τον αριθμό σε μια μεταβλητή N_c. Η τιμή του “π” υπολογίζεται κατά προσέγγιση και δίνεται μέσω της σχέσης

$\pi = 4 * Nc / N.$

Στην περίπτωση αυτή ονομάζουμε το αρχείο **Parametric_ci.cpp** και ο κώδικας είναι ο εξής:

```
#include <cmath>
#include <algorithm>
#include <iostream>

using namespace std;

class point
{
private:
  double x,y;
public:
  point(){};
  point(double xx,double yy){  x = xx; y = yy; }
  ~point(){};
  double magn()   const{ return sqrt( x*x + y*y );}
};

double drand(){
  return (double) rand()/RAND_MAX;
}

point prand(){   // returns a random point inside [0,1)x[0,1)
  point tmp(drand(),drand());
  return tmp;
}

int main( int argc, char *argv[] )
{
  int N = 1000000;
  int N_c = 0;
  int seed = 0;

  seed=atoi(argv[1]);
  printf("seed=%d\n",seed);
  srand( seed );   // set the seed
  for(int i=0; i<N; i++){
    if( prand().magn() < 1.0 )   N_c++;
  }
  cout<< 4.0*((double) N_c)/((double) N)<<endl;
  return 0;
}
```

Χρησιμοποιούμε ένα επιπλέον script που θα ονομάσουμε **parametric_run.sh** για την κατάρτιση του παραπάνω κώδικα. Αυτό που θα κάνει, είναι να μεταγλωττίσει (compile) το **Parametric_ci.cpp** και θα δημιουργήσει το εκτελέσιμο **Parametric_ci** στους απομακρυσμένους Worker Nodes (WNs), βοηθώντας τους κατά κάποιο τρόπο να το εντοπίσουν και να το εκτελέσουν.

```
#!/bin/bash
g++ -o Parametric_pi `pwd`/Parametric_pi.cpp
`pwd`/Parametric_pi $1
```

Με το “g++” δηλώνουμε οτί το αρχείο που θέλουμε να εκτελέσουμε είναι γραμμένο σε C++. Στην πραγματικότητα, η τελευταία γραμμή δεν αφορά το path για το εκτελέσιμο στον κατάλογο του χρήστη, αλλά στον κατάλογο του WN, όπου το εκτελέσιμο αρχείο έχει φορτωθεί. Το αρχείο αυτό θα χρησιμοποιηθεί στην ιδιότητα executable στο jdl αρχείο. Εάν βάζαμε το **Parametric_ci.cpp** ο WN δεν θα μπορούσε να το διαβάσει. [45]

Τέλος το jdl αρχείο parametric_run.jdl :

```
JobType = "Parametric";
Executable = "parametric_run.sh";
Arguments = "_PARAM_";
StdOutput = "std_PARAM_.out";
StdError = "std_PARAM_.err";
Parameters = 4;
ParameterStart = 1;
ParameterStep = 1;
InputSandbox = {"parametric_run.sh","Parametric_pi.cpp"};
OutputSandbox = {"std_PARAM_.out","std_PARAM_.err" };
```

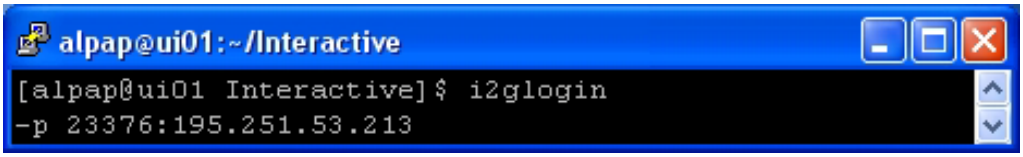
Άρα οι εργασίες που θα εκτελεστούν είναι :

Πλήθος εργασιών = (Parameters - ParameterStart) / ParameterStep = (4-1)/1 = 3

Παράδειγμα 7^{ο} : Interactive Job

Μία Interactive (διαδραστική) εργασία είναι αυτή όπου ο χρήστης μπορεί να ανοίξει έναν SSH δίαυλο επικοινωνίας ανάμεσα στο UI του και σε έναν απομακρυσμένο Worker Node, σε πραγματικό χρόνο.

