ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός και ανάπτυξη τεχνικών μέτρησης Πόρων και Υπηρεσιών σε Συστήματα Πλέγματος (Grid)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΝΙΚΟΛΑΟΥ Ι. ΜΑΚΡΗ

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιανουάριος 2009

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός και Ανάπτυξη Τεχνικών Μέτρησης Πόρων και Υπηρεσιών σε Συστήματα Πλέγματος (Grid)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΝΙΚΟΛΑΟΥ Ι. ΜΑΚΡΗ

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31η Ιουλίου 2008.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Θεοδώρα Βαρβαρίγου Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιανουάριος 2009

(Υπογραφή)

.....

ΝΙΚΟΛΑΟΣ Ι. ΜΑΚΡΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Για την περάτωση της παρούσας διπλωματικής, θέλω να ευχαριστήσω τους καθηγητές της τριμελούς επιτροπής και ιδιαίτερα την επιβλέπουσα καθηγήτρια κ. Θεοδώρα Βαρβαρίγου.

Περίληψη

Σήμερα πολλοί είναι οι χρήστες που αναζητούν όλο και περισσότερες υπολογιστικούς πόρους για την επίλυση προβλημάτων που ζητούν πολύπλοκους υπολογισμούς. Παρατηρείται όμως το εξής φαινόμενο ότι η μεμονωμένη χρήση υπολογιστών έχει ως αποτέλεσμα την υπερφόρτωση του συστήματος, την επανεκκίνηση του σε κάποιες περιπτώσεις , αλλά και την μακρόχρονη εκτέλεση υπολογισμών που ενδέχεται να κρατήσουν μέρες ή ακόμα και εβδομάδες μειώνοντας με αυτό τον τρόπο την παραγωγικότητα. Η αναζήτηση για μια λύση στο παραπάνω πρόβλημα απαίτηση οδήγησε στην δημιουργία κατανεμημένων υπολογιστικών συστημάτων με την ονομασία Grid .

Σκοπός της διπλωματικής είναι η ανάπτυξη ενός προγράμματος σε γλώσσα Java το οποίο είναι υπεύθυνο για την καταγραφή των διαθέσιμων υπολογιστικών και αποθηκευτικών πόρων μιας υπολογιστικής μονάδας , δηλαδή ενός υπολογιστή. Το πρόγραμμα αυτό λειτουργεί σε περιβάλλον Windows χρησιμοποιώντας dlls και ένα Software Widget Toolkit με την ονομασία Win32 SWT.

Στην διπλωματική εργασία έχει γίνει ανάλυση των τεχνολογιών Grid, OGSA, WSRF , Web Services, Globus καταλήγοντας στο ζητούμενο Java πρόγραμμα, δίνοντας στον αναγνώστη μια σφαιρική εικόνα του τρόπου λειτουργίας κατανεμημένου συστήματος Grid, με συνέπεια την καλύτερη κατανόηση λειτουργίας του προγράμματος αυτού.

Λέξεις Κλειδιά:

Grid	Κατανεμημένο υπολογιστικό συστημάτων
Java	Αντικειμενοστραφής γλώσσα προγραμματισμού
Jar	Τύπος αρχείου βιβλιοθήκης Java
CPU	Επεξεργαστής
RAM	Μνήμη τυχαίας προσπέλασης
DLL	Τύπος αρχείου βιβλιοθήκης Windows.
WSRF	Web Services Resource Framework
OGSA	Open Grid Services Architecture
platform independent	Ανεξάρτητο λογισμικού λειτουργικού συστήματος
native	Τύπος προκατασκευασμένης μεθόδου η οποία σχετίζεται με το λειτουργικό σύστημα Windows. Υπάρχει στο Win32 SWT

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

In our times many are the users that are in constant research of computable resources regarding the solution of problems associated with complicated calculations. Continuous reboots, system overload, time consuming problems that the chance of their calculations reach to an end might need days or even weeks is a phenomenon that has to do with the use of individual computers. All these events lead to low productivity of any company. The solution to the problem mentioned above was given with the development of distributed computing systems named as Grids.

The purpose of this thesis is the development of a Java program which is responsible for monitoring the free and therefore the used resources which are consumed by each computer individually. This program is used to monitor resources in the Windows Operational System and uses dlls for this purpose and a Software Widget Toolkit known as Win32 SWT.

Analysis of the technologies Grid, OGSA, WSRF, Web Services, Globus was considered necessary so as the reader will have better comprehension of the Java Program which is placed in the end of this thesis.

Keywords:

Grid	Distributed Computing System
Java	Object Oriented Language
Jar	Java Library
CPU	Central Processor Unit
RAM	Random Access Memory
DLL	Τύπος αρχείου βιβλιοθήκης Windows.
WSRF	Web Services Resource Framework
OGSA	Open Grid Services Architecture
platform independent	Software that is used or executed in all known Operating Systems
native	Prebuild Method that is associated with the Operating System. These kind of methods exist in the Win32 SWT

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

Εικόνες

1 *Εισαγωγή*

1.1 Τεχνολογίες Πλέγματος

Συνεχώς αυξανόμενες γίνονται οι ανάγκες στις μέρες μας για αναζήτηση περισσότερων πόρων από τη στιγμή που τα προβλήματα γίνονται όλο και περισσότερα, όλο και πιο πολύπλοκα. Αυτή η ζήτηση στρέφει το ενδιαφέρον προς τα κατανεμημένα συστήματα τα οποία μπορούν να παρέχουν αφθονία τέτοιων πόρων και ειδικότερα τα συστήματα Πλέγματος, γνωστά ως Grids. Ωστόσο κρίνεται αναγκαίο η μέτρηση των διαθέσιμων πόρων αλλά και των πόρων που απασχολούνται σε ένα τέτοιο σύστημα.

1.2 Αντικείμενο διπλωματικής

Η διπλωματική αύτη εργασία πραγματεύεται την ανάπτυξη τεχνικής μέτρησης η οποία θα είναι υπεύθυνη για την καταμέτρηση πόρων της εκάστοτε υπολογιστικής μονάδας. Πιο συγκεκριμένα η τεχνική αυτή απασχολείται με τη χρήση του επεξεργαστή (CPU Usage), με τη διαθέσιμη μνήμη (RAM) και με το διαθέσιμο αποθηκευτικό χώρο (Disk Storage) για κάθε στιγμή που καλείται να ληφθεί πληροφορία για τις συγκεκριμένες παραμέτρους.

Η τεχνική υλοποιείται με τη μορφή βιβλιοθήκης Java (JAR) το οποίο μπορεί να χρησιμοποιηθεί ως εξωτερική βιβλιοθήκη ή ακόμα και να εκτελεστεί αυτοδύναμα. Ωστόσο μια τέτοια υλοποίηση είχε και κάποια προβλήματα να αντιμετωπίσει τα οποία και παραθέτονται.

Το μεγαλύτερο πρόβλημα για αυτή την υλοποίηση είναι ότι χρειαζόταν να γίνει με βάση τη γλώσσα προγραμματισμού Java. Η χρήση της γλώσσας Java είναι ευρέως διαδεδομένη καθώς έχει την μοναδική ιδιότητα να είναι ανεξάρτητη πλατφόρμας (platform independent) και ο αριθμός εφαρμογών που βασίζονται σε αυτή είναι συνεχώς αυξανόμενος. Ωστόσο αυτή η ιδιότητα της την καθιστά αδύναμη για συλλογή ειδικών πληροφοριών από κάθε λειτουργικό σύστημα.

Για το λόγο αυτό χρειάστηκε να χρησιμοποιηθεί κάποια εξωτερική βιβλιοθήκη η οποία παρέχει ειδικές πληροφορίες για το λειτουργικό σύστημα, στην προκειμένη περίπτωση Windows και μέσω της οποίας το πρόγραμμα σε Java θα ήταν σε θέση να τις αποδεσμεύσει με κατάλληλη χρήση μεθόδων. Η εξωτερική αυτή βιβλιοθήκη ονομάζεται Win32 SWT.

Για την καλύτερη κατανόηση της όλης διαδικασίας μέτρησης πόρων σε ένα κατανεμημένο υπολογιστικό σύστημα κρίθηκε απαραίτητη η παράθεση και ανάλυση τεχνολογιών όπως Grid, OGSA, WSRF, Web Services, Globus toolkit.

1.2.1 Συνεισφορά

Η διπλωματική εργασία αφορά την υλοποίηση ενός προγράμματος το οποίο θα καταγράφει για κάθε χρονική στιγμή τα resources τα οποία χρησιμοποιεί η κάθε υπολογιστική μονάδα ενός πλέγματος Grid σε περιβάλλον Windows. Η δυσκολία αυτού του εγχειρήματος έγκειται στο ότι το πρόγραμμα αυτό έπρεπε να υλοποιηθεί με βάση την γλώσσα προγραμματισμού Java η οποία δεν περιλαμβάνει native μεθόδους για κανένα λειτουργικό σύστημα που θα βοηθούσαν στην αποδέσμευση των επιθυμητών πληροφοριών.

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελέτη για το αν η Java είχε κάποιο API από το οποίο θα περιελάμβανε κλήσεις σε native μεθόδους. Προς το παρόν δεν παρέχεται κάτι τέτοιο από την Java.
2. Έρευνα στο internet για API το οποίο θα βοηθούσε στην λήψη πληροφοριών για τα resources σε περιβάλλον Windows.
3. Εύρεση ως καταλληλότερου και πιο ολοκληρωμένου API το Win32SWT.
4. Ενσωμάτωση των μεθόδων που χρησιμοποιεί το Win32SWT σε κώδικα γραμμένο κατάλληλα για την αποδέσμευση αυτών των πληροφοριών.
5. Μετά τη σωστή λειτουργία για μονοπύρηνους επεξεργαστές, έλεγχος για υποστήριξη μέσω του Win32SWT API για διπύρηνους και τετραπύρηνους επεξεργαστές.

1.3 Οργάνωση κειμένου

Στην διπλωματική εργασία η αρχή γίνεται με μια εισαγωγή στο **Κεφάλαιο 1** για τις τρέχουσες τεχνολογίες στο χώρο των κατανεμημένων υπολογιστικών συστημάτων και αναφορά της έννοιας των Grid υπολογιστικών συστημάτων. Στο **Κεφάλαιο 2** γίνεται περαιτέρω ανάλυση για τον τρόπο λειτουργίας αλλά και τα είδη των Grid. Τα κατανεμημένα συστήματα απαιτούν και εύλογα ένα κοινό πρότυπο για τις εφαρμογές που βασίζονται στην τεχνολογία Grid, στην προκειμένη περίπτωση το OGSA που η λειτουργία του περιγράφεται στο **Κεφάλαιο 3**. Είναι προφανές ότι ένα τέτοιο κατανεμημένο σύστημα έχει και ως λειτουργία την επικοινωνία με άλλες υπολογιστικές μονάδες ή χρήστες. Αυτό επιτυγχάνεται με την τεχνολογία των Web Services τα οποία και

αναλύονται στο **Κεφάλαιο 4.** Τα Web Services από μόνα τους δεν κρίνονται ικανά για την υποστήριξη των Grid συστημάτων για αυτό και κρίθηκε απαραίτητο η υλοποίηση ενός framework μέσω του οποίου θα γινόταν αυτή η επικοινωνία με την τεχνολογία των Web Services, το WSRF για το οποίο περισσότερες πληροφορίες αναγράφονται στο **Κεφάλαιο 5.** Ένα κατανεμημένο υπολογιστικό σύστημα απαιτεί και ένα πρόγραμμα το οποίο να μπορεί να συντονίσει ένα τέτοιο αριθμό υπολογιστικών μονάδων, ένα από τα διαδεδομένα προγράμματα αποτελεί το Globus Toolkit. Η τελευταία έκδοση του, Globus Toolkit 4 περιγράφεται όσον αφορά τις λειτουργίες του στο **Κεφάλαιο 6**. Ακολουθεί ανάλυση του resource monitoring στο **Κεφάλαιο 7**. Το UML Diagram του κώδικα βρίσκεται στο **Κεφάλαιο 8** μαζί με την ανάλυση του, και ο κώδικας παραθέτεται στο **Κεφάλαιο 9**.

2 *Τεχνολογία Grid*

2.1 Εισαγωγή στα Grids

Στην εποχή μας είναι γεγονός ότι υπάρχει συνεχής αύξηση του ρυθμού μετάδοσης δεδομένων, η χρήση όλο και πιο ισχυροί υπολογιστές , η αποδοχή του Internet έχει οδηγήσει την αγορά για εύρεση νέων και πιο αποδοτικών τρόπων για υπολογισμούς. Αυτό το γεγονός αποτέλεσε κίνητρο για έρευνα με θέμα νέες τεχνολογίες και πρακτικές για βελτίωση των επιχειρήσεων τους[2].

Σήμερα πολλοί είναι οι χρήστες που αναζητούν όλο και περισσότερες υπολογιστικούς πόρους για την επίλυση προβλημάτων που ζητούν πολύπλοκους υπολογισμούς. Παρατηρείται όμως το εξής φαινόμενο ότι η μεμονωμένη χρήση υπολογιστών έχει ως αποτέλεσμα την υπερφόρτωση του συστήματος, την επανεκκίνηση του σε κάποιες περιπτώσεις , αλλά και την μακρόχρονη εκτέλεση υπολογισμών που ενδέχεται να κρατήσουν μέρες ή ακόμα και εβδομάδες μειώνοντας με αυτό τον τρόπο την παραγωγικότητα. Είναι προφανές ότι οι χρήστες οφείλουν να εστιάσουν όχι στη διαθεσιμότητα υπολογιστικών πόρων, αλλά στην αρχιτεκτονική και σχεδίαση λογισμικών συστημάτων τα οποία συντελούν στην βελτίωση της παραγωγικότητας.

Φυσικά δεν υπάρχει επιχείρηση η οποία να μην έχει περιορισμούς σε υπολογιστικούς πόρους. Το κόστος των πόρων αυτών αποτελεί κρίσιμη προτεραιότητα και κάθε οργανισμός θα πρέπει να φροντίζει την βέλτιστη αξιοποίηση των διαθέσιμων πόρων του. Βέβαια εκτός από το κόστος αναβάθμισης hardware το οποίο θα βοηθήσει στην βελτιστοποίηση της απόδοσης του συστήματος, οι διαχειριστές θα πρέπει να λάβουν υπόψη ότι πρέπει υπάρχει ενιαία δομή λογισμικού και hardware ανεξάρτητα από το κάθε σύστημα. Ένα τέτοιο σύστημα θα πρέπει να μπορεί να διαχειρίζεται δυναμικά το φόρτο εργασίας. Αυτό είναι πλέον εφικτό με την τεχνολογία Grid.

2.2 Κατηγοριοποίηση Συστημάτων Grid

Ο όρος Grid αναφέρεται σε ένα σύνολο από υπολογιστικούς πόρους οι οποίοι αναλαμβάνουν διεργασίες. Έτσι ένας χρήστης βλέπει ένα υπολογιστικό πόρο, ενώ υπόκειται ένα πολυδύναμο κατανεμημένο πλέγμα από υπολογιστικούς πόρους. Ο διαχειριστής υπολογιστικών πόρων αναλαμβάνει εργασίες που υποβάλουν οι χρήστες και τις δρομολογεί ανάλογα γοα εκτέλεση στο Grid. Έτσι ο κάθε χρήστης μπορεί να υποβάλει ένα μεγάλο αριθμό εργασιών χωρίς να τον απασχολεί πού θα εκτελεστούν.

