Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ ΧΡΟΝΟΔΡΟΜΟΛΟΓΗΣΗΣ ΣΕ ΠΕΡΙΒΑΛΛΟΝΤΑ ΥΠΟΛΟΓΙΣΤΙΚΟΥ ΝΕΦΟΥΣ** **SCHEDULING ALGORITHMS IN CLOUD COMPUTING ENVIRONMENTS**
Ονοματεπώνυμο Φοιτητή	**Ιάσων – Χαράλαμπος Τσετσενέκος**
Πατρώνυμο	**Παναγιώτης**
Αριθμός Μητρώου	**ΜΠΣΠ/ 12085**
Επιβλέπων	**Καθηγητής, Χρήστος Δουληγέρης**

Ημερομηνία Παράδοσης **Μάρτιος 2017**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

Περιεχόμενα

Περίληψη

Η χρονοδρομολόγηση αποτελεί αναπόσπαστο κομμάτι της τεχνολογίας της πληροφορικής καθώς χρησιμοποιείται κατά κόρον στα λειτουργικά συστήματα, όπου σκοπό έχει να κατανείμει τις ως προς εκτέλεση διεργασίες. Η μεθοδολογία αυτής της κατανομής των διεργασιών πραγματοποιείται με αλγορίθμους χρονοδρομολόγησης. Παράλληλα, η τεχνολογία της υπολογιστικής «νέφους» έχει σημειώσει τα τελευταία χρόνια ραγδαία ανάπτυξη στην επιστήμη της πληροφορικής και συνεχώς εξελίσσεται, με πολλές εφαρμογές να βασίζονται πάνω σε αυτή. Χαρακτηριστικό της συγκεκριμένης τεχνολογίας είναι η επαναχρησιμοποίηση υπολογιστικών πόρων η οποία επιτυγχάνεται με συγκεκριμένες μεθοδολογίες. Μια από αυτές είναι και η χρησιμοποίηση αλγορίθμων χρονοδρομολόγησης.

Αντικείμενο της παρούσας εργασίας είναι η παρουσίαση και ανάλυση αλγορίθμων χρονοδρομολόγησης όπου εφαρμόζονται στην τεχνολογία υπολογιστικής «νέφους». Στην πρώτη ενότητα γίνεται η περιγραφή της Υπολογιστικής Νέφους και των βασικών εννοιών που σχετίζονται με αυτήν. Αναφέρονται τα χαρακτηριστικά και οι υπηρεσίες που προσφέρει στις υπάρχουσες πλατφόρμες καθώς και τα θέματα ασφαλείας μπορεί να προκύψουν.

Στην δεύτερη ενότητα γίνεται εισαγωγή στους τύπους εικονικοποίησης και πώς αυτή σχετίζεται με την υπολογιστική νέφους. Επίσης επισημαίνονται οι αρχιτεκτονικές εικονικοποίησης καθώς και οι οντότητες που την αποτελούν. Στην τρίτη ενότητα γίνεται αναφορά στο πρόβλημα του χρονοπρογραμματισμού καθώς και στους τύπους χρονοπρογραμματισμού που υπάρχουν. Τέλος γίνεται εισαγωγή στις ροές εργασίας, οι οποίες έχουν εφαρμογή στην υπολογιστική νέφους.

Στην τέταρτη ενότητα παρουσιάζονται διάφορα παραδείγματα αλγορίθμων χρονοδρομολόγησης, καθώς και τα πλεονεκτήματά – μειονεκτήματά τους έναντι των άλλων. Οι αλγόριθμοι αυτοί ποικίλουν καθώς στοχεύουν στην βελτίωση διαφορετικών προβλημάτων. Τέλος στην πέμπτη ενότητα πραγματοποιείται προσομοίωση διαφόρων αλγορίθμων της τέταρτης ενότητας με την βοήθεια της πλατφόρμας WorkflowSim. Τα αποτελέσματα δείχνουν την συμπεριφορά τους ως προς τον χρόνο εκτέλεσης και την ανάθεση πόρων σε διαφορετικές συνθήκες.

Abstract

Scheduling is an integral part of information technology (IT) as it is applied to operating systems, aiming on task allocation. The allocation is being managed by scheduling algorithms. Furthermore, cloud computing technology has emerged in the world of computer science where most of the applications are based on it. Main characteristic of this technology is the reusability of computational resources which is achieved by scheduling algorithms.