Αυτό γίνεται με την εντολή **i2glogin** ή **/opt/i2g/bin/i2glogin**. Το διαδραστικό αμφίδρομο κανάλι που παρέχεται από τη i2glogin συνδέει οποιαδήποτε εφαρμογή που τρέχει στο Grid με το τερματικό του χρήστη. Η εντολή αυτή επιστρέφει μία port (θύρα) και μία IP διεύθυνση όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 55 : Η εντολή i2glogin

Η TCP-θύρα αυτή (23376) επιλέγεται τυχαία από το port-range (δηλ. το εύρος θυρών) του GridFTP και μέσω αυτής θα ανοίξει το κανάλι επικοινωνίας. Η διεύθυνση IP (195.251.53.213) είναι του UI του χρήστη, ώστε να ξέρει ο WN με ποιόν θα επικοινωνήσει.

Για μία Interactive εργασία, χρειάζονται δύο παράθυρα στο puTTY. Το πρώτο είναι το παραπάνω όπου ζητείται να ανοίξει η επικοινωνία. Στο δεύτερο θα υποβληθεί η εργασία με το jdl αρχείο και όταν αρχίσει να εκτελείται θα γίνει η σύνδεση.

Το jdl αρχείο που σε αυτή την περίπτωση ονομάζεται interactive.jdl είναι το εξής :

```
JobType = "normal";
Executable = "job.sh";
Arguments = "-p 22486:195.251.53.213";
InputSandbox = {"/opt/i2g/bin/i2glogin", "job.sh"};
```

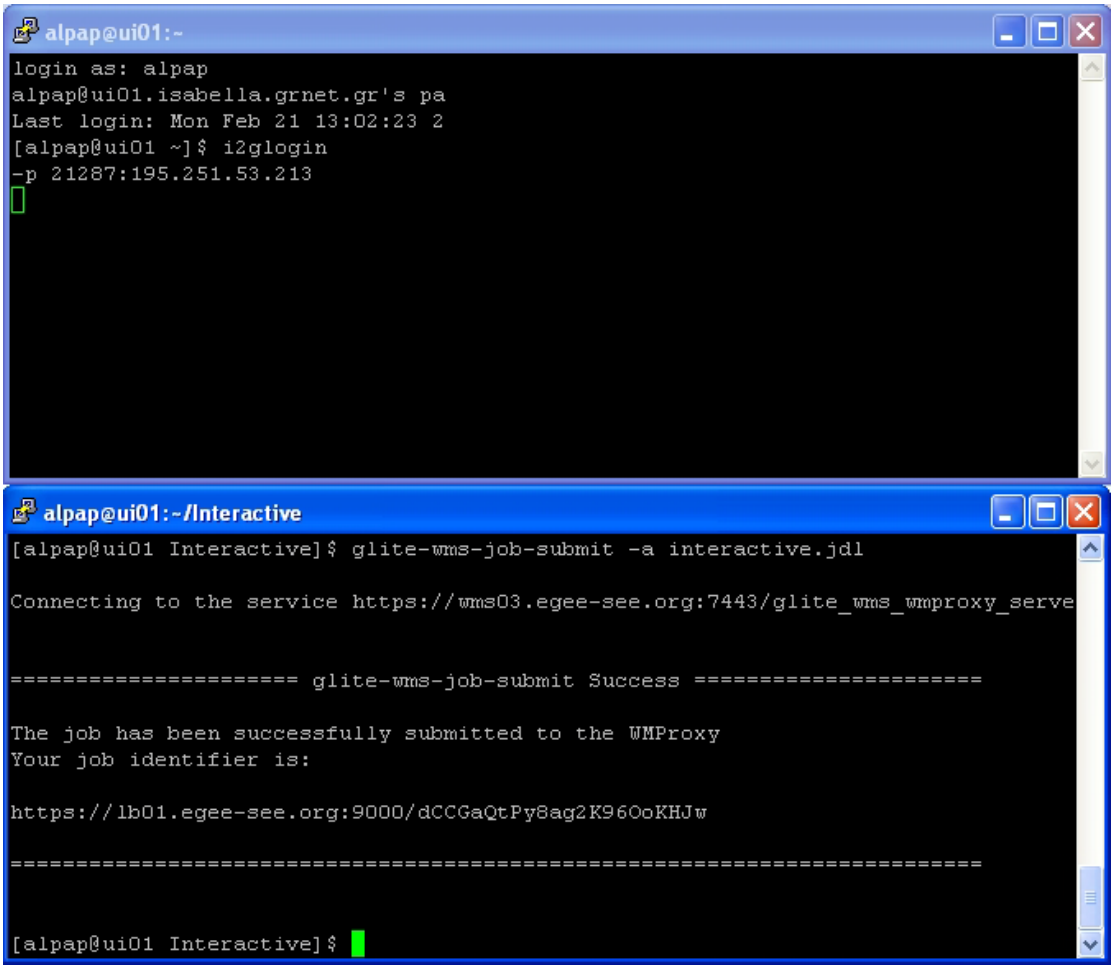
Όπως φαίνεται, στην ιδιότητα Arguments θα πρέπει να γράψουμε την port και την IP που έδωσε η εντολή i2glogin στο πρώτο παράθυρο (προσοχή : δεν πρέπει να γίνει copy με ctrl+c στο πρώτο παράθυρο γιατί θα τερματιστεί η σύνδεση). Στο InputSandbox στέλνουμε το εκτελέσιμο job.sh και την εντολή /opt/i2g/bin/i2glogin η οποία χρησιμοποιείται στο εκτελέσιμο όπως θα φανεί παρακάτω για να εκτελεστεί στον WN έτσι να επιτευχθεί η σύνδεση.