Η τεχνολογία Grid αναπτύχθηκε γιατί έγινε πλέον αποδεκτό στην επιχειρηματική κοινότητα η βαρύτητα των κατανεμημένων συστημάτων σε εφαρμογές όπως οικονομική μοντελοποίηση και data mining. Έτσι μέσα σε ένα Grid υπολογιστές, servers, αποθηκευτικά μέσα, βάσεις δεδομένων μπορούν να συνδυαστούν ώστε να δημιουργήσουν μαζική υπολογιστική ισχύ όποτε και όταν χρειαστεί επί το πλείστον. Η αφομοίωση της τεχνολογίας Grid γίνεται με πολύ υψηλούς ρυθμούς. Το έτος 2003 πηγές της Sun Microsystems,Inc αναφέρουν ότι αυξήθηκε η χρήση τους κατά 300%. Η τεχνολογία Grid υπόσχεται να προσφέρει την ισχύ των κατανεμημένων υπολογιστικών συστημάτων στην οθόνη του χρήστη για τα επόμενα χρόνια.

Η τεχνολογία Grid μπορεί να κατανεμηθεί σε 3 μέρη: *Cluster*, *Enterprise* και *Global*.

2.2.1 Cluster Grids

Σήμερα τα Cluster Grids αποτελούν την πιο διαδεδομένη και απλή μορφή Grid. Προσπαθώντας να ανταποκριθούν στις απαιτήσεις των περισσότερων οργανισμών, τα Cluster Grids αποτελούνται από ένα ή περισσότερα συστήματα τα οποία συνεργάζονται για να παράγουν προς το χρήστη ένα σημείο πρόσβασης. Τα Cluster Grids απασχολούν ένα μικρό αριθμό χρηστών οι οποίοι βρίσκονται σε ένα συγκεκριμένο project ή τμήμα μιας επιχείρησης προσφέροντας υποστήριξη σε εργασίες που απαιτούν υψηλές απαιτήσεις και υψηλή υπολογιστική ισχύ. Οι πόροι ενός Grid μπορούν να εστιαστούν είτε σε ένα σύνολο επαναλαμβανόμενων εργασιών, ή σε εργασίες που υποστηρίζουν παράλληλη εκτέλεση. Έτσι ένα Grid μπορεί να αναλάβει τη διεκπεραίωση εργασιών ανεξαρτήτων μεταξύ τους, αλλά και την ολοκλήρωση εργασιών που εκτελούνται σε διαφορετικές υπολογιστικές μονάδες και επικοινωνούν ωστόσο μεταξύ τους για την επίλυση ενός συνόλου προβλημάτων.

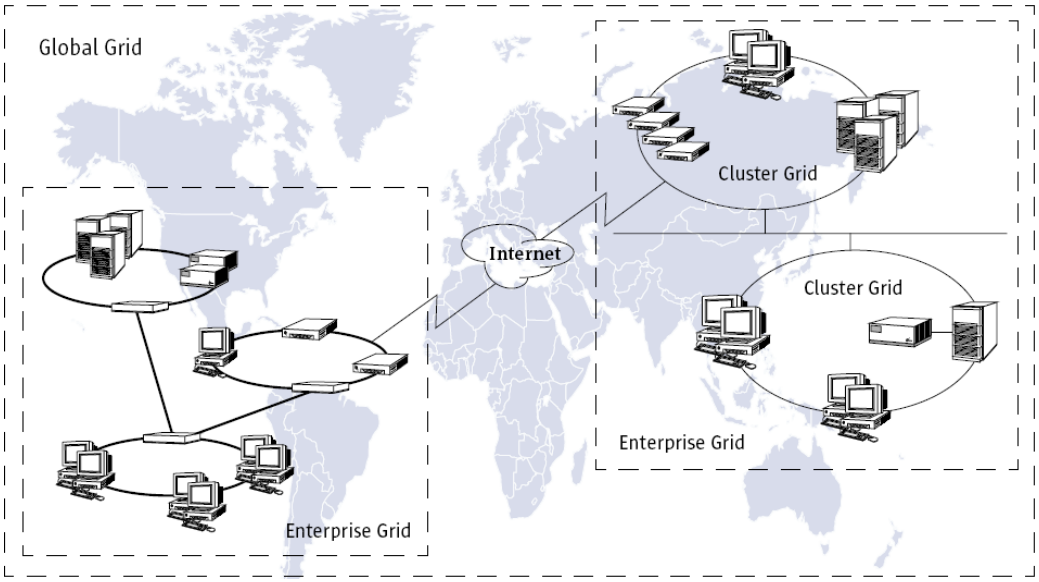
2.2.2 Enterprise Grids

Καθώς οι ανάγκες για εξοικονόμηση χωρητικότητας συνεχώς αυξάνονται , οι οργανισμοί συνδυάζουν τα Cluster Grids σχηματίζοντας έτσι τα Enterprise Grids. Τα Enterprise Grids επιτρέπουν σε πολλαπλά project ή τμήματα μιας υπηρεσίας να μοιράζονται υπολογιστικές πηγές. Τα Enterprise Grids αποτελούνται από ένα εξαπλωμένο σύνολο από servers οι οποίοι μπορεί να είναι σε διάφορα μέρη μιας επιχείρησης ωστόσο αλληλεπιδρούν μεταξύ τους. Μια επιχείρηση

χρησιμοποιεί τα Enterprise Grid για να διαχειριστεί μια μεγάλη ποικιλία εργασιών, συμπεριλαμβανομένου data mining, επεξεργασία frames για animation, απορρόφηση μεγάλου φόρτου εργασίας κατά περιόδους σε μια επιχείρηση και πολλά άλλα.

2.2.3 Global Grids

Όταν ένα Enterprise Grid δεν μπορεί να ανταποκριθεί στις ελάχιστες απαιτήσεις μιας εφαρμογής, μια επιχείρηση μπορεί να αναζητήσει διαθέσιμους υπολογιστικούς πόρους μέσω ενός Global Grid. Σχεδιασμένα για τις ανάγκες εφαρμογών που απαιτούν πολλούς υπολογιστικούς πόρους τα Global Grids παρέχουν την ισχύ των κατανεμημένων πόρων σε χρήστες παντού στον κόσμο. Τα Global Grids είναι ένα σύνολο από Enterprise Grids , τα οποία είναι αφιερωμένα σε για παγκόσμια χρήση. Οι υπολογιστικοί πόροι ενδέχεται να είναι γεωγραφικά διασκορπισμένοι, ωστόσο είναι συνδεδεμένοι μεταξύ τους. Μπορούν να χρησιμοποιηθούν από κάθε χρήστη ανεξάρτητα, από επιχειρήσεις και οργανισμούς, ανταποκρινόμενα σε βαρύ φόρτο εργασίας.



Εικόνα 2-1 Sun Microsystems : The Power Of Grid

2.3 Το Grid στην πράξη

Πιο σημαντικό από οτιδήποτε άλλο είναι, ότι το Grid είναι παράγοντας κλειδί για την γρηγορότερη διεκπεραίωση εργασιών παρέχοντας μεγαλύτερες χωρητικότητες και απόδοση. Προσφέροντας σε επιχειρήσεις έτοιμη πρόσβαση σε ανεκμετάλλευτους πόρους , το Grid προσφέρει νέες ευκαιρίες. Οι οργανισμοί αυτοματοποιημένου ηλεκτρονικού σχεδίου είναι ελεύθεροι να δοκιμάσουν νέα σχέδια σε περισσότερο βάθος και λεπτομέρεια. Οικονομικές επιχειρήσεις μπορούν να διεξάγουν περισσότερες εξομοιώσεις τύπου Monte Carlo ώστε να ανακαλύπτουν ανεκμετάλλευτες επιχειρηματικές ευκαιρίες. Οι κατασκευαστές αυτοκινήτων

μπορούν να διεξάγουν περισσότερες εικονικές δοκιμές ώστε να γίνονται τα οχήματα τους πιο ασφαλή. Επιστήμονες και ερευνητές μπορούν να διεξάγουν περισσότερους και δυσκολότερους υπολογισμούς μέσω ηλεκτρονικού υπολογιστή για την μελέτη και καλύτερη κατανόηση πολύπλοκων φαινομένων. Οι δυνατότητες είναι πάρα πολλές.

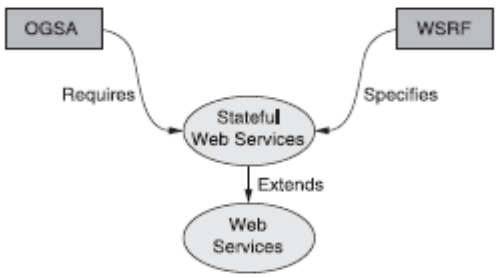
Με την τεχνολογία Grid, οι επιχειρήσεις μπορούν να γίνουν περισσότερο ανταγωνιστικές. Η δύναμη του Grid έγκειται στην ιδιότητα του να παρέχει άφθονους πόρους κάτι το οποίο μεγιστοποιεί την διαθέσιμη ισχύ του τοπικού δικτύου με τους εξής τρόπους:

- Αύξηση της προσωπικής παραγωγικότητας, αυξάνοντας την πρόσβαση στους υπολογιστικούς πόρους.
- Αύξηση της εταιρικής παραγωγικότητας, παρέχοντας περισσότερα προϊόντα ταυτόχρονα.
- Βελτίωση της ποιότητας του προϊόντος, υποστηρίζοντας μια αναπτυξιακή επαναληπτική διεργασία.
- Αύξηση των απολαβών στις επενδύσεις, προωθώντας την βέλτιστη αξιοποίηση των περισσότερων διαθέσιμων, αξιόπιστων δικτυακών και υπολογιστικών πόρων.

3 *OGSA (The Open Grid Services Architecture)*

Ένα κατανεμημένο σύστημα είναι λογικό ότι χρειάζεται να έχει μια αρχιτεκτονική η οποία να επιτρέπει τον συντονισμό και τη διαχείριση του κατά το δυνατόν καλύτερο τρόπο. Αυτό σημαίνει ότι ένα σύστημα Grid χωρίζεται σε μικρότερες και διαφορετικές μονάδες οι οποίες διαθέτουν διαφορετικό ρόλο για την καλύτερη αξιοποίηση χρόνου και πόρων κατά τη διάρκεια μιας ή και πολλών εργασιών[2].Ένα σύστημα Grid αποτελείται συνήθως από διαφορετικές υπολογιστικές μονάδες.

- *Υπηρεσία διαχείρισης VO (Εικονικού Οργανισμού):* Έχει ως ρόλο την διαχείριση των κόμβων και χρηστών που αποτελούν μέλη του Εικονικού Οργανισμού.
- *Υπηρεσία Αναζήτησης και Διαχείρισης Πόρων:* Με αυτό τον τρόπο οι εφαρμογές Grid θα ζητούν πόρους που ανταποκρίνονται στις απαιτήσεις τους και έπειτα θα τους διαχειρίζονται.
- *Υπηρεσία διαχείρισης εργασιών:* Ο κάθε χρήστης μπορεί να υποβάλει εργασίες προς το Grid.
- *Άλλες υπηρεσίες που έχουν να κάνουν με την ασφάλεια, με την διαχείριση δεδομένων κτλ.*



Εικόνα 3-1 Globus Toolkit 4, Σχήμα 2.1(Σχέση μεταξύ OGSA, WSRF και Web Services)

Όλες αυτές οι υπηρεσίες αλληλεπιδρούν συνεχώς. Για παράδειγμα η **υπηρεσία Διαχείρισης Εργασιών** μπορεί να συμβουλευτεί την **υπηρεσία Αναζήτησης Πόρων** για να βρει υπολογιστικούς πόρους που ανταποκρίνονται στις απαιτήσεις τις τρέχουσας εργασίας. Ένα τόσο μεγάλο πλήθος υπηρεσιών δε θα ήταν δύσκολο να προκαλέσει χάος. Ο κάθε πάροχος τέτοιων υπηρεσιών θα μπορούσε να υλοποιήσει δικές του υπηρεσίες με τα ίδια καθήκοντα , αλλά με διαφορετικό τρόπο, παρέχοντας με αυτόν τον τρόπο διαφορετικές λειτουργίες. Κάτι τέτοιο θα αποτελούσε ανασταλτικό παράγοντα στην συνεργασία ενός μεγάλου αριθμού υπηρεσιών με διαφορετικές υλοποιήσεις.

Η λύση σε αυτό το πρόβλημα είναι στο να υπάρχει ένα κοινό πρότυπο για κάθε τύπο υπηρεσίας, μία βάση που θα εξυπηρετούσε στην υλοποίηση υπηρεσιών με κανόνες τους κανόνες της βάσης αυτής. Αναφέροντας ως παράδειγμα τον Παγκόσμιο Ιστό (World Wide Web) γίνεται ξεκάθαρη η πολυτιμότητα των προτύπων για εφαρμογές προς ευρεία χρήση. Ένας από τους λόγους που καθιέρωσε τον Παγκόσμιο Ιστό (World Wide Web) προς τον περισσότερο κόσμο είναι το γεγονός ότι βασίζεται σε πρότυπα (HTML–, HTTP κτλ) τα οποία ακολουθούν όλοι οι κατασκευαστές των Internet Browsers (Microsoft, Mozilla , κτλ) . Έτσι έχοντας ένα Internet Browser ο οποίος ακολουθεί αυτά τα πρότυπα είμαστε σε θέση να έχουμε πρόσβαση στα περισσότερα websites ανεξάρτητα από το τι τεχνολογία χρησιμοποιούν.

Το OGSA, υλοποιήθηκε από το Global Grid Forum ώστε να θέσει ένα κοινό πρότυπο για τις εφαρμογές που βασίζονται στην τεχνολογία Grid. Ο στόχος του OGSA είναι να ορίσει πρακτικά ένα κοινό πρότυπο για όλες τις υπηρεσίες που είναι διαθέσιμες σε ένα Grid σύστημα (Υπηρεσία διαχείρισης εργασιών, Υπηρεσία Αναζήτησης και Διαχείρισης Πόρων, υπηρεσίες ασφαλείας, κτλ) υλοποιώντας ένα σύνολο από interfaces για τις υπηρεσίες αυτές. Το OGSA έχει θέσει τα πρότυπα για τις πιο σημαντικές υπηρεσίες που θα συναντήσει κάποιος σε ένα Grid σύστημα και που η μοντελοποίηση τους θα αποφέρει μεγάλα οφέλη.