The subject of this thesis is the analysis and demonstration of scheduling algorithms that are used in cloud computing. In the first chapter cloud computing technology is described along with the applications that are based on it. The mentioned subjects are the characteristics and the services that are offered in the existing platforms, as well as the security issues that may arise.

In the second chapter, virtualization types are introduced and their association with cloud computing. Furthermore, virtualization architectures are presented, as well as the entities that compose it. In the third chapter, the scheduling problem and its types are being presented, along with workflows.

In the fourth chapter, examples of scheduling algorithms are presented, as long as the advantages and the disadvantages between them. The subjects of these algorithms vary as they aim to improve different scheduling problems. At last, in the fifth chapter, simulation of some of the aforementioned algorithms is performed with the help of WorkflowSim platform. The results show their behavior, in terms of execution time and resource allocation under different circumstances.

Εισαγωγή

Η Υπολογιστική Νέφους είναι μία τεχνολογία που έχει αποκτήσει φήμη τα τελευταία χρόνια και έχει αρχίσει να εφαρμόζεται ανά τον κόσμο. Παρέχει τη στιγμή που ζητούνται υπηρεσίες υλικού, λογισμικού, αποθήκευσης και υποδομών, δυναμικά στον χρήστη με βάση την συμφωνία που έχει ορισθεί με τον πάροχο. Επιπρόσθετα, προσφέρει την δυνατότητα φιλοξενίας εφαρμογών που χρησιμοποιούνται από επιχειρήσεις, μέσα κοινωνικής δικτύωσης και επιστημονικούς οργανισμούς.

Οι συγκεκριμένες υπηρεσίες δεν θα ήταν διαθέσιμες αν δεν είχε δημιουργηθεί η τεχνολογία της εικονικοποίησης. Ως εικονικοποίηση ορίζεται η τμηματοποίηση των φυσικών πόρων του υλικού, όπως της υπολογιστικής ισχύος, της μνήμης, του αποθηκευτικού χώρου κτλ. Αν κάθε τμήμα αυτών των πόρων μπορεί ενοποιηθεί μπορεί να δημιουργήσει έναν εικονικό υπολογιστή/διακομιστή ο οποίος έχει την δυνατότητα να φιλοξενήσει ξεχωριστά λειτουργικά συστήματα. Στην δυνατότητα αυτή βασίζεται η υπολογιστική νέφους η οποία μπορεί να χρησιμοποιεί διάφορα τμήματα του φυσικού υλικού, να εκτελεί συγκεκριμένες διεργασίες και στην συνέχεια να τα ελευθερώνει.

Από την άλλη μεριά, η υπολογιστική νέφους υποφέρει από διάφορους τύπους προβλημάτων όπως η ασφάλεια, η απόδοση , η διαχείριση δεδομένων, η ανάθεση πόρων, η ανοχή σε σφάλματα και η ανάθεση διεργασιών. Για να μειωθούν τα παραπάνω προβλήματα οι επιστήμονες έχουν αρχίσει να χρησιμοποιούν διάφορους αλγορίθμους χρονοπρογραμματισμού. Τα τελευταία χρόνια έχουν προταθεί πολλοί τέτοιοι αλγόριθμοι οι οποίοι στοχεύουν στην βελτίωση της απόδοσης, των σφαλμάτων καθώς και στην ισομερή ανάθεση πόρων. Η εφαρμογή τους, στις περισσότερες των περιπτώσεων, πραγματοποιείται πριν την εκτέλεση των διεργασιών. Δημιουργεί μια λίστα από διεργασίες προς εκτέλεση και εφαρμόζει διάφορους κανόνες ως προς την ανάθεσή τους στους πόρους με στόχο την εκάστοτε βελτίωση του προβλήματος.

Αντικείμενο Εργασίας

Ο χρονοπρογραμματισμός διεργασιών αποτελεί βασικό κομμάτι των λειτουργικών συστημάτων και ευθύνεται για την ομαλή ανάθεση και εκτέλεσή τους, στους πόρους του συστήματος. Αυτό έγκειται στην επιλογή της κατάλληλης διεργασίας ώστε να τηρηθούν όλες οι αντίστοιχες προθεσμίες και χρονικοί περιορισμοί. Στην υπολογιστική νέφους οι περιορισμοί ποικίλουν καθώς δεν έχουν ως μόνο στόχο την εκτέλεση διεργασιών σε ένα ορισμένο χρονικό διάστημα, άλλα και στην ορθή ανάθεση των πόρων, στην μείωση του κόστους εκτέλεσης, μέχρι και στην μείωση της κατανάλωσης ενέργειας.