Το εκτελέσιμο job.sh είναι :

```
#!/bin/sh
chmod +x i2glogin
./i2glogin -r $1 $2
echo "Welcome!"
echo -n "Please tell me your name: "
read name
echo "That is all, $name."
echo "Bye bye."
exit 0
```

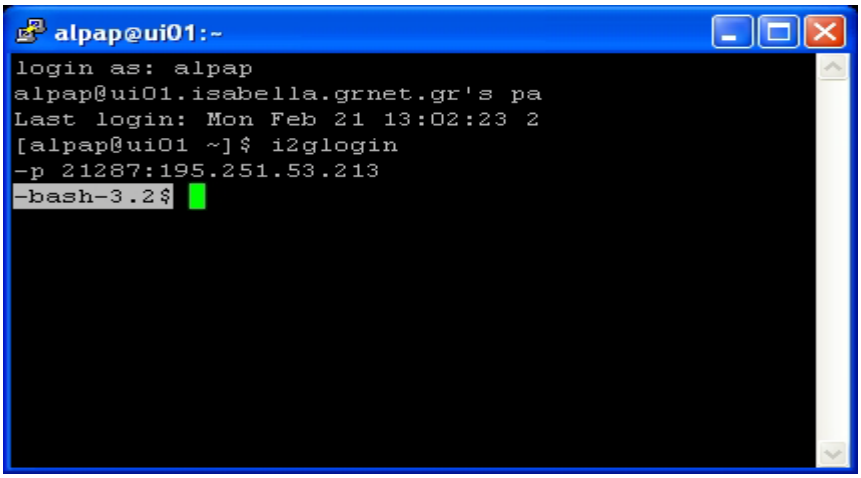
Με τις δύο πρώτες γραμμές του script η i2glogin που στείλαμε με το InputSandbox γίνεται εκτελέσιμη και έπειτα εκτελείται με τα arguments (μεταβλητές) που δώσαμε στο jdl αρχείο, δηλαδή το port και την IP. Έτσι κατευθύνουμε τον WN στην επικοινωνία που έχουμε ανοίξει και του ζητάμε να συνδεθεί με αυτήν. Έπειτα θα εκτελεστεί το υπόλοιπο πρόγραμμα, που ζητάει απλά το όνομα του χρήστη και του επιστρέφει ένα μήνυμα. [46]

Η υποβολή γίνεται με την εντολή **glite-wms-job-submit -a interactive.jdl** [δεν χρειάζεται να δώσουμε την ταυτότητα της εργασίας (-o interactive.jid) όπως γινόταν στις προηγούμενες εργασίες αφού η output θα εμφανιστεί στο παράθυρο επικοινωνίας]. Τα δύο παράθυρα θα πρέπει να είναι κάπως έτσι :



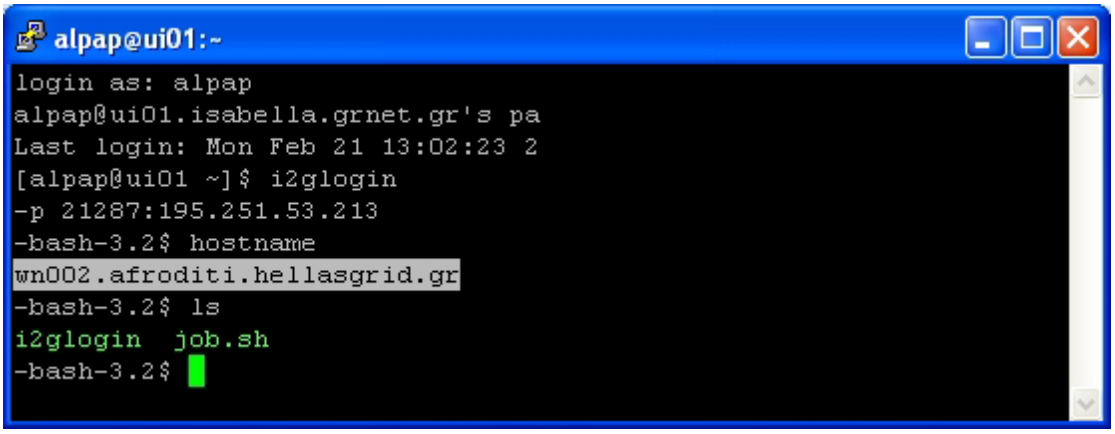
Εικόνα 56 : Τα παράθυρα επικοινωνίας και υποβολής