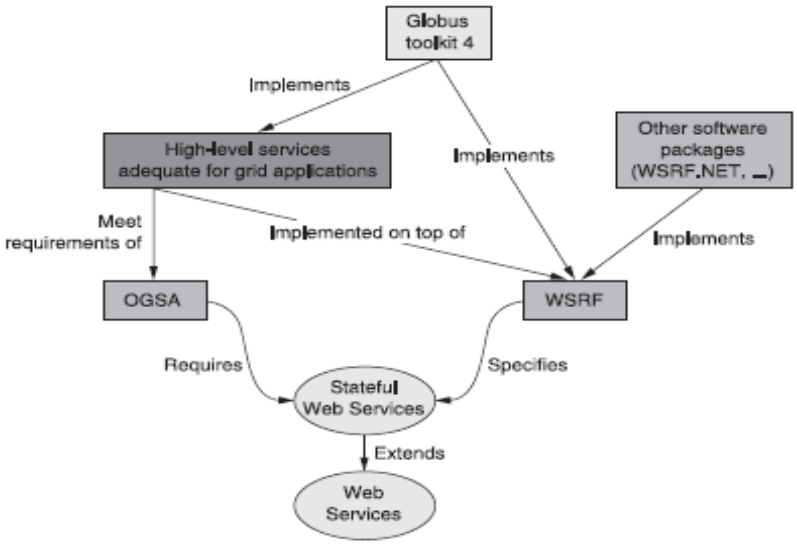
Από τη στιγμή που είχε σχεδιαστεί η αρχιτεκτονική αυτή, έγινε αισθητή η ανάγκη για ένα κατανεμημένο middleware , στο οποίο θα βασιζόταν η αρχιτεκτονική αυτή. Το middleware αυτό θα είχε ως στόχο τον κοινό τρόπο κλήσης λειτουργιών , από τη στιγμή που η αρχιτεκτονική αυτή έπρεπε να έχει ευρεία χρήση από επιχειρήσεις και όχι μόνο. Υπάρχουν πολλά κατανεμημένα middleware τα οποία είναι σε θέση να αποτελέσουν την βάση αυτή της αρχιτεκτονικής (CORBA, RMI, RPC), ωστόσο προτιμήθηκε η λύση των Web Services, για λόγους που διευκρινίζονται παρακάτω στην ανάλυση τους.

Αν και η λύση των Web Services ήταν η ιδανική λύση, δεν υποστήριζαν μια βασική απαίτηση του OGSA το οποίο ήταν ότι το middleware έπρεπε να υποστηρίζει *stateful* ιδιότητες, δηλαδή να είναι σε θέση να συγκρατεί πληροφορία σε κάποιο state. Ωστόσο τα Web Services ενώ

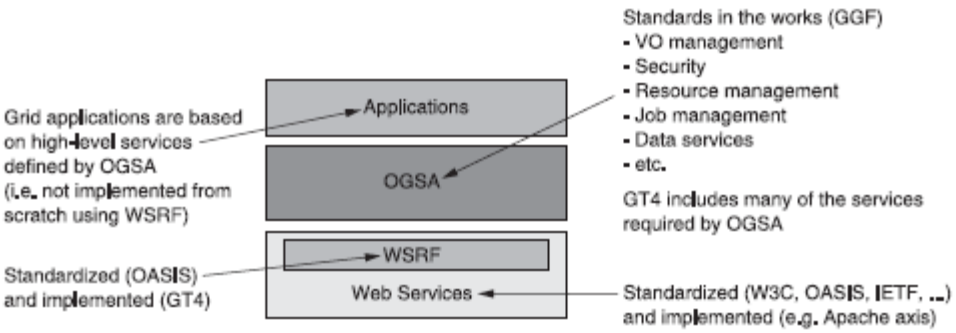
στην θεωρία μπορούν να είναι είτε *stateless* είτε *stateful*, στην πράξη χρησιμοποιούνται ως *stateless*.

Σε αυτό το σημείο τη λύση έρχεται να δώσει το WSRF (The Web Services Resource Framework). Το WSRF καθορίζει πως τα Web Services μπορούν να συμπεριφέρονται ως *stateful* και τους προσδίδουνε τέτοια συμπεριφορά, συμπεριλαμβάνοντας και άλλα επιπρόσθετα χαρακτηριστικά. Έτσι λοιπόν σχηματίζεται μια σχέση μεταξύ OGSA και WSRF, το OGSA είναι η *αρχιτεκτονική* και το WSRF είναι η *υποδομή* πάνω στην οποία έχει κτιστεί η αρχιτεκτονική αυτή.

Εκτός της αρχιτεκτονικής αυτής και της υποδομής της , απαραίτητη κρίνεται η ύπαρξη ενός λογισμικού το οποίο θα είναι αρμόδιο για τη δημιουργία Grid συστημάτων. Ένα τέτοιο λογισμικό είναι το Globus . Το Globus περιλαμβάνει υπηρεσίες υψηλού επιπέδου οι οποίες με κατάλληλη χρήση είναι ικανές για τη βοήθεια στη δημιουργία ενός Grid συστήματος. Αυτές οι υπηρεσίες, ανταποκρίνονται στις απαιτήσεις που θέτει το OGSA. Έτσι κάποιες λειτουργίες του Globus είναι , παρακολούθηση πόρων, υπηρεσία αναζήτησης, υποδομή υποβολής εργασιών, υποδομή ασφαλείας, και υπηρεσία διαχείρισης δεδομένων. Οι περισσότερες από αυτές τις υπηρεσίες υλοποιούνται πλήρως πάνω σε ήδη υπάρχουσες υπηρεσίες του WSRF . Αυτό γίνεται πιο κατανοητό μελετώντας το επόμενο σχήμα.



Εικόνα 3-2 : Globus Toolkit 4, Σχήμα 2.2 (Σχέση μεταξύ OGSA,GT4,WSRF και Web Services)



Εικόνα 3-3 : Globus Toolkit 4, Σχήμα 2.3(Διαστρωμάτωση OGSA, GT4, WSRF και Web Services)

4 *Web Services*

4.1 Εισαγωγή στα Web Services

Οι σύγχρονες τάσεις στις τεχνολογίες πλέγματος δεικτοδοτούν την αξιοποίηση τεχνολογιών από το χώρο των υπηρεσιών διαδικτύου (Web Services) για την υλοποίηση συστημάτων Grid. Η τάση αυτή ενισχύεται ακόμα περισσότερο από τη δυνατότητα που παρέχουν οι τεχνολογίες Υπηρεσιών Διαδικτύου για λειτουργικότητα, ευχρηστία, απόδοση και κλιμάκωση. Καθώς προχωρά ο χρόνος η τεχνολογία των Web Services γίνεται όλο και πιο απαραίτητη και σημαντική στους τομείς του web development και intergration. Τα Web Services επιτρέπουν σε web based εφαρμογές να επικοινωνούν με μηνύματα XML και να σχηματίζουν κατανεμημένα συστήματα ανεξάρτητα το ένα από το άλλο. Αναφερόμενος κάποιος σε Web Services εννοεί αυτόνομα προγράμματα τα οποία είναι διαθέσιμα στο Internet και μπορούν να αλληλεπιδρούν το ένα με το άλλο με μηνύματα μορφής XML . Είναι αυτοπεριγραφόμενα , κάτι το οποίο σημαίνει ότι το public interface ενός Web Service, αποτελούμενο από public μεθόδους και παραμέτρους πρέπει να συνοδεύει το service.

Τα Web Services λοιπόν αποτελούν ένα ιδιαίτερα χρήσιμο μέσο επικοινωνίας για εφαρμογές τύπου client / server [2]. Τα κύρια στοιχεία ενός Web Service είναι το service request το οποίο ζητά μια πληροφορία και το service response το οποίο αποστέλει την πληροφορία για το response που τη ζήτησε. Έστω λοιπόν ότι θέλει κάποιος να γνωρίσει τον καιρό μέσω μιας εφαρμογής που προσφέρει αυτές τις πληροφορίες και το μέσο μετάδοσης είναι τα web services , ένα απλό σχήμα που περιγράφει την ιδέα ενός web service φαίνεται παρακάτω.

Εικόνα 4-1: Globus Toolkit 4, Σχήμα 2.4(Web Services)

Ωστόσο η χρήση Web Services είναι κάτι το οποίο πρέπει να γίνεται λαμβάνοντας υπόψη τα πλεονεκτήματα αλλά και τα μειονεκτήματα τους, τα οποία και παραθέτονται .

4.1.1 Πλεονεκτήματα

Τα Web Services είναι platform-independent και language-independent , λόγω της χρήσης της γλώσσας XML. Αυτό σημαίνει ότι ένα πρόγραμμα client ενδέχεται να έχει υλοποιηθεί σε C++ και να εκτελείται σε περιβάλλον Windows, ενώ το Web Service είναι προγραμματισμένο σε Java και εκτελείται σε περιβάλλον Linux.

Τα περισσότερα Web Services χρησιμοποιούν το πρωτόκολλο HTTP για την μετάδοση μηνυμάτων (όπως τα service request και service response). Αυτό αποτελεί μεγάλο πλεονέκτημα για όσους θέλουνε να αναπτύξουν μια Internet εφαρμογή και θέλει να αποφύγει επιπλοκές με Internet proxies και Firewalls αφού τα ίδια δεν είναι υπεύθυνα για τον έλεγχο μεταφοράς δεδομένων μέσω HTTP.

Ένα ακόμα σημαντικό πλεονέκτημα των Web Services είναι ότι επιτρέπουν λόγω της κατασκευής τους την ανάπτυξη loosely coupled συστημάτων. Αυτοί οι τύποι συστημάτων είναι πολύ πιο επεκτάσιμα από τα strongly coupled συστήματα, έχοντας λιγότερες απαιτήσεις στην αρχιτεκτονική υλοποίησης των Web Services. Το βασικό επιχείρημα που στηρίζει την ιδιότητα των Web Services να είναι ιδανική λύση για την κατασκευή loosely coupled συστημάτων είναι γιατί είναι message-oriented και βασίζονται στην τεχνολογία XML για την αποστολή και λήψη μηνυμάτων.

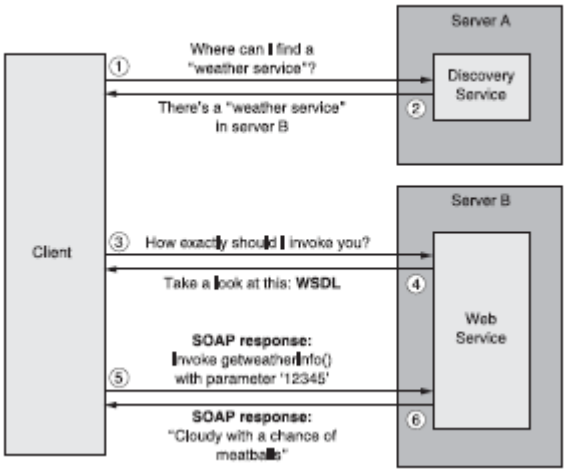
4.1.2 Μειονεκτήματα

- *Δημιουργούν ιδιαίτερο βάρος στην μετάδοση της πληροφορίας.* Η μετάδοση όλων των δεδομένων με τη μορφή XML είναι λιγότερο αποδοτικό από την μετάδοση δεδομένων με δυαδική μορφή. Το κέρδος σε φορητότητα που προσφέρει το XML , χάνεται σε απόδοση. Ακόμα και έτσι, αυτό το overhead είναι τις περισσότερες φορές αποδεκτό από τις περισσότερες εφαρμογές.
- *Είναι ακόμα σε πρώιμο στάδιο.* Τα Web Services είναι σχετικά νέα τεχνολογία, ακόμα και αν χρησιμοποιούν γλώσσες όπως XML, WSDL και πρωτόκολλα όπως HTTP, SOAP

τα οποία είναι ιδιαίτερα σταθερά. Ο ρυθμός εξέλιξής τους παρόλο το ότι βρίσκονται σε πρώιμο στάδιο είναι ιδιαίτερα γρήγορος.

4.2 Ένα τυπικό παράδειγμα κλήσης Web Service

1. Ένας Client ενδέχεται να μην έχει γνώση για το ποιο Web Service θα καλέσει. Έτσι σαν πρώτο βήμα πρέπει να ερευνηθεί αν το Web Service είναι συμβατό με τις απαιτήσεις μας. Θα χρησιμοποιηθεί πάλι το παράδειγμα της αίτησης για πληροφορία για τις καιρικές συνθήκες που επικρατούν σε μια περιοχή. Αυτό επιχειρείται με κλήση ενός διερευνητικού Web Service.



Εικόνα 4-2 : Globus Toolkit 4, Σχήμα 3.2(Τυπικό παράδειγμα κλήσης Web Service)

2. Αυτό το διερευνητικό Web Service θα στείλει μια απάντηση για το αν το Web Service ανταποκρίνεται στις απαιτήσεις του client.

3. Είναι πλέον γνωστή η τοποθεσία του Web Service, αλλά δεν είναι γνωστό προς το παρόν ο τρόπος κλήσης του. Μπορεί είναι γνωστό ότι θα μας δώσει τα προγνωστικά καιρού για την τοποθεσία που θέλουμε, ωστόσο ο τρόπος κλήσης της υπηρεσίας αυτής παραμένει κάτι άγνωστο. Ίσως η μέθοδος να έχει την ονομασία ***String*** *getCityForecast(**int** CityPostalCode)* ή ακόμα και την ονομασία *getUSCityWeather(**String** cityName, bool isFarenheit)*. Για αυτόν τον λόγο πρέπει να γίνει μια αίτηση προς το Web Service για την αυτοπεριγραφή του.

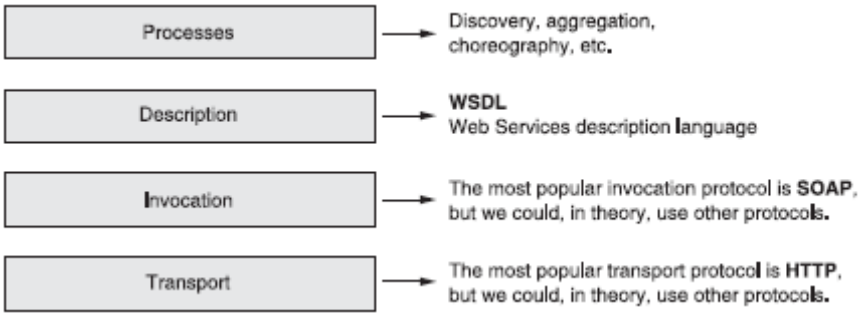
4. Η απάντηση του Web Service είναι σε γλώσσα WSDL.

5. Τελικά γίνεται γνωστό προς τον client η τοποθεσία του Web Service καθώς και ο τρόπος κλήσης του. Η κλήση ενός Web Service είναι διαμορφωμένη σύμφωνα με τη γλώσσα SOAP. Για αυτό το λόγω αποστέλλεται πρώτα μια ερώτηση SOAP η οποία ζητά πληροφορία για τα προγνωστικά καιρού μιας τοποθεσίας.

6. Το Web Service θα αποστείλει μια απάντηση SOAP η οποία θα περιλαμβάνει την πληροφορία που ζητήθηκε, ή ένα μήνυμα λάθους άν η ερώτηση SOAP ήταν λανθασμένη.

4.3 Η αρχιτεκτονική των Web Services

Σε αυτό το κομμάτι θα αναλυθεί η αρχιτεκτονική ενός Web Service, δίνοντας περισσότερο φως στις έννοιες SOAP και WSDL. Τα βασικά στοιχεία ενός Web Service δίνονται στο παρακάτω σχήμα.