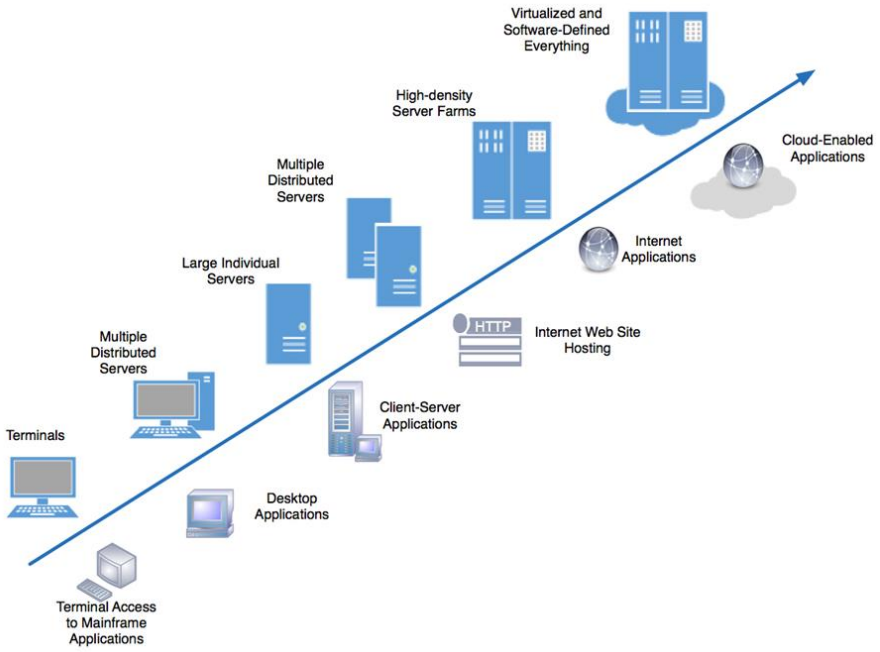
Η παρούσα μεταπτυχιακή διατριβή έχει ως αντικείμενο μελέτης τον χρονοπρογραμματισμό διεργασιών για περιβάλλον υπολογιστικής νέφους. Στόχος της είναι η παρουσίαση διαφόρων αλγορίθμων, οι οποίοι έχουν εφαρμογή σε συνθήκες πραγματικού χρόνου καθώς και σε ροές εργασίας. Η ανάλυσή και η σύγκρισή τους πραγματοποιήθηκε μέσω προσομοίωσης για διαφορετικά κριτήρια όπως ο συνολικός χρόνος εκτέλεσης και η ισομερής κατανομή των διεργασιών στους πόρους. Ο τύπος χρονοπρογραμματισμού στον οποίο εφαρμόσθηκαν οι αλγόριθμοί είναι αυτός της ροής εργασίας.

1 Υπολογιστική Νέφους

Στις αρχές της δεκαετίας του 60 , ο John McCarthy δήλωσε ότι “η αξιοποίηση του χρόνου χρήσης των υπολογιστικών πόρων, κάποια μέρα θα μπορεί να οργανωθεί και ως δημόσια παροχή”. Πάνω σε αυτή την άποψη εφαρμόζεται και η Υπολογιστική νέφους. Στις δεκαετίες 60 και 70 , οι υπολογιστές συνδεόντουσαν και σχημάτιζαν μια συστοιχία (cluster) δημιουργώντας ένα υπερ-υπολογιστή. Η συγκεκριμένη τεχνολογία έγινε γνωστή στην βιομηχανία και χρησιμοποιούνταν από πολλές εταιρίες πληροφορικής καθώς έδινε την δυνατότητα εξισορρόπησης του υπολογιστικού φόρτου μεταξύ των μηχανημάτων της συστοιχίας. Αυτό γινόταν εφικτό με την βοήθεια ειδικά σχεδιασμένων πρωτοκόλλων. Ο χρήστης της συγκεκριμένης τεχνολογίας δεν γνώριζε σε ποίο μηχάνημα θα τρέξει το πρόγραμμά του, καθώς η συστοιχία αναλάμβανε την εκτέλεσή του στο καλύτερο δυνατό μηχάνημα την συγκεκριμένη στιγμή.[1]