Μετά από περίπου 30 δευτερόλεπτα και όταν αρχίσει να εκτελείται η εργασία, θα εμφανιστεί στο παράθυρο επικοινωνίας το prompt του worker node. Αυτό σημαίνει ότι ξεκίνησε η σύνδεση.



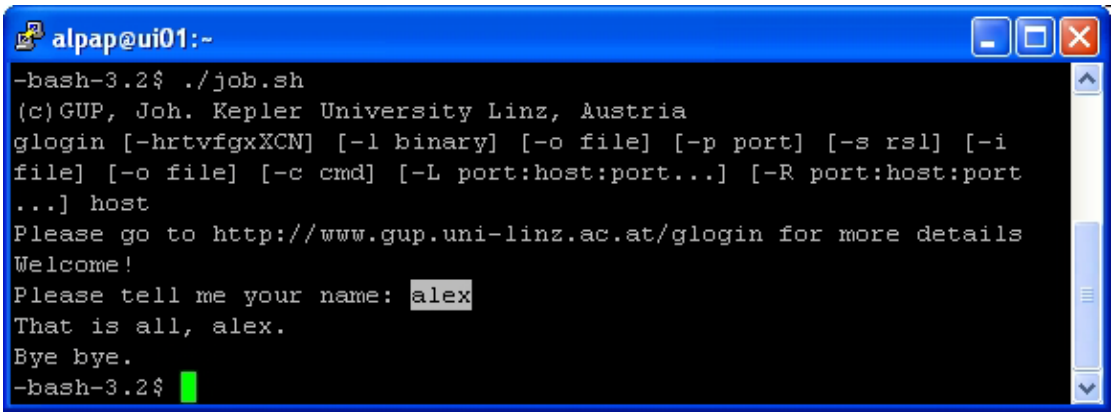
Εικόνα 57 : Έναρξή επικοινωνίας με τον WN

Πληκτρολογώντας **hostname**, μπορεί ο χρήστης να δει σε ποιόν WN έχει συνδεθεί και με ls τα αρχεία που έχει ανεβάσει με το InputSandbox. Ο WN επιλέγεται με βάση την διαθεσιμότητα και την απόσταση από τον χρήστη.



Εικόνα 58 : Οι εντολές hostname και ls

Τέλος για να εκτελεστεί η εργασία πληκτρολογούμε **./job.sh** ,εκτελείται το script στον WN, εμφανίζει το μήνυμα που ζητάει το όνομα και επιστρέφει έπειτα το τελικό μήνυμα.



Εικόνα 59 : Εκτέλεση του script και επικοινωνία με τον χρήστη

Για τον τερματισμό της σύνδεσης πληκτρολογούμε exit.

Παράδειγμα 8^{ο} : MPICH Job

Μία εργασία MPI (Message Passing Interface) χρησιμοποιείται για παράλληλο προγραμματισμό. Το gLite-WMS υποστηρίζει την υποβολή MPI, που αποτελείται από έναν αριθμό επεξεργαστών που λειτουργούν σε διαφορετικούς Worker Nodes, όλοι μέρος της ίδιας συστάδας (Computing Element).

Ο χρήστης χρησιμοποιεί συνήθως ένα wrapper script (ένα script που ενσωματώνει μια εντολή στο σύστημα ή χρησιμότητα, που αποθηκεύει ένα σύνολο παραμέτρων και την περνάει στην εν λόγω εντολή) για την εκκίνηση της επεξεργασίας MPI-START. Το ακόλουθο script wrapper με όνομα mpi-start-wrapper.sh είναι γενικό και οι χρήστες δεν θα χρειαστεί να κάνουν σημαντικές τροποποιήσεις σε αυτό, σε περίπτωση που θέλουν να τροποποιήσουν την εργασία.