Εικόνα 4-3 Globus Toolkit 4, Σχήμα 3.3(Η αρχιτεκτονική των Web Services)

- *Διεργασίες του Web Service (Service Processes)*: Το μέρος αυτό της αρχιτεκτονικής έχει να κάνει με περισσότερα από ένα Web Service. Παραδείγματος χάρη η αναζήτηση ενός Web Service μας επιτρέπει την αναζήτηση ενος συγκεκριμένου Web Service μέσα σε ένα πλήθος από Web Services.
- *Αυτοπεριγραφή του Web Service (Service Description)*: Ένα από τα πιο ενδιαφέροντα προσόντα των Web Services είναι ότι είναι αυτοπεριγραφόμενα. Αυτό σημαίνει ότι, όταν έχει εντοπίσει κάποιος ένα Web Service, είναι σε θέση να ζητήσει την αυτοπεριγραφή του μεταδίδοντας με αυτόν τον τρόπο προς τον ενδιαφερόμενο , τις λειτουργίες που υποστηρίζει και τον τρόπο κλήσης του. Αυτό το ελέγχει η περιγραφική γλώσσα του Web Service (Web Services Description Language -WSDL).
- *Κλήση του Web Service (Service Invocation)*: Η κλήση ενός Web Service (και γενικά κάθε είδους κλήση σε ένα κατανεμημένο service όπως για παράδειγμα τα Enterprise Java Beans) περιλαμβάνει την μετάδοση μηνυμάτων μεταξύ client και server. Το SOAP (Simple Object Access Protocol) δηλώνει τον τρόπου που πρέπει να έχει η μορφή της αίτησης προς τον server καθώς και τον τρόπο που ο server θα πρέπει να μορφοποιεί της απαντήσεις του.
- *Μετάδοση (Transport)*: Όλα αυτά τα μηνύματα πρέπει να μεταδοθούν με ένα τρόπο μεταξύ του client και του server. Το κομμάτι αυτό της αρχιτεκτονικής καλύπτεται με τη

χρήση του πρωτόκολλου HTTP (Hyper Text Transfer Protocol), το ίδιο δηλαδή πρωτόκολλο που χρησιμοποιείται για την πρόσβαση των ιστοσελίδων στο Internet.

4.4 Ένα τυπικό παράδειγμα κλήσης Web Service (Ανάλυση)

Λαμβάνοντας ως δεδομένο ότι έχουμε το Web Service έχει εντοπιστεί και ο client είναι έτοιμος για κλήση του, παραθέτονται τα βήματα τα οποία ακολουθούνται μέχρι να έρθει σε πέρας η διαδικασία αυτή.

1. Όταν η εφαρμογή του client επιθυμεί την κλήση του Web Service, θα μορφοποιήσει κατάλληλα την αίτηση του σύμφωνα με το πρότυπο που επιβάλει το SOAP. Αυτή η διαδικασία ονομάζεται ως διαδικασία marshalling ή serializing.

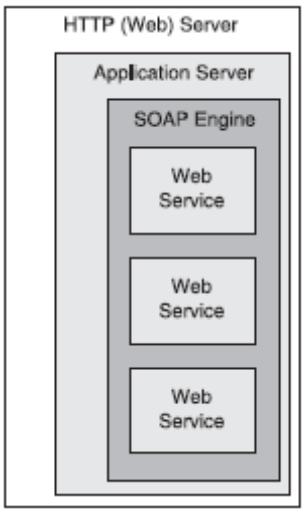
Εικόνα 4-4 : Globus Toolkit 4, Σχήμα 3.5 (Τυπικό παράδειγμα κλήσης Web Service Invocation (πιο λεπτομερές)

2. Η αίτηση SOAP αποστέλλεται μέσω του HTTP. Ο server λαμβάνει την αίτηση SOAP και την τροποποιεί σε κατάλληλη μορφή ώστε να μπορεί να αποδέσμευση την αίτηση της εφαρμογής του client.
3. Όταν επέλθει η αίτηση του client στην κατάλληλη μορφή τότε ο server εκτελεί την αντίστοιχη μέθοδο προς την αίτηση του client.
4. Το αποτέλεσμα της μεθόδου αυτής τροποποιείται κατάλληλα ώστε να είναι συμβατή με μια απάντηση SOAP.
5. Η απάντηση SOAP αποστέλλεται μέσω HTTP. Ο client λαμβάνει την απάντηση από τον server και την τροποποιεί κατάλληλά ώστε να αποδέσμευση την απαραίτητη πληροφορία για την εφαρμογή που κάλεσε το Web Service.
6. Τελικά ο client λαμβάνει την πληροφορία και την χρησιμοποιεί για τους δικούς τους σκοπούς.

4.5 Ο Εξυπηρετητής

Σε αυτό το μέρος γίνεται μια ανάλυση για τη δομή που έχει ένας εξυπηρετητής (server) για Web Service.

- *Web Service:* Το Web Service είναι και το πρωταρχικό στοιχείο. Είναι πλέον γνωστό ότι ένα Web Service είναι μια εφαρμογή λογισμικού που περιέχει ένα σύνολο *λειτουργιών*.



Εικόνα 4-5 Globus Toolkit 4, Σχήμα 3.6(Μορφή μιας εφαρμογής Web Service από το μέρος του Server)

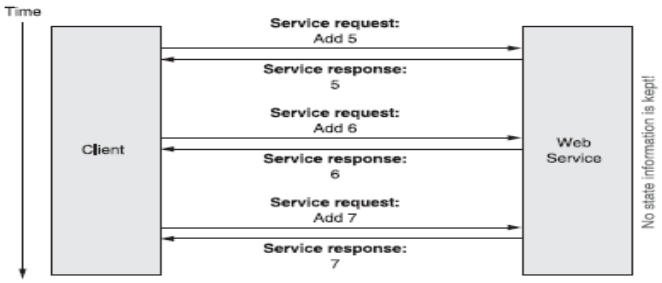
Αν το Web Service ήταν υλοποιημένο με τη γλώσσα προγραμματισμού Java , τότε θα ήταν μια κλάση τύπου Java και οι λειτουργίες του θα ήταν Java μέθοδοι) . Προφανώς, χρειάζεται ένα σύνολο clients ώστε να είναι σε θέση να καλούν όλες αυτές τις λειτουργίες. Ωστόσο δεν είναι αρμοδιότητα του Web Service η μετατροπή των SOAP αιτήσεων, ούτε τη δημιουργία SOAP απαντήσεων. Αυτή είναι αρμοδιότητα του SOAP engine.

- *Soap Engine*: Πρόκειται για εκείνο το κομμάτι λογισμικού που διαχειρίζεται τις αιτήσεις και της απαντήσεις SOAP. Ένα πολύ καλό παράδειγμα Soap Engine είναι το Apache Axis. Ωστόσο η λειτουργία ενός Soap Engine περιορίζεται στην διαχείρηση του SOAP. Για να μπορέσει να λειτουργήσει ως ένας server ο οποίος να λαμβάνει αιτήσεις από διαφορετικούς clients, το Soap Engine πρέπει να λειτουργήσει εντός ενός Application Server.
- *Application Server*: Πρόκειται για το μέρος του server το οποίο επιτρέπει στις εφαρμογές να απασχολούνται από διαφορετικούς χρήστες. Το Soap Engine εκτελείται ως μια εφαρμογή εντός του Application Server . Ένα καλό παράδειγμα είναι ο server Jakarta Tomcat, ένα container για Java Servlets και Java Server Pages του οποίου η χρήση συνδυάζεται μαζί με τον Apache Axis. Οι περισσότεροι application servers διαθέτουν λειτουργίες HTTP, οπότε μπορούμε να έχουμε στη διάθεσή μας εγκαθιστώντας ένα Soap Engine και ένα Application Server. Αν όμως ο Application Server δεν υποστηρίζει λειτουργικότητα HTTP τότε χρειαζόμαστε ένα HTTP Server.
- *HTTP Server*: Ο HTTP Server καλείται πολύ συχνά και ως Web Server. Πρόκειται για ένα server ο οποίος είναι αρμόδιος για τη διαχείριση των HTTP μηνυμάτων. Ένα καλό παράδειγμα είναι ο Apache HTTP Server, ένας από τους πιο διάσημους Web Servers στο Internet.

5 *Web Services Resource Framework (WSRF)*

Γνωρίσαμε προηγουμένως τα Web Services και τη σημαντικότητα που κατέχουν ως επιλογή για Internet εφαρμογές με loosely coupled client και servers. Το γεγονός αυτό καθιερώνει τα Web Services ως την φυσική επιλογή για την επόμενη γενιά των Grid εφαρμογών. Ωστόσο δεν πρέπει να παραληφθεί ότι τα Web Services έρχονται μαζί με κάποιους περιορισμούς. Ουσιαστικά τα απλά Web Services όπως έχουν καθοριστεί από το W3C χαρακτηρίζονται ανεπαρκείς για την κατασκευή μιας Grid εφαρμογής. Αυτός είναι και ο λόγος ύπαρξης του WSRF, η βελτίωση αρκετών παραμέτρων των Web Services καθιστώντας τα ικανά για την κατασκευή και χρήση εφαρμογών Grid[2].

Τα απλά Web Services είναι *stateless* (έστω και αν θεωρητικά, δεν υπάρχει αρχιτεκτονική για Web Services η οποία να διευκρινίζει να είναι stateful). Ο όρος *stateless* σημαίνει ότι το Web Service δεν είναι σε θέση να κρατά σε μνήμη πληροφορίες, από την μια κλήση στην άλλη. Για να γίνει πιο ξεκάθαρη η έννοια του όρου *stateless* ας σκεφτεί κάποιος ένα πολύ απλό Web Service, του οποίου η λειτουργία του είναι αθροιστής ακεραίων. Αυτός ο αθροιστής αρχικοποιείται στο μηδέν και ζητούμενο είναι να αθροίσουμε αριθμούς με αυτόν.



Εικόνα 5-1 Globus Toolkit 4, Σχήμα 4.1 (Κλήση Stateless Web Service)

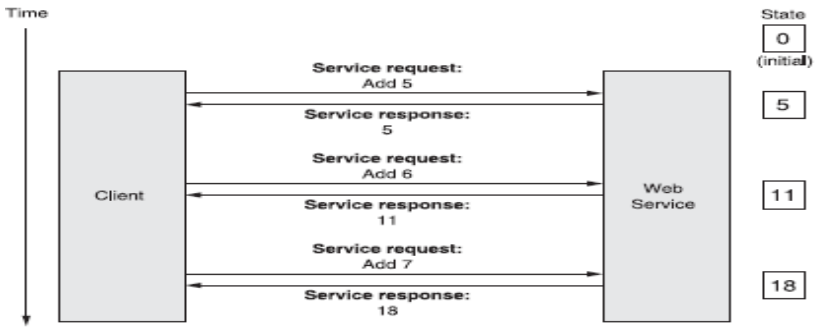
Υποθέτοντας ότι έχουμε την πράξη της πρόσθεσης η οποία λαμβάνει την τιμή που θέλουμε να προσθέσουμε και κατόπιν της πράξης επιστρέφει την τιμή που προκύπτει από τον αθροιστή. Παρατηρώντας το παρακάτω σχήμα κάτι τέτοιο φαίνεται να λειτουργεί στην πρώτη κλήση του Web Service. Όμως λαμβάνοντας υπόψιν ότι το Web Service είναι *stateless* , στις επόμενες κλήσεις δεν είναι σε θέση να γνωρίζουν τα αποτελέσματα προηγουμένων κλήσεων. Για αυτόν τον λόγο στην δεύτερη κλήση που προστίθεται το 6, λαμβάνεται ως απάντηση του Web Service προς τον client το 6, αντί του 11 (το οποίο θα ήταν και το αναμενόμενο αν το Web Service ήταν *stateful*).

Το γεγονός ότι τα Web Services είναι *stateless* δεν είναι απαραίτητα ένα σημαντικό μειονέκτημα. Υπάρχουν πολλές εφαρμογές στις οποίες δεν υπάρχει λόγος για την ύπαρξη ενός *stateful* Web Service. Ένα Web Service για προγνωστικά καιρού, είναι ένα καλό παράδειγμα στο οποίο δεν υπάρχει λόγος ύπαρξης *stateful* Web Service.

Όσον αφορά τις Grid εφαρμογές όμως η χρήση *stateful* Web Services κρίνεται απαραίτητη. Επιθυμητό κρίνεται για τα Web Services, με βάση τον παραπάνω συλλογισμό να συγκρατούν με κάποιο τρόπο πληροφορία. Αυτό έρχεται σε αντιπαράθεση με το γεγονός ότι τα Web Services είναι *stateless*. Προσδίδοντας την ιδιότητα στα Web Services να συγκρατούν πληροφορία σε κάποιο state ενώ παράλληλα πρέπει να είναι *stateless*, αποτελεί ένα περίπλοκο πρόβλημα. Η λύση σε αυτό το πρόβλημα δίνεται αν κρατήσουμε τα Web Services από την πληροφορία που φυλάσσεται σε κάποιο state σε δύο εντελώς ανεξάρτητα επίπεδα.

Έτσι αντί να εμπλουτίσουμε ένα Web Service με το state, κάτι το οποίο μετατρέπει το Web Service σε *stateful*, θα κρατήσουμε την πληροφορία σε ένα αποθηκευτικό πόρο. Κάθε αποθηκευτικός πόρος θα έχει ένα *μοναδικό κλειδί* με το οποίο θα ξεχωρίζει από τους υπόλοιπους, έτσι δίνουμε την ιδιότητα *stateful* στα Web Services τα οποία θα ανατρέξουν για κάποια πληροφορία που πρέπει να αποθηκευτεί όχι σε κάποιο state εντός του Web Service, αλλά σε ένα συγκεκριμένο αποθηκευτικό πόρο εκτός του.

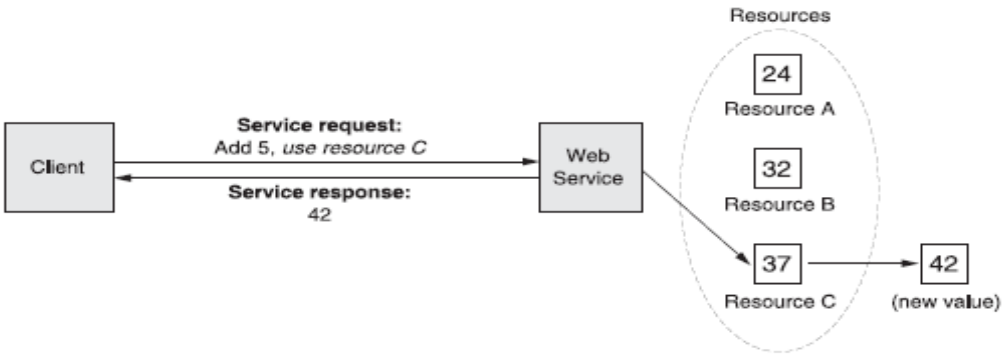
Στο προηγούμενο παράδειγμα του αθροιστή, το Web Service μπορεί να έχει στη διάθεση του τρεις διαφορετικούς αποθηκευτικούς πόρους. Έτσι ανάλογα με το ποιός καλεί το Web Service, θα καλείται ο ανάλογος αποθηκευτικός πόρος που θα ανταποκρίνετα στα ζητούμενα



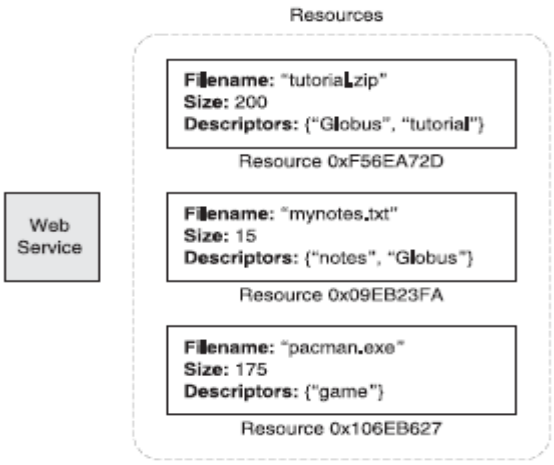
Εικόνα 5-2: Globus Toolkit 4 , Σχήμα 4.2(Κλήση Web Service τύπου statefull)

του client. Στο παρακάτω σχήμα φαίνεται πως ο client ζητά την κλήση της πρόσθεσης συγκρατώντας πληροφορίες σε ένα συγκεκριμένο αποθηκευτικό πόρο τον οποίο ονομάζουμε Α. Όταν το Web Service λάβει μια αίτηση για πρόσθεση, ανατρέχει στον πόρο Α, ώστε να πραγματοποιήσει την πρόσθεση λαμβάνοντας υπόψη την υπάρχουσα πληροφορία που έχει καταχωρηθεί στον Α. Οι αποθηκευτικοί πόροι μπορεί να είναι μνήμη, σκληρός δίσκος, ή ακόμα και βάση δεδομένων. Αξιοσημείωτο είναι ότι ένα Web Service μπορεί να έχει πρόσβαση σε παραπάνω από ένα πόρους.