Κατά την δεκαετία του 90, με την ανάπτυξη του διαδικτύου η τεχνολογία της συστοιχίας αντικαταστάθηκε από τους διακομιστές (servers), οι οποίοι παρείχαν υψηλή ισχύ με αποτέλεσμα να υποστηρίζουν “φορτίο” από το διαδίκτυο. Από τότε μέχρι σήμερα , όλο και περισσότερες υπηρεσίες προσφέρονται μέσω του διαδικτύου, με τις απαιτήσεις για υπολογιστική ισχύ και αποθηκευτικό χώρο να γίνονται όλο και μεγαλύτερες. Σήμερα, όλο και περισσότερο πραγματοποιείται “μετακίνηση” των εφαρμογών από τους προσωπικούς υπολογιστές προς τους διακομιστές , κυρίως λόγο της ευριζωνικής σύνδεσης και των υψηλών ταχυτήτων πρόσβασης στο διαδίκτυο. Η απαίτηση για υψηλή υπολογιστική ισχύ καλύπτεται από εξειδικευμένους παρόχους προσφέροντας την δυνατότητα στους χρήστες να μοιράζονται τους ίδιους πόρους, μεγιστοποιώντας την αποδοτικότητα και μειώνοντας το κόστος.

Το 1999, η Salesforce.com ξεκίνησε την παροχή υπηρεσιών νέφους απο τους διακομιστές της σε επιχειρήσεις. Πρόκειται για ένα CRM το οποίο υποστηρίζεται σε διάφορες γλώσσες και πλατφόρμες. Το 2002 , η Amazon παρουσίασε το Amazon Web Services (AWS), μια λύση που παρείχε υπηρεσίες αποθήκευσης και υπολογιστικής ισχύος. Επιπρόσθετα, το 2006 η Amazon παρουσίασε το Elastic Compute Cloud (EC2) με το οποίο μικρές επιχειρήσεις καθως και χρήστες μπορούσαν να “τρέξουν” τις εφαρμογές τους σε περιβάλλον νέφους. Τέλος το2009 η Google ξεκίνησε την παροχή υπηρεσιών για επιχειρήσεις βασισμένες για περιηγητές ιστοσελίδων.[1] Στην Εικόνα 1.1 βλέπουμε την εξέλιξη των υποδομών της υπολογιστικής νέφους και των εφαρμογών που υποστηρίζουν.



Εικόνα 1.1 - Εξέλιξη τεχνολογίας Υπολογιστικής Νέφους (Πηγή www.slideshare.net/ranpararipal/overview-of-computing-paradigm)

1.1 Γενική έννοια υπολογιστικής νέφους

Η χρήση του διαδικτύου από κοινούς χρήστες και επιχειρήσεις είναι μέρος της καθημερινής ζωής. Μεγάλος όγκος πληροφορίας είναι διαθέσιμος οποιαδήποτε στιγμή από τα περισσότερα μέρη του κόσμου, το οποίο δεν ήταν εφικτό τα προηγούμενα χρόνια. Σήμερα, πολλοί χρήστες μπορούν να διαβάζουν την αλληλογραφία τους διαδικτυακά, να συγγράφουν κείμενα, καθώς και να δημιουργούν και να μοιράζονται "εικονικά" φωτογραφικά άλμπουμ. Χρησιμοποιούν εφαρμογές όπου αποθηκεύουν προσωπικά δεδομένα σε αποθηκευτικούς χώρους που βρίσκονται στο διαδίκτυο, αντί των προσωπικών τους υπολογιστών. Τον τελευταίο καιρό, οι συγκεκριμένες υπηρεσίες καλούνται και "Υπηρεσίες υπολογιστικού νέφους". Ο τίτλος είναι μεταφορικός, καθώς οι χρήστες δεν έχουν ακριβή εικόνα πού αποθηκεύονται τα δεδομένα τους.

Οι υπηρεσίες υπολογιστικού νέφους μπορεί να προσφέρονται στους χρήστες δωρεάν αλλά και με πληρωμή ανά χρήση, το οποίο σημαίνει ότι οι κοινοί χρήστες και οι επιχειρήσεις μπορούν να αποφύγουν την εγκατάσταση επιπλέον λογισμικού. Επίσης, μπορούν να ζητήσουν επιπλέον υπολογιστική ισχύ ή αποθηκευτικό χώρο καθώς και να δημιουργήσουν το δικό τους ιδιωτικό νέφος.[2]