Το wrapper script "**mpi-start-wrapper.sh**" είναι :

```
#!/bin/bash

# Pull in the arguments.
MY_EXECUTABLE=`pwd`/$1
MPI_FLAVOR=$2

# Convert flavor to lowercase for passing to mpi-start.
MPI_FLAVOR_LOWER=`echo $MPI_FLAVOR | tr '[:upper:]' '[:lower:]'`

# Pull out the correct paths for the requested flavor.
eval MPI_PATH=`printenv MPI_${MPI_FLAVOR}_PATH`

# Ensure the prefix is correctly set. Don't rely on the defaults.
eval I2G_${MPI_FLAVOR}_PREFIX=$MPI_PATH
export I2G_${MPI_FLAVOR}_PREFIX

# Touch the executable. It exist must for the shared file system check.
# If it does not, then mpi-start may try to distribute the executable
# when it shouldn't.
touch $MY_EXECUTABLE

# Setup for mpi-start.
export I2G_MPI_APPLICATION=$MY_EXECUTABLE
```

```
export I2G_MPI_APPLICATION_ARGS=
export I2G_MPI_TYPE=$MPI_FLAVOR_LOWER
export I2G_MPI_PRE_RUN_HOOK=mpi-hooks.sh
export I2G_MPI_POST_RUN_HOOK=mpi-hooks.sh

# If these are set then you will get more debugging information.
export I2G_MPI_START_VERBOSE=1
#export I2G_MPI_START_DEBUG=1

# Invoke mpi-start.
$I2G_MPI_START
```

Το script ενεργοποιεί, το περιβάλλον για την επιλεγμένη εργασία του MPI χρησιμοποιώντας μεταβλητές περιβάλλοντος που παρέχονται από το διαχειριστή του συστήματος. Ορίζει τότε το εκτελέσιμο, τα επιχειρήματα και την τοποθεσία των hook scripts (χρησιμοποιούνται για την κατάρτιση του executable σε εκτελέσιμο) για το MPI-START. Το wrapper συμπεριλαμβάνεται στο InputSandbox της εργασίας στο αρχείο JDL.

Έπειτα ορίζεται ένα άλλο script, μια συνάρτηση που καλείται πριν και μετά την εκτέλεση του MPI executable. Για παράδειγμα, η λειτουργία pre-hook μπορεί να χρησιμοποιηθεί για την κατάρτιση (compile) του ίδιου εκτελέσιμου ή για το κατέβασμα των δεδομένων εισόδου. Η λειτουργία post-hook μπορεί να χρησιμοποιηθεί για να αναλύσει τα αποτελέσματα ή να τα αποθηκεύσει στο δίκτυο.

Το ακόλουθο script mpi-hooks.sh καταρτίζει (compile) το εκτελέσιμο πριν από την εκτέλεση του. Η λειτουργία post-hook εκτυπώνει μόνο ένα μήνυμα προς την έξοδο.

Το script “**mpi-hooks.sh**” είναι :

```
#!/bin/sh

#
# This function will be called before the MPI executable is started.
# You can, for example, compile the executable itself.
#
```

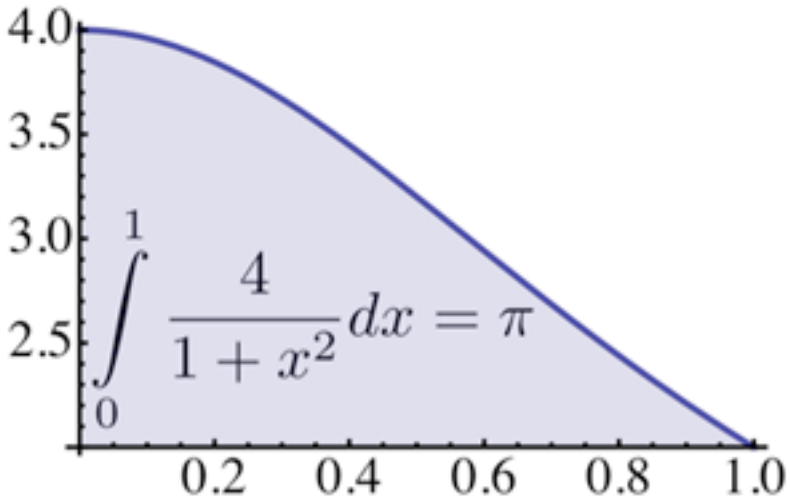
```
pre_run_hook () {

  # Compile the program.
  echo "Compiling ${I2G_MPI_APPLICATION}"

  # Actually compile the program.
  cmd="mpicc ${MPI_MPICC_OPTS} -o ${I2G_MPI_APPLICATION}
${I2G_MPI_APPLICATION}.c"
  echo $cmd
  $cmd
  if [ ! $? -eq 0 ]; then
    echo "Error compiling program. Exiting..."
    exit 1
  fi

  # Everything's OK.
  echo "Successfully compiled ${I2G_MPI_APPLICATION}"

  return 0
}

#
# This function will be called before the MPI executable is finished.
# A typical case for this is to upload the results to a storage element.
#
post_run_hook () {

  echo "Executing post hook."
  echo "Finished the post hook."

  return 0
}
```