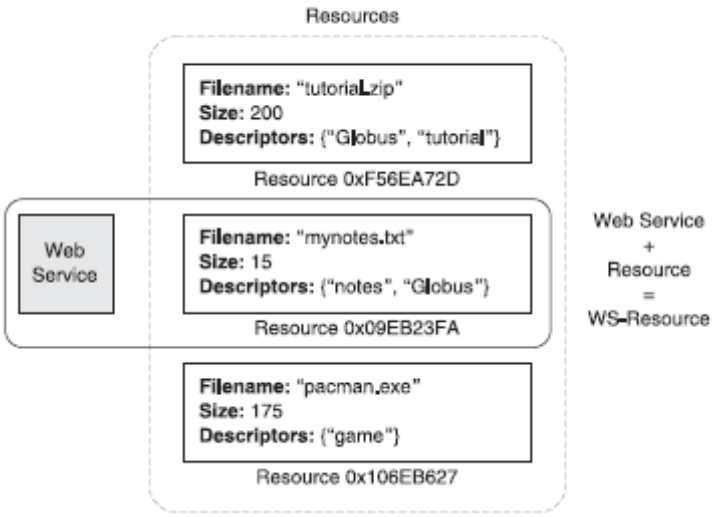
Ερώτημα αποτελεί ο τρόπος που ένας client καθορίζει τον αποθηκευτικό πόρο που πρέπει να χρησιμοποιηθεί. Το ταίριασμα ενός Web Service με ένα αποθηκευτικό πόρο ονομάζεται *WS-Resource* . Άρα αυτό το οποίο είναι το ζητούμενο είναι πως κατευθύνουμε τα Web Services προς τα *WS-Resources*. Ο πλέον ιδανικός τρόπος ονομάζεται *WS-Addressing* ο οποίος προσφέρει ένα πιο ευέλικτο τρόπο διευθυνσιοδότησης Web Services.



Εικόνα 5-3: Globus Toolkit 4 , Σχήμα 4.3(Κατανομή πόρων σε statefull εφαρμογές)



Εικόνα 5-4: Globus Toolkit 4 , Σχήμα 4.4(A Web Service με διάφορες πηγές)

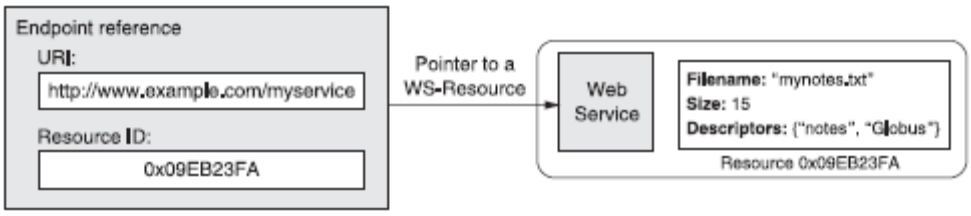


Εικόνα 5-5: Globus Toolkit 4 , Σχήμα 4.5(WS-Resource).

Εικόνα 5-6 Globus Toolkit 4 , Σχήμα 4.6 (Endpoint – reference).

Το WS-Addressing καθορίζει μια δομή με την ονομασία *endpoint – reference* η οποία επιτρέπει να καθορίσουμε μια διεύθυνση προς το Web Service. Το παραπάνω σχήμα δίνει μια καλύτερη εικόνα για την έννοια του *endpoint – reference*.

Το ερώτημα για το πως ένα Web Service αναζητά το κατάλληλο αποθηκευτικό πόρο παραμένει αναπάντητο. Τη λύση δίνει το *endpoint – reference* το οποίο είναι σε θέση να συμπεριλάβει και άλλες πληροφορίες εκτός του URI, όπως και τον τρόπο αναγνώρισης πόρων. Ένα τέτοιο *endpoint – reference* ονομάζεται ως *WS-Resource-qualified endpoint reference* και φαίνεται καλύτερα στο παρακάτω σχήμα.



Εικόνα 5-7: Globus Toolkit 4 , Σχήμα 4.7 (WS-Resource σε σύνδεση με endpoint reference)

Ουσιαστικά ένα endpoint reference (ERP, για συντομία) είναι ένας δείκτης σε ένα *WS-Resource*. Για τους κατασκευαστές ενός Web Service, είναι απαραίτητο να γνωρίζουνε τη δομή ενός ERP. Ωστόσο για τις εφαρμογές του client τα ERP πρέπει να χρησιμοποιούνται ως έχουν, χωρίς καμία επέμβαση στη δομή τους, ούτε χρειάζεται ο χρήστης να γνωρίζει οτιδήποτε για την υλοποίηση των ERP.

Όσον αφορά τους αποθηκευτικούς πόρους, τα δεδομένα μέσα σε αυτούς ονομάζονται resource properties και μας παρέχουν μια όψη της παρούσας κατάστασης του αποθηκευτικού πόρου. Για παράδειγμα τρεις παράμετροι των resource properties είναι το όνομα του αρχείου, το μέγεθος και η περιγραφή του. Τα resource properties χρησιμεύουν για την αποθήκευση των παρακάτω πληροφοριών:

- *Δεδομένα για το Web Service:* Παροχή πληροφοριών για την παρούσα κατάσταση του Web Service, όπως αποτελέσματα από λειτουργίες του service, ενδιάμεσα αποτελέσματα, πληροφορίες κατά την εκτέλεση κτλ. Όνομα του αρχείου, το μέγεθος και η περιγραφή του είναι κάποια από αυτά τα δεδομένα.
- *Πληροφορίες για τα δεδομένα του Web Service (Metadata):* Πληροφορίες σχετικά με τα δεδομένα του Web Service, όπως για παράδειγμα την ημερομηνία αλλαγής ενός αρχείου ή τον χρήστη που το άλλαξε.
- *Πληροφορίες που απαιτούνται για την διαχείριση του state:* Είναι ένας τύπος πληροφορίας παρόμοιος με τα metadata, αλλά αναφέρεται στον αποθηκευτικό πόρο ως σύνολο. Ένα τέτοιο παράδειγμα είναι ο χρόνος λήξης του resource , κάτι που σημαίνει μια παράμετρο που δείχνει πότε ένας πόρος έχει τερματίσει την λειτουργία του.

6 *Globus Toolkit 4*

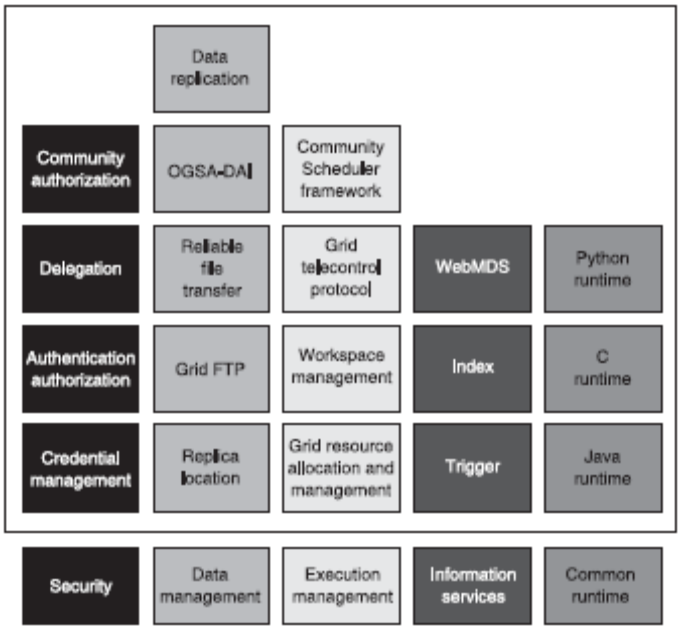
Όπως είδαμε και προηγουμένως το Globus, περιλαμβάνει πολλές υπηρεσίες υψηλού επιπέδου για την κατασκευή Grid συστημάτων. Επίσης περιλαμβάνει μια υλοποίηση του WSRF, η οποία αποτελεί βάση για την ανάπτυξη πολλών άλλων υπηρεσιών του.

Το Globus είναι ένα open source λογισμικό δομημένο ως μια συλλογή από loosely coupled components[1][2]. Αυτά τα components αποτελούνται από υπηρεσίες, βιβλιοθήκες και εργαλεία σχεδιασμένα για την κατασκευή Grid εφαρμογών. Τα components αυτά απευθύνονται στους εξής πέντε τομείς:

1. Ασφάλεια
2. Διαχείριση δεδομένων
3. Διαχείριση εκτέλεσης
4. Υπηρεσία πληροφοριών
5. Common Runtime

6.1 Ασφάλεια

- *Authentication και Authorization:* Περιλαμβάνει βιβλιοθήκες και εργαλεία για τον έλεγχο της πρόσβασης σε υπηρεσίες και σε πόρους, μαζί με ένα framework το οποίο επιτρέπει τη χρήση διαφορετικών μεθόδων authorization, συμπεριλαμβανομένου και μεθόδων υλοποιημένες από τον χρήστη.



Εικόνα 6-1: Globus Toolkit 4 , Σχήμα 5.1 (μέρη του GT4)

- *Delegation:* Το Globus περιλαμβάνει μια υπηρεσία που delegates credentials προς ένα container.
- *Community Authorization:* Εικονικοί οργανισμοί μπορούν να χρησιμοποιήσουν το Community Authorization Service (CAS) για την διαχείριση της πολιτικής authorization που αφορούν τους πόρους των εικονικών οργανισμών.
- *Credential Management:* Αυτό το component περιλαμβάνει το SimpleCA (a Simple Certificate Authority) για χρήστες χωρίς πλήρη πρόσβαση και το MyProxy, ένα online credential repository.

6.2 Διαχείρηση Δεδομένων

Το component για τη διαχείριση δεδομένων προσφέρει στην αναζήτηση , μεταφορά και πρόσβαση σε δεδομένα μεγάλης ποσότητας.

- *GridFTP*: Το component αυτό περιλαμβάνει ένα πλήρως λειτουργικό GridFTP server και πολλές εφαρμογές για τον client. Το πρωτόκολλο GridFTP είναι ειδικά σχεδιασμένο για την μεταφορά πολύ μεγάλων ποσοτήτων δεδομένων.
- *RFT (The Reliable File Transfer)*: Αυτή η υπηρεσία είναι μια υπηρεσία WSRF, η οποία χρησιμοποιεί το GridFTP εσωτερικά για την μετακίνηση μεγάλων ποσοτήτων δεδομένων. Παρέχει και κάποια ενδιαφέροντα χαρακτηριστικά πάνω στο GridFTP , όπως την δυνατότητα για επαναφορά μεταφορών από το σημείο διακοπής.

- *Replica Location*: Η υπηρεσία Replica Location(RLS) επιτρέπει στους χρήστες να μπορούν να ενημερώνονται για την τοποθεσία από διαφορετικές ρεπλίκες ενός dataset μέσα σε ένα εικονικό οργανισμό.
- *Data Replication*: Το Data Replication Service(DRS) χρησιμοποιεί το RLS και το RFT γι α να εγγυηθεί ότο τα τοπικά αντίγραφα από ρεπλίκες είναι διαθέσιμα για τα hosts που τα χρειάζονται.
- *OGSA – DAI*: Το OGSA Data Access and Integration παρέχει ένα framework για την πρόσβαση και την ολοκλήρωση datasets σε ένα Grid τα οποία μπορεί να είναι διαθέσιμα σε διαφορετικά format (απλό κείμενο, βάση δεδομένων, XML).

6.3 Διαχείρηση Εκτέλεσης

Το component για την διαχείριση εκτέλεσης ασχολείται με το deployment, τη δρομολόγηση και την παρακολούθηση εκτελέσιμων προγραμμάτων τα οποία αναφέρονται και ως εργασίες.

- *Grid Resource Allocation & Management (GRAM)*: Το GRAM είναι η καρδιά του component διαχείρησης εκτέλεσης του Globus, παρέχοντας υπηρεσίες για deploy και παρακολούθηση εργασιών σε ένα Grid.
- *Community Scheduler Framework (CSF)*: Το component αυτό παρέχει ένα **int**erface σε διάφορους δρομολογητές πόρων όπως το PBS, Condor, LSF και SGE.
- *Workspace Management:* Ένα νέο component στο Globus το οποίο επιτρέπει στους χρήστες να δημιουργήσουν δυναμικά και να διαχειριστούν workspaces σε απομακρυσμένους hosts.
- *Grid Telecontrol Protocol:* Αυτό το component παρέχει ένα **int**erface μιας WSRF υπηρεσίας για τηλε-έλεγχο(έλεγχος απομακρυσμένων οργάνων).

6.4 Υπηρεσία Πληροφοριών

Οι υπηρεσίες πληροφοριών, που αναφέρονται κοινώς ως Monitoring and Discovery System (MDS), περιλαμβάνουν μια συλλογή από components για παρακολούθηση και αναζήτηση πόρων σε έναν εικονικό οργανισμό.

- *Index Service*: Αυτό το component χρησιμοποιείται για συλλογή πόρων ενδιαφέροντος σε έναν εικονικό οργανισμό.
- *Trigger Service*: Όπως και στο Index Service, το Trigger Service συλλέγει δεδομένα από τους πόρους, αλλά είναι προγραμματισμένο να εκτελεί συγκεκριμένες εργασίες βασισμένες στα δεδομένα.

- *WebMDS*: Παρέχει μια παρουσίαση των δεδομένων εντός ενός web browser που συλλέγοντες από τις αρμόδιες υπηρεσίες.