Η υπολογιστική νέφους ανήκει στην ομάδα των κατανεμημένων συστημάτων. Σε αυτά τα περιβάλλοντα, πολλοί πόροι βρίσκονται σε διαφορετικές τοποθεσίες όπου καθένας εκτελεί τις δικές του διεργασίες ενώ παράλληλα μοιράζεται κοινούς πόρους και πληροφορίες. Επίσης είναι απαραίτητο να συντονίζονται μεταξύ τους με σκοπό την επιτυχή εκτέλεση των διεργασιών. Στα κατανεμημένα συστήματα ανήκουν επίσης και άλλα είδη υπολογισμού:

Κατανεμημένη Υπολογιστική (Distributed Computing)

Είναι μια ομάδα υπολογιστών που είναι στενά συνδεδεμένοι με δικτύωση υψηλών ταχυτήτων οι οποίοι συνεργάζονται για κοινό σκοπό. Μπορούμε να πούμε ότι αποτελούν ένα ενιαίο υπολογιστή, και λειτουργούν ως επί το πλείστον σε κοινόχρηστη μνήμη. Οι υπολογιστές που σχηματίζουν μια συστάδα παρουσιάζουν ομοιογένεια ως προς το λειτουργικό τους σύστημα και τις προδιαγραφές του υλικού. Συνήθως βρίσκονται σε ενιαία θέση και δεν διανέμονται. Ωστόσο, μπορεί να περιγραφεί ως μια ένωση από τους τομείς του παράλληλου και της υψηλής απόδοσης και διαθεσιμότητας του κατανεμημένου υπολογισμού (computing).[29]

Υπολογιστική Πλέγματος (Grid Computing)

Η υπολογιστική πλέγματος είναι μια μορφή κατανεμημένου υπολογιστικού συστήματος η οποία προσπαθεί να ενώσει υπολογιστικούς πόρους που βρίσκονται σε διαφορετικές τοποθεσίες ανά τον κόσμο. Αποτελεί μια συλλογή από υπολογιστικά συστήματα τα οποία αναφέρονται συχνά σαν κόμβοι ή πόροι και μπορεί να χρησιμοποιηθούν από όλους τους χρήστες του πλέγματος. Στο χρήστη το σύστημα φαίνεται να είναι ένα τεράστιο ενιαίο υπολογιστικό σύστημα. [29]

Υπολογιστική Κοινής Ωφέλειας (Utility Computing)

Στα περιβάλλοντα κοινής ωφέλειας, οι χρήστες αναθέτουν μια τιμή «χρησιμότητας» στις εργασίες τους, όπου η χρησιμότητα είναι μια σταθερή ή χρονικώς μεταβαλλόμενη τιμή που περιλαμβάνει διάφορους περιορισμούς ποιότητας των υπηρεσιών (προθεσμία, σημασία, ικανοποίηση). Η τιμή αυτή αντιπροσωπεύει το ποσό που είναι διατεθειμένοι οι χρήστες να πληρώσουν σε ένα πάροχο για να ικανοποιήσει τα αιτήματά τους. [29]

1.2 Ορισμός

Η λέξη "νέφος" αποτελεί μια μεταφορική έννοια η οποία χρησιμοποιείται για να περιγράψει την τοπολογία του δικτύου από την πλευρά των χρηστών του, καθώς δεν γνωρίζουν λεπτομέρειες για το πού βρίσκεται η συγκεκριμένη υποδομή. Δεν υπάρχει σαφής ορισμός για την υπολογιστική νέφους. Σχηματικά θα μπορούσαμε να πούμε ότι οι πόροι ανασύρονται από ένα «νέφος» πόρων όταν αποδοθούν σε ένα χρήστη και αυτές επιστρέφουν σε αυτό όταν απελευθερωθούν. Ένα «νέφος» είναι ένα σύνολο μηχανημάτων και διαδικτυακών υπηρεσιών

που υλοποιούν την υπολογιστική νέφους. Παρακάτω παραθέτουμε κάποιες προσπάθειες ορισμού του από διάφορους οργανισμούς:

Σύμφωνα με τον ινστιτούτο **IEEE** (Institute of Electrical and Electronics Engineers) η υπολογιστική νέφους είναι:

> «Ένα παράδειγμα οπού η πληροφορία αποθηκεύεται μόνιμα σε διακομιστές που βρίσκονται στο διαδίκτυο και αποθηκεύονται προσωρινά στους χρήστες».

Το Πανεπιστήμιο του Μπέρκλεϋ ορίζει ότι η υπολογιστική νέφους:

> «Αναφέρεται ταυτόχρονα στις εφαρμογές που διανέμονται ως υπηρεσίες μέσω του διαδικτύου καθώς και τις υποδομές (υλικό, λογισμικό) που παρέχουν τις υπηρεσίες αυτές».