Στο αρχείο JDL το JobType τίθεται σε “Normal” και ο χρήστης μπορεί να ορίσει το CPUNumber με τον αριθμό των επιθυμητών πυρήνων CPU επεξεργαστών. Το εκτελέσιμο είναι το wrapper-script “mpi-start-wrapper.sh” και τα arguments είναι το είδος MPI εργασίας που θέλουμε να υποβάλουμε (MPICH, MPICH2, OPENMPI). Στην περίπτωση αυτή είναι το MPICH. Η ιδιότητα Requirements διευκρινίζει ότι το MPI-START και το είδος της MPI εργασίας θα πρέπει να υποστηρίζονται από το site Grid όπου θα εκτελεστεί η εργασία.

Το jdl αρχείο “**mpi-start.jdl**”είναι :

```
JobType = "Normal";
CPUNumber = 3;
Executable = "mpi-start-wrapper.sh";
Arguments = "pi MPICH";
StdOutput = "mpi-start.out";
StdError = "mpi-start.err";
InputSandbox = {"mpi-start-wrapper.sh","mpi-hooks.sh","pi.c"};
OutputSandbox = {"mpi-start.err","mpi-start.out"};
Requirements =
  Member("MPI-START", other.GlueHostApplicationSoftwareRunTimeEnvironment)
  && Member("OPENMPI", other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

Η εργασία αυτή υπολογίζει μία περιοχή/εμβαδό ολοκληρώνοντας : 4/(1 + x2) στο διάστημα [0,1] το οποίο ισούται με “π”. Το παρακάτω σχήμα απεικονίζει το πρόβλημα:



Εικόνα 60 : Το εμβαδό που υπολογίζεται

Το πρόγραμμα είναι σε γλώσσα προγραμματισμού C και το script “**pi.c**” είναι :

```
# include <stdio.h>
# include <math.h>
# include "mpi.h"

/* Evaluation of the inregration function
  f(x)=1/(1+x*x) (0<=x<=1) */
double f(double x)
{
  double value;
  value=4.0/(1.0+x*x);
  return value;
}

int main (int argc, char *argv[])
{
  int counter;
  double walltime_start,walltime_end,walltime_diff;

  int master_node = 0;
  int number_of_nodes;
  int node_id;

  int number_of_intervals=300000000;
  double node_integral;
  double step;

  double x;

  double pi_estimate;
  double pi_diff;
  double pi_exact = 3.14159265358979323846264 3;

/* Establish the MPI environment */
  MPI_Init (&argc, &argv);
/* Get the number of processes */
  MPI_Comm_size (MPI_COMM_WORLD,&number_of_nodes);
/* Determine this processes's rank */
  MPI_Comm_rank (MPI_COMM_WORLD,&node_id);

  if (node_id==master_node)
  {
    printf("Compiled on %s at %s\n",__DATE__, __TIME__);
    printf("Number of nodes: %d\n",number_of_nodes);
    printf ( "Number of intervals: %d\n",number_of_intervals);
```

```
    walltime_start=MPI_Wtime();
  }

  MPI_Bcast (&number_of_nodes,1,MPI_INT,master_node,MPI_COMM_WORLD );

  step=1.0/(double)number_of_intervals;
  node_integral=0.0;
  for
(counter=node_id+1;counter<=number_of_intervals;counter=counter+number_of_nod
es)
  {
    x=step*((double)counter-0.5);
    node_integral=node_integral+f(x);
  }

  printf("\nNode: %d\n",node_id);
  printf("Node integral: %f\n",node_integral*step);

  MPI_Reduce
(&node_integral,&pi_estimate,1,MPI_DOUBLE,MPI_SUM,master_node,MPI_COMM_W
ORLD);