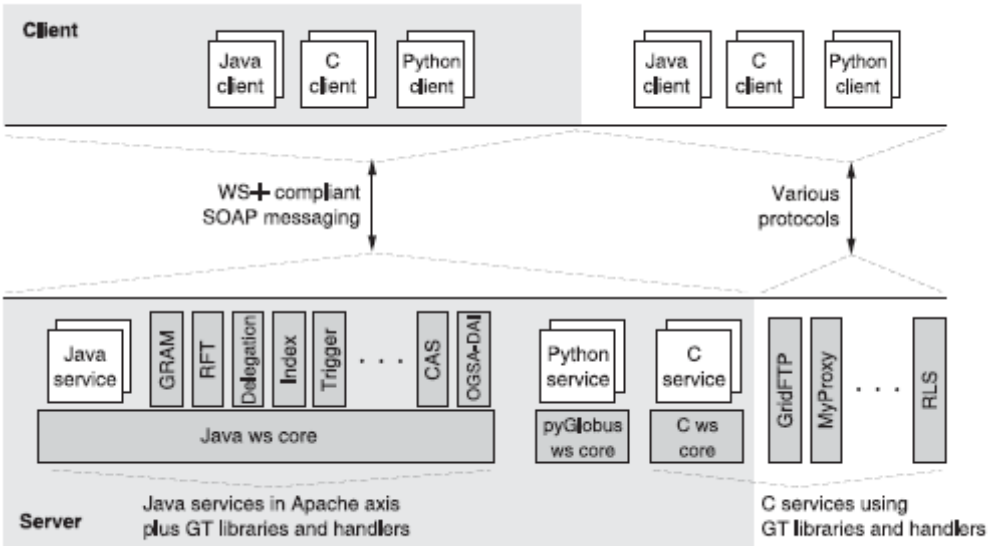
6.5 Common Runtime

Το component Common Runtime παρέχει ένα σύνολο από θεμελιώδεις βιβλιοθήκες και εργαλεία για το hosting υπάρχουσων υπηρεσιών, καθώς και για ανάπτυξη νέων υπηρεσιών.

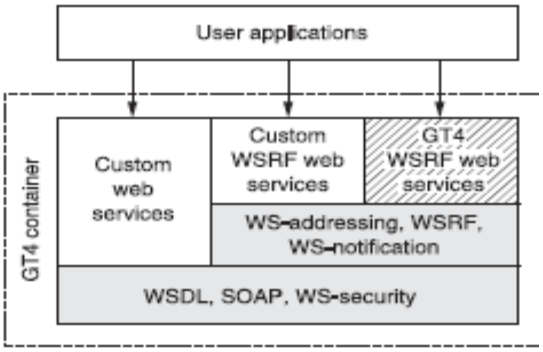
- *C Runtime*: Περιλαμβάνει εργαλεία, βιβλιοθήκες και ένα WS hosting περιβάλλον για C Developers.
- *Python Runtime*: Περιλαμβάνει εργαλεία, βιβλιοθήκες και ένα WS hosting περιβάλλον για Python Developers.
- *Java Runtime*: Περιλαμβάνει εργαλεία, βιβλιοθήκες και ένα WS hosting περιβάλλον για Java Developers.

Όπως έχει αναφερθεί, ένα Web Services Container είναι ένα Container το οποίο περιλαμβάνει SOAP Engine και application server και πιθανώς και ένα HTTP server. Ένα Container παρέχει το περιβάλλον εκτέλεσης για ένα Web Service. Πολλά από τα components του Globus είναι Web Services και πρέπει να εκτελεστούν εντός ενός Web Services Container ώστε να χειριστούν κατάλληλα τις αιτήσεις του client.

Το Globus περιλαμβάνει ένα απλό Web Service Container, βασισμένο στο Apache Axis, το οποίο αναφέρεται συνήθως ως *standalone container*. Όπως φαίνεται και στο παρακάτω σχήμα το container αυτό επιτρέπει την εκτέλεση τόσων των υπηρεσιών που προσφέρονται από το Globus, όσον και των υπηρεσιών που είναι υλοποιημένες από τον χρήστη, βασιζόμενες πάντα στις υλοποιήσεις που προσφέρει το Globus.



Εικόνα 6-2: Globus Toolkit 4 , Σχήμα 5.3 (επισκόπηση GT4 service)



Εικόνα 6-3: Globus Toolkit 4 , Σχήμα 5.2 (Ο Java GT4 container)

7 *Resource monitoring analysis*

Ένα από τα σημαντικότερα κεφάλαια στο κεφάλαιο που ονομάζεται Grids,αποτελεί το resource monitoring. Η επιλογή βέλτιστων resources για την υποβολή εργασιών σε ένα computational Grid ή την πρόσβαση σε δεδομένα από ένα data Grid είναι ένα από τα πιο σημαντικά ζητήματα για κάθε Grid middleware. Τα σύγχρονα λειτουργικά Grid πληρούν αυτή την απαίτηση δίνοντας έτσι τη βέλτιστη απόδοση για την επίλυση ενός προβλήματος. Σχεδόν όλες οι αποφάσεις σχετικά με τη δρομολόγηση των εργασιών αλλά και την πρόσβαση σε δεδομένα γινόταν αυτόματα δίνοντας στον χρήστη ελάχιστες πληροφορίες σχετικά με τα resources που καταναλώνονται ή είναι διαθέσιμα αφήνοντας τον χρήστη χωρίς ουσιαστικό έλεγχο της εργασίας του. Πλέον όμως υπάρχουν services οι οποίες επιτρέπουν στον χρήστη συλλογή πληροφοριών σχετικά με το τι συμβαίνει στο Grid τα resources τα οποία είναι διαθέσιμα καθώς και τις επιδόσεις των εκάστοτε resources , επιτρέποντας έτσι στον χρήστη καλύτερο έλεγχο.

Η αποδοτική διαχείριση των resources αποτελεί ένα από τα πιο σημαντικά χαρακτηριστικά σε κάθε Grid computing middleware[5]. Τα Resources μπορεί να περιλαμβάνουν επιδόσεις του επεξεργαστή, των αποθηκευτικών μέσων , αλλά και του εύρους ζώνης που απαιτούνται από κάθε διεργασία που εκτελείται σε περιβάλλον Grid. Στα περισσότερα Grid Middleware η ολοκλήρωση μιας εργασίας απαιτεί διαχείριση των resources είτε αυτό έχει να κάνει με τη δρομολόγηση της εργασίας , με την επιλογή των κατάλληλων αποθηκευτικών μέσων. Όλα αυτά όμως συμβαίνουν χωρίς ο χρήστης να έχει γνώση. Κάτι τέτοιο κρίνεται επιθυμητό για τους μέσους χρήστες. Για τους Advanced χρήστες όμως που διαθέτουν παραπάνω γνώσεις , κάτι τέτοιο τους εμποδίζει να χρησιμοποιήσουν την βέλτιστη ισχύ που τους παρέχεται από το computing Grid. Οι advanced χρήστες προτιμούν να διαχειρίζονται οι ίδιοι τις εργασίες τους , λαμβάνοντας όσον το δυνατό περισσότερες πληροφορίες για την κατάσταση των Grid resources, παρατηρώντας την

πρόοδο των εργασιών τους, μετακινώντας τες σε άλλες πηγές ή ακόμα έχουν την δυνατότητα για επανεκκίνηση τους σε άλλα resources πιο αποδοτικά για την εργασία τους. Με αυτόν τον τρόπο κατορθώνουν να έχουν περισσότερο έλεγχο στα διαθέσιμα resources επιτυγχάνοντας βέλτιστε επιδόσεις.

Είτε πρόκειται για τον μέσο χρήστη είτε για advanced χρήστη , η καταγραφή και η παρατήρηση των resources κρίνεται ιδιαίτερα σημαντική[4]. Στην περίπτωση του μέσου χρήστη ο οποίος θέλει να υποβάλει μια εργασία, ωστόσο δεν είναι σε θέση να επιλέξει μόνος του και ούτε το επιθυμεί να εισέλθει στη διαδικασία διαχείρισης της εργασίας του, αλλά ενδιαφέρεται για την πρόοδο της αλλά και την ολοκλήρωση της , κρίνεται απαραίτητη η ύπαρξη ενός δρομολογητή εργασιών , αλλά και διαχειριστή πόρων. Κάθε Grid middleware παρέχει αυτή τη δυνατότητα. Όμως χωρίς την παροχή πληροφοριών για τα διαθέσιμα resources την εκάστοτε χρονική στιγμή αυτή η δυνατότητα δεν θα ήταν εφικτή.

Δεν χρειάζεται να αναφέρουμε ότι χωρίς την καταγραφή των διαθέσιμων resources ο όρος δρομολόγηση θα ήταν μη επαρκής , αφού χωρίς γνώση των παραπάνω δε θα μπορούσε να γίνει σωστή δρομολόγηση ανάλογα με τις απαιτήσεις της κάθε διεργασίας[3]. Η τοποθέτηση τους σε μια σειρά δρομολόγησης θα ήταν κάτι μη επαρκές όσον αφορά τις επιδόσεις που κάθε πάροχος υπηρεσιών πρέπει να προσφέρει. Έτσι λοιπόν ανάλογα με τις απαιτήσεις που έχει η κάθε διεργασία τόσο σε υπολογιστικό χρόνο , όσο και σε κατανάλωση μνήμης και bandwidth πρέπει να τοποθετείται την κατάλληλη χρονική στιγμή , εξασφαλίζοντας έτσι στο Grid ικανοποιητική απόδοση τόσο για την ίδια την εργασία , όσο και για τις υπόλοιπες εργασίες που εκτελούνται παράλληλα. Επίσης εκτός από την υπολογιστική δύναμη που κρίνεται σημαντικός παράγοντας για την δρομολόγηση μιας εργασίας , δεν πρέπει να παραλείπεται και η μνήμη που αυτή η εργασία καταναλώνει. Έτσι μπορεί να έχει δρομολογηθεί μια εργασία με βάση την υπολογιστική ισχύ που απαιτεί, όμως αν δεν ληφθεί υπόψη οι απαιτήσεις σε μνήμη RAM αλλά και σε αποθηκευτικό χώρο , είναι πολύ πιθανό να υπάρξουν αστάθειες στο σύστημα. Η κάθε εργασία λοιπόν μπορεί να χρειαστεί να μεταφερθεί σε διαφορετικές υπολογιστικές ή αποθηκευτικές μονάδες, όχι μόνο κατά την έναρξη της , αλλά και κατά την εκτέλεση της. Κάτι το οποίο δεν θα ήταν εφικτό χωρίς το resource monitoring.

Εκτός από τους μέσους χρήστες , η ύπαρξη των advanced χρηστών , δημιουργεί την ανάγκη παροχής περισσότερου ελέγχου όσον αφορά την εκτέλεση της εργασίας που αυτός ο χρήστης έχει υποβάλει. Έτσι λοιπόν ο χρήστης θα είναι σε θέση να παρατηρεί τα διαθέσιμα resources πριν την υποβολή της εργασίας του, κατανέμοντας την ο ίδιος στις υπολογιστικές και αποθηκευτικές μονάδες επιλογής του. Κατά τη διάρκεια της εκτέλεσης της εργασίας του μπορεί να παρατηρεί την πρόοδο της και να αλλάξει υπολογιστικές ή αποθηκευτικές μονάδες ανάλογα με την κρίση του μεγιστοποιώντας με αυτό τον τρόπο την επίδοση για την εργασία του. Όλες αυτές οι δυνατότητες θα ήταν ανέφικτες χωρίς την real time παροχή πληροφοριών για τα διαθέσιμα υπολογιστικά και αποθηκευτικά resources.

8 *Πειραματικό μέρος*

Έχοντας αναφέρει την αρχιτεκτονική, την υποδομή, τα εργαλεία που χρησιμοποιούνται για την κατασκευή ενός Grid συστήματος, καθώς και τις υπηρεσίες που εκείνο παρέχει αλλά και την τεχνολογία που χρησιμοποιεί γίνεται φανερή η σημαντικότητα ενός τέτοιου συστήματος. Συνοπτικά ένα Grid σύστημα έχει σαν σκοπό την διεκπεραίωση δύσκολων και χρονοβόρων εργασιών με την χρήση πολλαπλών υπολογιστικών και αποθηκευτικών πόρων. Ο λόγος της απασχόλησης των υπολογιστικών πόρων είναι προφανής αφού χωρίς την ύπαρξη τους δεν θα ήταν δυνατός ο υπολογισμός των επιμέρους αλλά και του συνολικού προβλήματος της κάθε εργασίας. Ο μεγάλος αριθμός των υπολογιστικών πόρων οφείλεται στη δυσκολία που παρουσιάζουνε οι εργασίες αυτές, ώστε η περάτωση τους να απαιτεί παραπάνω από ένα υπολογιστικό πόρο. Οι αποθηκευτικοί πόροι έχουνε λόγο ύπαρξης όχι μόνο για την απασχόληση που προκαλεί μια εργασία κατά τη διάρκεια εκτέλεσης της, αλλά και στο γεγονός ότι τα Web Services στα Grids είναι stateful. Δηλαδή είναι σε θέση να συγκρατούν πληροφορία κατά τις κλήσεις τους μεταξύ client – server και η πληροφορία αυτή συγκρατείται σε αποθηκευτικό πόρο(δίσκος συστήματος, βάση δεδομένων κτλ) .

Σημαντικό είναι η παρακολούθηση αυτών των πόρων, ώστε να γίνεται γνωστό προς τον πάροχο Grid υπηρεσιών πόσο απασχολείται η κάθε υπολογιστική του μονάδα. Κάποια στοιχεία τα οποία πρέπει να είναι προς παρακολούθηση για κάθε υπολογιστική μονάδα είναι:

- *CPU Usage*: Το ποσοστό χρήσης της CPU που καταναλώνεται κάθε στιγμή.
- *Ram Usage*: Η χρήση της RAM, δηλαδή το ποσό της RAM το οποίο είναι διαθέσιμο και το συνολικό ποσό της.
- *Disk Usage*: Για όλα τα αποθηκευτικά μέσα της υπολογιστικής μονάδας, η διαθέσιμη χωρητικότητα και η συνολική χωρητικότητα.

Έτσι θα έπρεπε να υλοποιηθεί μια βιβλιοθήκη Java η οποία θα παρείχε πληροφορίες για τις παραπάνω παραμέτρους σε περιβάλλον Windows.

8.1 Java και Windows

Η Java έγινε ευρέως γνωστή ως μια επαναστατική γλώσσα προγραμματισμού , η οποία ήταν platform independent, δηλαδή τα προγράμματα τα οποία είχαν υλοποιηθεί σε περιβάλλον Java ήταν εκτελέσιμα σε οποιοδήποτε γνωστό λειτουργικό σύστημα ανεξαρτήτως από τον κατασκευαστή του. Το ίδιο πρόγραμμα Java μπορούσε να εκτελεστεί σε Windows, Solaris, Linux και Mac OS, χωρίς καμία μετατροπή του. Επίσης οι εφαρμογές και οι βιβλιοθήκες οι οποίες υπάρχουν σε Java και αφορούν κατά κύριο λόγο το διαδίκτυο είναι ατελείωτες, καθώς και μια ολόκληρη γενιά application servers έχουν υλοποιηθεί πάνω σε Java.

Παρότι η Java παρέχει απίστευτη ευελιξία στους παραπάνω τομείς, παρουσιάζει αδυναμία όταν πρόκειται για λήψη συγκεκριμένων πληροφοριών που έχουν να κάνουν αποκλειστικά με το λειτουργικό σύστημα. Το γεγονός ότι η Java είναι platform – independent σημαίνει ότι δεν είναι στη δικαιοδοσία της να γνωρίζει συγκεκριμένες λειτουργίες του κάθε λειτουργικού συστήματος. Έτσι γίνεται κατανοητό ότι η πρόσβαση σε τέτοιου είδους πληροφορίες που απαιτεί το resource monitoring δεν είναι δυνατό να γίνει απευθείας μέσω Java.

Η C ωστόσο που είναι και η γλώσσα πάνω στην οποία έχουν κατασκευαστεί τα *Windows* μπορεί να παρέχει αυτή την πληροφορία. Άρα η λύση θα ήταν αν μπορούσε να γίνει ένας τρόπος κλήσης της κατάλληλης μεθόδου για αποδέσμευση αυτής της πληροφορίας μέσω C και έπειτα μετάδοση αυτής της πληροφορίας σε γλώσσα Java και τελικά αποδέσμευση της.

8.2 Win32 SWT

Ο παραδοσιακός τρόπος επικοινωνίας Java με native μεθόδους της C είναι το JNI, ωστόσο στην προκειμένη περίπτωση χρησιμοποιήθηκε ένας πιο κομψός τρόπος με τον οποίο ο ενδιαφερόμενος προγραμματιστής απασχολείτο μόνο με την χρήση ήδη υπαρχόντων μεθόδων μέσω μιας βιβλιοθήκης και τροποποίηση τους κατ' επίκληση. Η τεχνολογία που χρησιμοποιήθηκε στην προκειμένη περίπτωση ονομάζεται ***Win32 SWT(Standard Widget Toolkit)***.

Το ***Win32 SWT*** είναι μια βιβλιοθήκη η οποία είναι σε θέση να παρέχει ένα σύνολο από native μεθόδους, μεθόδους δηλαδή που αποτελούν μέρη του λειτουργικού συστήματος και μπορούν να παρέχουν πληροφορίες για αυτό. Με αυτό τον τρόπο μπορεί κάποιος να δημιουργήσει μια εφαρμογή με look and feel ακριβώς ίδια με το look and feel του λειτουργικού συστήματος, να καλέσει προς εκτέλεση εξωτερικά προγράμματα τα οποία καλούνται με συγκεκριμένο τρόπο από το λειτουργικό σύστημα ακόμα και να λάβει ή και να τροποποιήσει πληροφορίες που έχουν να κάνουν αποκλειστικά με το έκαστο λειτουργικό σύστημα. Σε αυτή την περίπτωση χρησιμοποιήθηκε για αποδέσμευση πληροφοριών όπως CPU Usage, Disk Usage, Ram Usage.

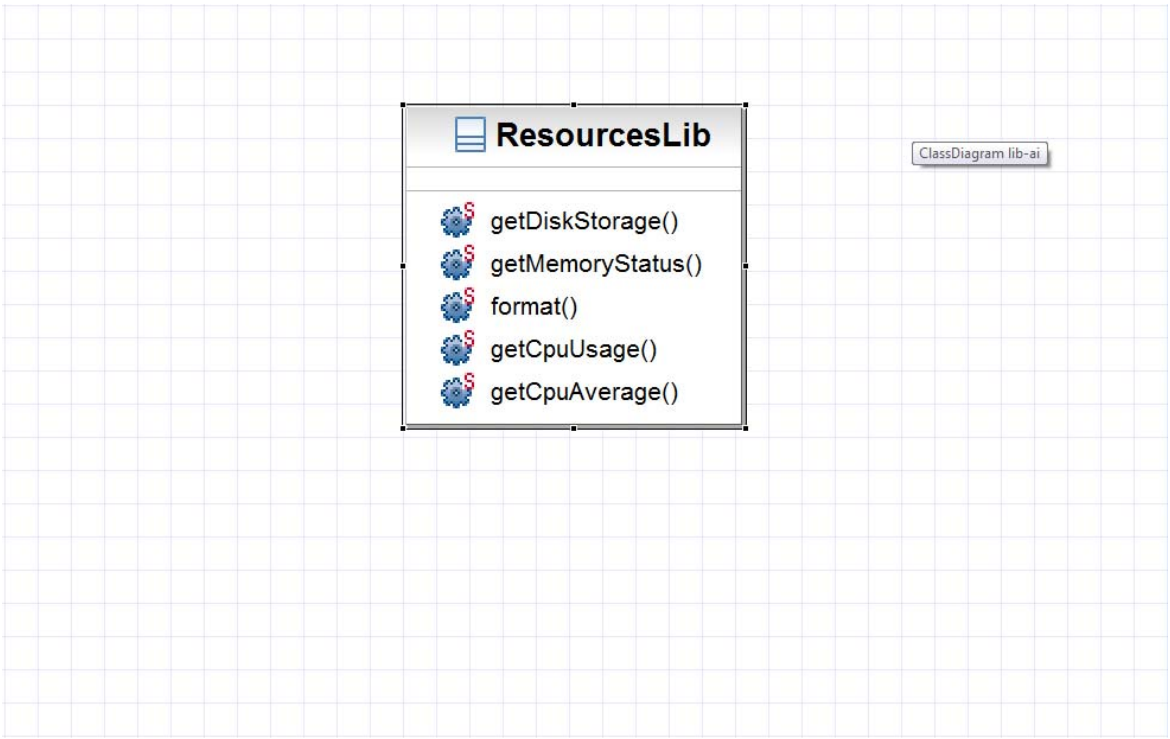
8.3 Εγκατάσταση του προγράμματος

Η εγκατάσταση του προγράμματος για την αποδέσμευση των πληροφοριών για τους πόρους που καταναλώνει μια υπολογιστική μονάδα στους τομείς CPU , Disk Usage, Ram Usage κρίνεται απλή και εύχρηστη. Πριν προχωρήσουμε στο πως εκτελείται το πρόγραμμα αυτό ας αναλύσουμε τα μέρη του προγράμματος για την εγκατάσταση του. Υπάρχει το resources.jar το οποίο περιέχει τον κώδικα αλλά και το εκτελέσιμο αρχείο. Έτσι λοιπόν μπορεί να χρησιμοποιηθεί όχι μόνο ως βιβλιοθήκη για παροχή των πληροφοριών πόρων σε ένα ήδη υπάρχον project , αλλά και ως standalone εκτελέσιμο πρόγραμμα.

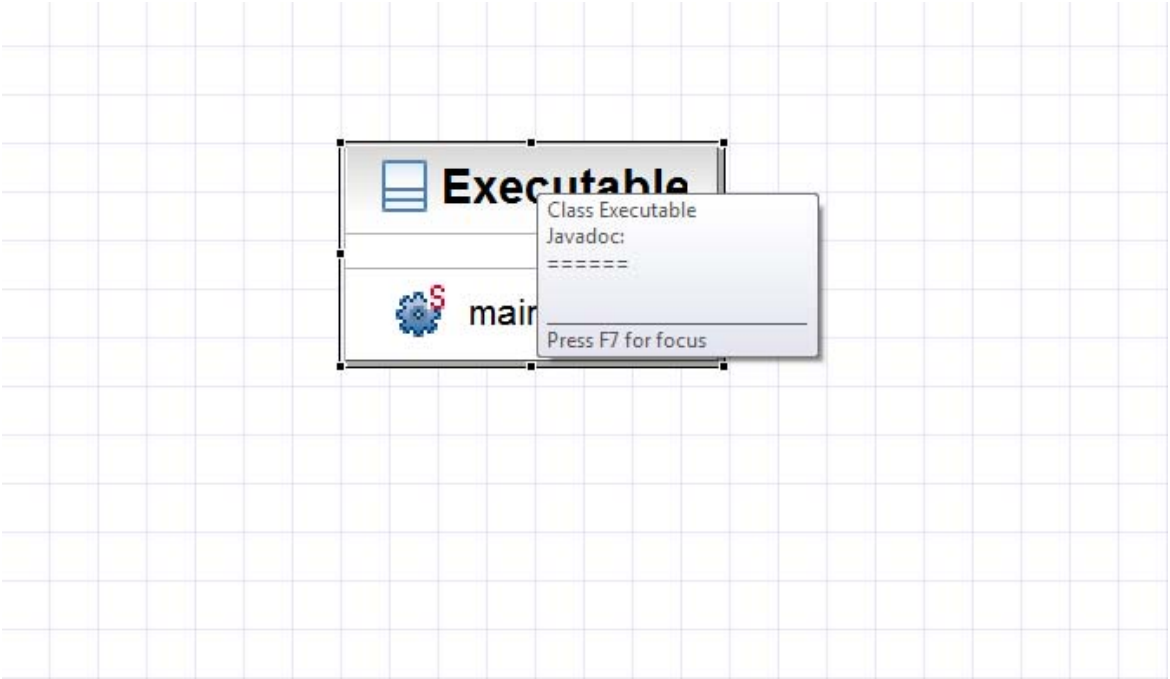
Εκτός από το Resources.jar τα αρχεία ***swt-win32.dll*** και ***swt-extension-win32.dll*** είναι απαραίτητα για την εκτέλεση του. Αυτά τα αρχεία περιέχουν τις native μεθόδους τις οποίες χρησιμοποιεί το ***win32 swt jar***. Το jar αυτό γίνεται import στο project και αποτελεί τον ενδιάμεσο μεταξύ των win32 swt dlls και του κώδικα μας, τα οποία και περιέχουν τις native μεθόδους του υπολογιστικού συστήματος. Τα dlls αυτά πρέπει να βρίσκονται μέσα στο java.library.path ώστε το ***win32 swt jar*** να είναι σε θέση να καλέσει τις αντίστοιχες μεθόδους. Αρκεί τα dll να εγκατασταθούν στο C:\Windows ώστε να μπορούν να χρησιμοποιηθούν κατάλληλα.

8.4 UML Diagram

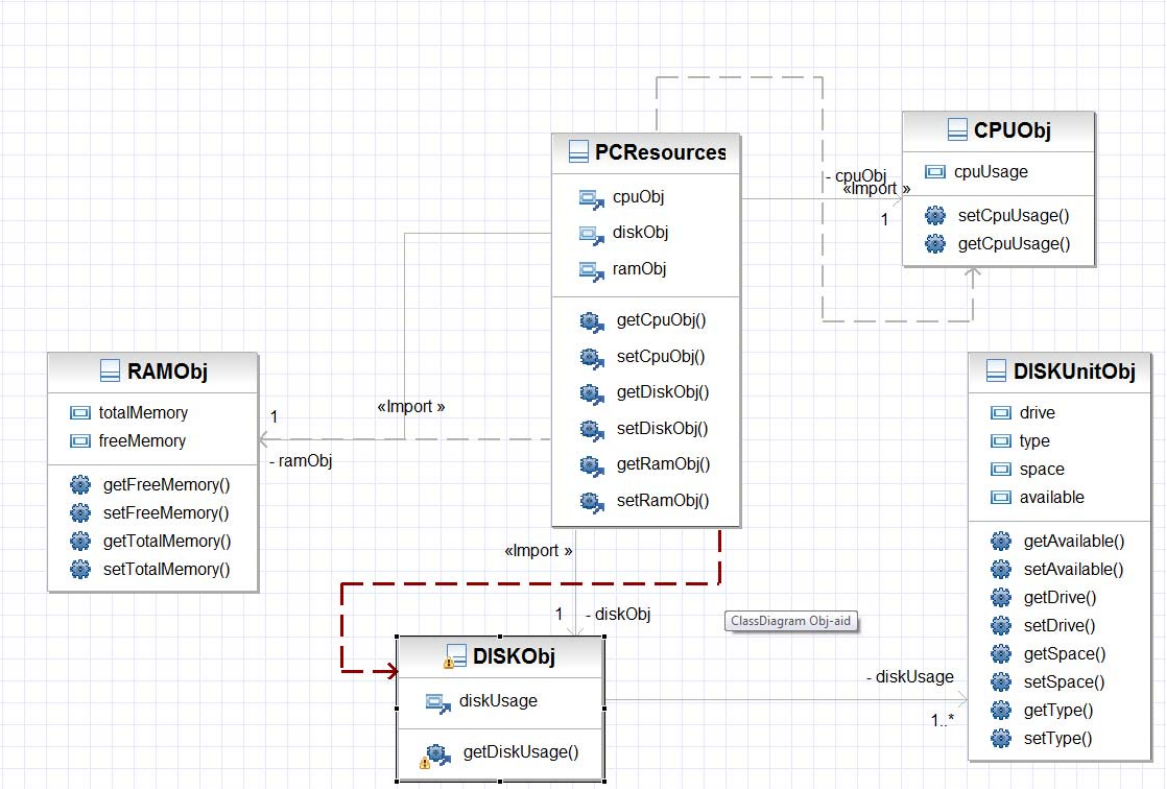
Για την καλύτερη κατανόηση του project αλλά και του κώδικα που παρατίθεται στη συνέχεια , κρίνεται απαραίτητη η υλοποίηση ενός διαγράμματος κλάσεων του προγράμματος μας. Ο λόγος ύπαρξης 3 διαγραμμάτων οφείλεται στο γεγονός ότι το πρόγραμμα χωρίζεται σε τρία μέρη. Στο εκτελέσιμο μέρος που απαρτίζεται αποκλειστικά από την κλάση Executable, το μέρος το οποίο αποτελεί την χρήση του προγράμματος ως εξωτερική βιβλιοθήκη για την λήψη των επιθυμητών πληροφοριών για τους πόρους. Η χρήση του ως εξωτερική βιβλιοθήκη οφείλεται αποκλειστικά στην κλάση ResourcesLib η οποία αποτελείται από static μεθόδους η λειτουργία των οποίων έγκειται στην λήψη των πληροφοριών συστήματος όσον αφορά τους πόρους που αξιοποιούνται. Το τρίτο μέρος του προγράμματος αποτελείται από τα objects που συγκρατούν την πληροφορία για την αξιοποίηση των πόρων και μετέπειτα την παραδίδουν μέσω την κλήση κατάλληλων μεθόδων από την κλάση ResourcesLib στο χρήστη.



UML διάγραμμα 1 : Η κλάση ResourcesLib υπεύθυνη για κλήσεις συναρτήσεων για την αξιοποίηση των πόρων



UML διάγραμμα 2: Η κλάση Executable επιτρέπει την αυτόνομη εκτέλεση του προγράμματος.



UML διάγραμμα 3 : Τα objects που συγκρατούν την πληροφορία για τους πόρους

8.5 Δομή και ανάλυση προγράμματος

Το πρόγραμμα για την καταγραφή των πόρων μιας υπολογιστικής μονάδας ενός πλέγματος Grid, είναι αναπτυγμένο σε γλώσσα Java και αποτελείται από τα ακόλουθα classes: ResourcesLib , RAMObj , DISKObj , DISKUnitObj, **CPUObj**, PCResources, Executable. Οι κλάσεις RAMObj , **CPUObj**, DISKUnitObj αποτελούν Java Beans συγκρατούν την πληροφορία που αρμόζει στο καθένα σε αντιστοιχία με το όνομα του.