/* The master node prints the answer */
  if (node_id==master_node)
  {
    pi_estimate=pi_estimate*step;
    pi_diff=fabs(pi_estimate-pi_exact);
    printf("PI exact   : %24.16f\n", pi_exact);
    printf("PI estimate: %24.16f\n", pi_estimate);
    printf("PI diff    : %24.16f\n", pi_diff);

    walltime_end=MPI_Wtime();
    walltime_diff=walltime_end-walltime_start;
    printf ("Wall clock: %f\n", walltime_diff );
  }

/* Shut down MPI */
  MPI_Finalize();
}
```

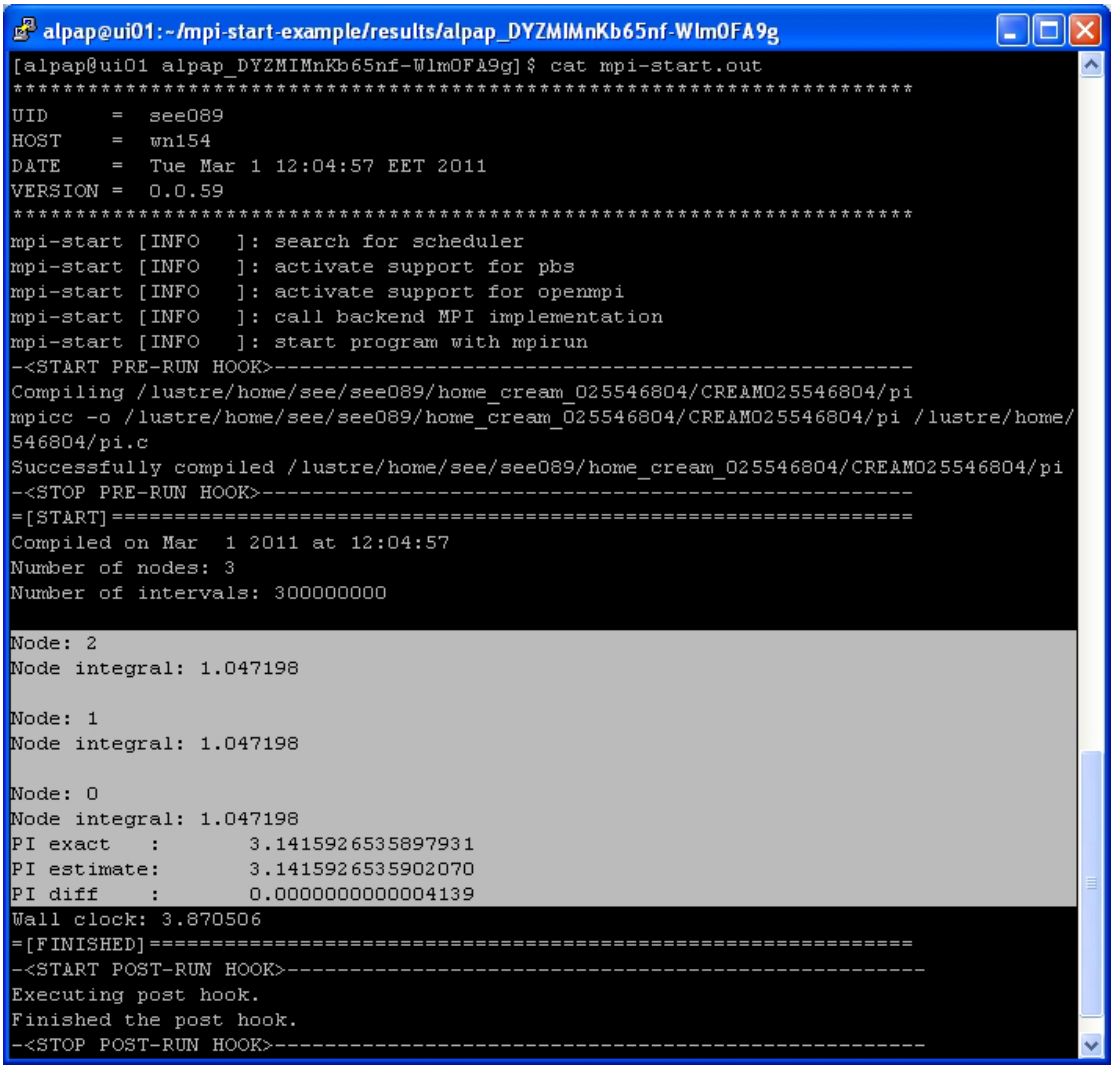
Υποβάλουμε την εργασία με **:**

glite-wms-job-submit -a -o mpi.jid mpi-start.jdl

και αφού εκτελεστεί ανακτούμε την έξοδο με την εντολή :

glite-wms-job-output --dir ./results -i mpi.jid

Τέλος για την εμφάνιση του αποτελέσματος : **cat mpi-start.out**



Εικόνα 61 : Τα αποτελέσματα της MPI εργασίας από τις τρείς διαφορετικές CPU

Παραπάνω φαίνεται η χρήση των τριών CPU. [47]

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] http://www.gsrt.gr/default.asp?V_ITEM_ID=2680

[2] http://antognini.ch/papers/GridComputing_20040723.pdf

[3] http://neohumanism.org/c/cp/cpu_scavenging.html

[4] http://www.gridcafe.org/version1/Gridhistory/key.html

[5] http://www.gridcafe.org/version1/Gridhistory/backtofuture.html

[6] http://www.eurogrid.org/

[7] http://eu-datagrid.web.cern.ch/eu-datagrid/

[8] http://www.see-grid-sci.eu/

[9] http://www.grnet.gr/default.asp?pid=89&la=1

[10] http://nes.aueb.gr/~sroutz/grid-project/greece.htm

[11] http://goc.grid.auth.gr/wiki/bin/view/Groups/ALL/GridUserRegistration#Certificate_backup

[12] http://www.webopedia.com/TERM/U/user_interface.html

[13] http://iag.iucc.ac.il/workshop/

[14] http://kb.mediatemple.net/questions/247/Common+SSH+Commands

[15] http://www-numi.fnal.gov/offline_software/srt_public_context/GridTools/docs/jobs_tutorial.html#proxies

[16] http://el.wikipedia.org/wiki/File_Transfer_Protocol

[17] http://msdn.microsoft.com/en-us/library/cc219661%28PROT.10%29.aspx

[18] http://el.wikipedia.org/wiki/Packet_sniffer

[19] http://el.wikipedia.org/wiki/Brute-force_attack

[20] http://en.wikipedia.org/wiki/GridFTP

[21] http://el.wikipedia.org/wiki/SSL

[22] http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci214091,00.html

[23] http://www.smartftp.com/support/kb/what-is-fxp-f14.html

[24]http://www.tcpipguide.com/free/t_TCPWindowSizeAdjustmentandFlowControl.htm

[25] http://glite.web.cern.ch/glite/wms/

[26] http://www.egee-see.org/content/modules/downloads/EGEE_Greek_News_Dec07.pdf

[27] http://egee-uig.web.cern.ch/egee-uig/production_pages/SimpleJobCycle.html

[28] http://grid.desy.de/testbed/EDG/CE.html

[29] http://cmsdoc.cern.ch/cms/grid/docs/DataLocationInterface.pdf

[30] http://www-numi.fnal.gov/offline_software/srt_public_context/GridTools/docs/glossary.html#is

[31] http://www-numi.fnal.gov/offline_software/srt_public_context/GridTools/docs/glossary.html#se

[32] http://www.euasiagrid.org/training/gLite_WMS_09.pdf

[33] http://en.wikipedia.org/wiki/Command-line_interface

[34] http://www-numi.fnal.gov/offline_software/srt_public_context/GridTools/docs/glossary.html

[35] http://doc.escience-lab.org/index.php/Grid/BDII

[36] http://www.youtube.com/watch?v=697xcHf1RZM

[37] http://en.wikipedia.org/wiki/Parsing

[38] http://www.grid.org.tr/servisler/dokumanlar/JDL_Atributes_DataGrid.pdf

[39] http://www.linfo.org/tail.html

[40] http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci1110704,00.html

[41] http://www.eu-egee.org/fileadmin/documents/UseCases/SimpleJobCycle.html paradeigma1

[42] http://www.youtube.com/watch?v=1XQ7oxo_H-c

[43] http://www.youtube.com/watch?v=KARPM_nPy1M&feature=related

[44] http://doc.escience-lab.org/index.php/Grid/AdvancedJobSubmission#parametric

[45]http://wiki.hellasgrid.gr/wiki/bin/view/HellasGrid/GOC/MultipleJobSubmissionn#Parametric_job_submission

[46] http://grid.ifca.es/wiki/Middleware/i2glogin#CA-5bb43ac097d8ccbf1831781d9642329e6b3998a3_3

[47] http://wiki.egee-see.org/index.php/SEE-GRID_MPI_User_Guide