Το RAMObj συγκρατεί πληροφορία σχετικά με την ελεύθερη αλλά και την συνολική RAM ενός υπολογιστή, το **CPUObj** την πληροφορία για το ποσοστό χρήση της CPU ανά χρονική στιγμή. Το DISKUnitObj συγκρατεί πληροφορία για κάθε drive που είναι διαθέσιμο σε κάθε υπολογιστή και αποτελείται όχι μόνο από το συνολικό και διαθέσιμο χώρο σε bytes ενός drive, αλλά και το όνομα του drive όπως επίσης και τον τύπο, αν είναι σκληρός δίσκος, cdrom, usb κτλ. Το DISKObj αποτελεί ένα Java Bean το οποίο είναι ουσιαστικά μια λίστα από DISKUnitObj μιας και ένα υπολογιστικό σύστημα έχει παραπάνω από ένα drives συνδεδεμένα και ενεργά. Τέλος το PCResources έχει ένα object που συγκεντρώνει συνολικά την πληροφορία για τους πόρους μιας υπολογιστικής μονάδας, έχοντας ως properties του, instances των **CPUObj**, RAMObj, DISKObj.

Όλα αυτά τα Objects έχουν σαν μοναδικό ρόλο τη συγκράτηση πληροφοριών για τους υπολογιστικούς και αποθηκευτικούς πόρους. Την πληροφορία αυτή την λαμβάνουνε μέσω της

κλάσης ResourcesLib. Το ResourcesLib αποτελείται από **static** μεθόδους και έχει ως σκοπό την εκτέλεση native μεθόδων μέσω του win32 swt jar οι οποίες επιστρέφουν ένα γεμάτο πληροφορία object τύπου **CPUObj** , RAMObj , DISKObj ανάλογα με την μέθοδο που καλείται. Έτσι λοιπόν αυτό που αρκεί για να γεμίσει για παράδειγμα ένα από τα παραπάνω java beans είναι μια κλήση του τύπου : **CPUObj** cpu = ResourcesLib. getCpuUsage() .

9 *Επίλογος*

Σήμερα πολλοί είναι οι χρήστες που αναζητούν όλο και περισσότερες υπολογιστικούς πόρους για την επίλυση προβλημάτων που ζητούν πολύπλοκους υπολογισμούς. Παρατηρείται όμως το εξής φαινόμενο ότι η μεμονωμένη χρήση υπολογιστών έχει ως αποτέλεσμα την υπερφόρτωση του συστήματος, την επανεκκίνηση του σε κάποιες περιπτώσεις , αλλά και την μακρόχρονη εκτέλεση υπολογισμών που ενδέχεται να κρατήσουν μέρες ή ακόμα και εβδομάδες μειώνοντας με αυτό τον τρόπο την παραγωγικότητα. Η αναζήτηση για μια λύση στο παραπάνω πρόβλημα απαίτηση οδήγησε στην δημιουργία κατανεμημένων υπολογιστικών συστημάτων με την ονομασία Grid . Ενός κατανεμημένου συστήματος αποτελούμενο από υπολογιστικές και αποθηκευτικές μονάδες οι οποίες με τα κατάλληλα εργαλεία λογισμικού, αλλά και αρχιτεκτονικής συνεργάζονται αρμονικά για την περάτωση δύσκολων και χρονοβόρων εργασιών που μια και μόνο υπολογιστική μονάδα θα ήταν ανέφικτο να αποδώσει. Στην διπλωματική αυτή έγινε εφικτή η καταμέτρηση resources ανά υπολογιστική μονάδα σε μια γλώσσα προγραμματισμού η οποία εκ φύσεως δεν υποστηρίζει τέτοιους σκοπούς , για τον λόγο ότι δεν αφιερώνεται σε κάποιο λογισμικό σύστημα, αλλά είναι ανεξάρτητη αυτών. Η τεχνολογία των Grids έχει ήδη αρχίσει την εφαρμογή της , στον ευρωπαϊκό οργανισμό πυρηνικών ερευνών (CERN) και η χρήση τους αναμένεται να είναι απαραίτητη σε όλο και περισσότερες εφαρμογές με υψηλές απαιτήσεις διευκολύνοντας με αυτόν τον τρόπο την πρόοδο της επιστήμης και όχι μόνο.

10 *Βιβλιογραφία*

[1]	Globus home page: http://www.globus.org
[2]	Borja Sotomayor, Lisa Childers, "**Globus Toolkit 4, Programming Web Services**", Morgan Kauffman publications, 2006
[3]	Arshad Ali, Ashiq Anjum, Tahir Azim, Julian Bunn, Atif Mehmood, Richard McClatchey, Harvey Newman, Waqas ur Rehman, Conrad Steenberg, Michael Thomas, Frank van Lingen, Ian Willers, Muhammad Adeel Zafar , **"Resource Management Services for a Grid Analysis Environment"**
[4]	Do-Hyeon Kim and Kyung-Woo Kang ,"**Design and Implementation of Integrated Information System for Monitoring Resources in Grid Computing**" , Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design
[5]	"Performance evaluation of a grid resource monitoring and discovery service", H.N. Lim Choi Keung, J.R.D. Dyson, S.A. Jarvis and G.R. Nudd

11 Παράρτημα

11.1 Κώδικας Προγράμματος

11.1.1 Η κλάση ResourcesLib

Το ResourcesLib αποτελείται από **static** μεθόδους και έχει ως σκοπό την εκτέλεση native μεθόδων μέσω του win32 swt jar οι οποίες επιστρέφουν ένα γεμάτο πληροφορία object τύπου **CPUObj** , RAMObj , DISKObj ανάλογα με την μέθοδο που καλείται. Έτσι λοιπόν αυτό που αρκεί για να γεμίσει για παράδειγμα ένα από τα παραπάνω java beans είναι μια κλήση του τύπου : **CPUObj** cpu = ResourcesLib. getCpuUsage().

```
package com.resources.lib;

/**
 * @author Nikos Makris
 * Class το το οποίο αποτελεί βιβλιοθήκη συναρτήσεων
 * σχετικά με τα resources συστήματος, χρησιμοποιεί
 * native μεθόδους μέσω του win32 swt.
 */

import java.text.NumberFormat;


import org.eclipse.swt.extension.Win32;
import org.eclipse.swt.internal.extension.DISKFREESPACE;
import org.eclipse.swt.internal.extension.Extension;
import org.eclipse.swt.internal.extension.MEMORYSTATUS;
import org.eclipse.swt.internal.extension.SYSTEMINFO;


import com.resources.Obj.CPUObj;
import com.resources.Obj.DISKObj;
```

```
import com.resources.Obj.DISKUnitObj;
import com.resources.Obj.RAMObj;

public class ResourcesLib {

	/**
	 * Μέθοδος η οποία παρέχει πληροφορία για την χρήση του αποθηκευτικού χώρου
	 * για όλα τα drives του υπολογιστή.
	 *
	 * @return DISKObj( περιέχει την πληροφορία για disk storage)
	 */

	@SuppressWarnings("unchecked")
	public static DISKObj getDiskStorage() {
		DISKObj diskObj = new DISKObj();
		diskObj.getDiskUsage();

		String[] drives = Extension.GetLogicalDrives();
		for (int i = 0; i < drives.length; i++) {
			DISKUnitObj diskUnitObj = new DISKUnitObj();
			diskUnitObj.setDrive(drives[i]);

			int diskType = Extension.GetDriveType(drives[i]);
			switch (diskType) {
			case Win32.DRIVE_TYPE_UNKNOW:
				diskUnitObj.setType("UNKNOWN");
				break;
			case Win32.DRIVE_TYPE_REMOVABLE:
				diskUnitObj.setType("REMOVABLE");
				break;
			case Win32.DRIVE_TYPE_REMOTE:
				diskUnitObj.setType("REMOTE Disk");
				break;
			case Win32.DRIVE_TYPE_RAMDISK:
				diskUnitObj.setType("RAM DISK");
				break;
			case Win32.DRIVE_TYPE_NOT_EXIST:
				diskUnitObj.setType("NOT EXIST");
```

```
                        break;
                    case Win32.DRIVE_TYPE_FIXED:
                        diskUnitObj.setType("Local Disk");
                        break;
                    case Win32.DRIVE_TYPE_CDROM:
                        diskUnitObj.setType("CD Drive");
                        break;
                    }
                    // Κλήση native μεθόδου
                    DISKFREESPACE space =
Extension.GetDiskFreeSpace(drives[i]);
                    String totleSpace = format(space.totalNumberOfBytes);

                    diskUnitObj.setSpace(totleSpace);

                    if (totleSpace.length() < 8)

        diskUnitObj.setAvailable(format(space.totalNumberOfFreeBytes)
                                + "\n");
                    else

        diskUnitObj.setAvailable(format(space.totalNumberOfFreeBytes)
                                + "\n");
                    diskObj.getDiskUsage().add(diskUnitObj);
                }
                return diskObj;
        }

        /**
         * Μέθοδος η οποία παρέχει πληροφορία για την χρήση της RAM του
υπολογιστή.
         *
         * @return RAMObj (περιέχει την πληροφορία για RAM USAGE)
         */
        public static RAMObj getMemoryStatus() {
                RAMObj ram = new RAMObj();

                // Κλήση native μεθόδου
                MEMORYSTATUS memoryStatus = Extension.GlobalMemoryStatus();
```

```
            ram.setTotalMemory(memoryStatus.dwTotalPhys / (1024) + "
KB");

            ram.setFreeMemory(memoryStatus.dwAvailPhys / (1024) + "
KB");
            return ram;

      }

      /**
       * Μέθοδος για τροποποίηση του αριθμού των bytes σε επιθυμητή
μορφή των
       * KiloBytes.
       *
       * @param totalNumberOfFreeBytes
       * @return
       */
      private static String format(float totalNumberOfFreeBytes) {
            NumberFormat format = NumberFormat.getInstance();
            format.setMaximumFractionDigits(1);
            format.setMinimumFractionDigits(0);
            if (totalNumberOfFreeBytes < 1024)
                  return format.format(totalNumberOfFreeBytes) + "
byte";
            totalNumberOfFreeBytes = totalNumberOfFreeBytes / 1024;
            if (totalNumberOfFreeBytes < 1024)
                  return format.format(totalNumberOfFreeBytes) + " KB";
            totalNumberOfFreeBytes = totalNumberOfFreeBytes / 1024;
            if (totalNumberOfFreeBytes < 1024)
                  return format.format(totalNumberOfFreeBytes) + " MB";
            totalNumberOfFreeBytes = totalNumberOfFreeBytes / 1024;
            if (totalNumberOfFreeBytes < 1024)
                  return format.format(totalNumberOfFreeBytes) + " GB";
            return null;
      }

      /**
       * Μέθοδος για λήψη CPU Usage. Η λήψη γίνεται με δειγματοληψία,
δηλαδή
       * λαμβάνεται ένα σύνολο στιγμών της CPU Usage και προκύπτει ένας
μέσος όρος
       * CPUUsage για αντικειμενικότερη πληροφόρηση.
       *
```

```
	 * @return CPUObj( περιέχει την πληροφορία σχετικά με το CPU
Usage)
	 */
	public static CPUObj getCpuUsage() {
		CPUObj cpuObj = new CPUObj();
		int cpuMoments[] = new int[10];
		try {
			for (int i = 0; i < 10; i++) {

				cpuMoments[i] = Extension.GetCpuUsages();
				Thread.sleep(500);
			}

		} catch (Exception e) {
			// TODO: handle exception
			e.printStackTrace();
			return null;
		}
		CPUObj.setCpuUsage(getCpuAverage(cpuMoments));
		return CPUObj;
	}

	/**
	 * Μέσος όρος CPU Usage
	 *
	 * @param a
	 * @return
	 */
	private static double getCpuAverage(int a[]) {
		int amount = 0;
		for (int i = 0; i < a.length; i++) {
			amount += a[i];
		}
		return amount / (a.length - 1);
	}

}
```

11.1.2 Η κλάση CPUObj

```
package com.resources.Obj;

/**
 * @author Nikos Makris
 * Class που περιέχει ποσοστό χρήσης CPU.
 * Η πληροφορία είναι αντιπροσωπευτική για 1 ή και
 * περισσότερους πυρήνες
 */
public class CPUObj {

	private double cpuUsage;

	public void setCpuUsage(double cpuUsage) {
		this.cpuUsage = cpuUsage;
	}

	public double getCpuUsage() {
		return cpuUsage;
	}
}
```

11.1.3 Η κλάση DISKObj

```
package com.resources.Obj;

import java.util.ArrayList;
import java.util.List;

/**
 * @author Nikos Makris
 * Class που περιέχει μια λίστα από DISKUnitObj
 * παρέχοντας συνολική πληροφορία για τη διαθεσιμότητα
 * αποθηκευτικών πόρων συστήματος
 */
public class DISKObj {

	private List<DISKUnitObj> diskUsage;

	public List getDiskUsage() {
		if (diskUsage == null) {
			diskUsage = new ArrayList<DISKUnitObj>();
		}

		return diskUsage;
	}
}
```

11.1.4 Η κλάση DISKUnitObj

```
package com.resources.Obj;

/**
 * @author Nikos Makris
 * Class το οποίο περιέχει πληροφορία για κάθε δίσκο συστήματος
 */
public class DISKUnitObj {

//	Όνομα Δίσκου
	private String drive;
```

```
//	Τύπος Δίσκου
	private String type;

//	Συνολικό ποσό Bytes
	private String space;

//	Διαθέσιμο ποσό Bytes
	private String available;

	public String getAvailable() {
		return available;
	}

	public void setAvailable(String available) {
		this.available = available;
	}

	public String getDrive() {
		return drive;
	}

	public void setDrive(String drive) {
		this.drive = drive;
	}

	public String getSpace() {
		return space;
	}

	public void setSpace(String space) {
		this.space = space;
	}

	public String getType() {
		return type;
	}

	public void setType(String type) {
		this.type = type;
	}
}
```

11.1.5 Η κλάση RAMObj

```
package com.resources.Obj;

/**
 * @author Nikos Makris
 * Class που περιέχει πληροφορία σχετικά με το συνολικό
 * και  και το διαθέσιμο ποσό Bytes της RAM
 */
public class RAMObj {

	private String totalMemory;

	private String freeMemory;

	public String getFreeMemory() {
		return freeMemory;
	}
```

```
	public void setFreeMemory(String freeMemory) {
		this.freeMemory = freeMemory;
	}

	public String getTotalMemory() {
		return totalMemory;
	}

	public void setTotalMemory(String totalMemory) {
		this.totalMemory = totalMemory;
	}

}
```

11.1.6 Η κλάση PCResources

```
package com.resources.Obj;

/**
 * @author Nikos Makris
 * Class το οποίο περιέχει objects σχετιζόμενα με την πληροφορία
 * για CPU Usage, Ram Usage , Disk Usage
 */
public class PCResources {

	private CPUObj CPUObj;

	private DISKObj diskObj;

	private RAMObj ramObj;

	public CPUObj getCPUObj() {
		return CPUObj;
	}

	public void setCPUObj(CPUObj CPUObj) {
		this.CPUObj = CPUObj;
	}

	public DISKObj getDiskObj() {
		return diskObj;
	}

	public void setDiskObj(DISKObj diskObj) {
		this.diskObj = diskObj;
	}

	public RAMObj getRamObj() {
		return ramObj;
	}

	public void setRamObj(RAMObj ramObj) {
		this.ramObj = ramObj;
	}
}
```