Σύμφωνα με το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST) η υπολογιστική νέφους ορίζεται ως:

> «Ένα μοντέλο που επιτρέπει την εύκολη, on-demand (τη στιγμή που ζητείται) πρόσβαση μέσω δικτύου σε ένα "κοινό ταμείο" από παραμετροποιημένους υπολογιστικούς πόρους (π.χ. Δίκτυα, διακομιστές, αποθηκευτικό χώρο, εφαρμογές και υπηρεσίες) οι οποίοι μπορούν πολύ εύκολα να παρακολουθηθούν και να αποδοθούν με πολύ μικρή παρέμβαση της διαχείρισης, ή αλληλεπίδρασης από τον πάροχο των υπηρεσιών».

1.2.1 Κατηγορίες Νέφους

Η τεχνολογία της υπολογιστικής νέφους απαιτεί διαφορετικού μεγέθους υποδομές, με διαφορετική διαχείριση και ποικιλία χρηστών. Παρακάτω παραθέτουμε τέσσερις βασικές κατηγορίες νέφους (Εικόνα 1.2).

Ιδιωτικό Νέφος

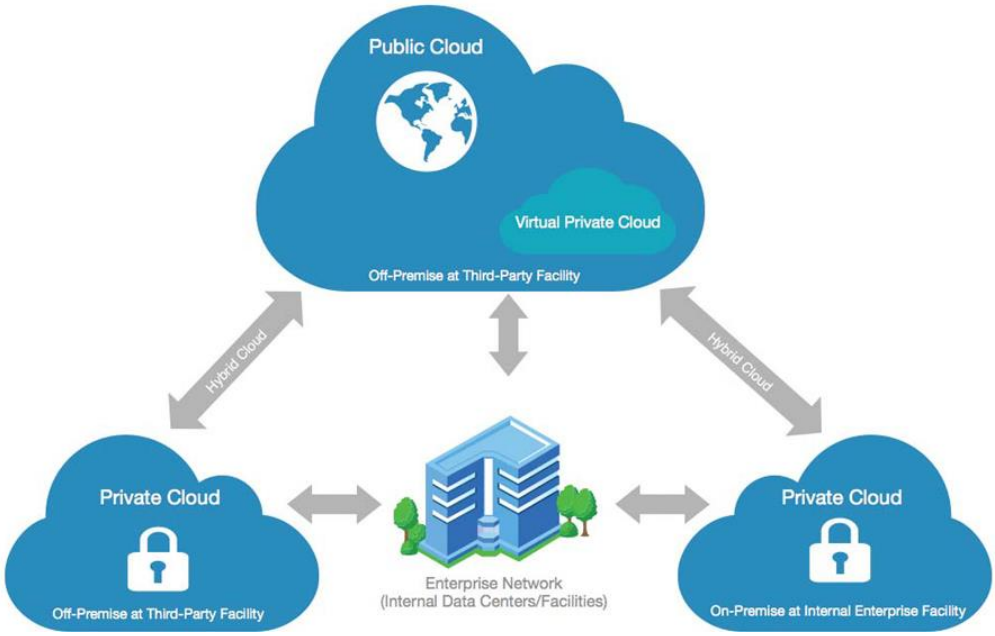
Η υποδομή του ιδιωτικού νέφους διαχειρίζεται και συντηρείται αποκλειστικά και μόνο από έναν οργανισμό, και η τοποθεσία του μπορεί να βρίσκεται εντός ή εκτός του κτιρίου του οργανισμού. Αυτό βέβαια αποτελεί σημαντικό μειονέκτημα καθώς το κόστος υλοποίησης και συντήρησης είναι υψηλό. Παράλληλα, τα πλεονεκτήματα που προκύπτουν είναι η ασφαλής διαχείρισή του, η καλύτερη προστασία ευαίσθητων δεδομένων, και τέλος η ευελιξία στις απαιτήσεις του χρήστη καθώς δίνει την δυνατότητα αποθήκευσης μεγάλου όγκου δεδομένων προσφέροντας υψηλές ταχύτητες μετάδοσης.[3]

Δημόσιο Νέφος

Το δημόσιο νέφος είναι διαθέσιμο για το ευρύ κοινό ή για επιχειρήσεις, ενώ κατέχεται και διαχειρίζεται από οργανισμούς/παρόχους που εμπορεύονται υπηρεσίες νέφους. Ως μοντέλο χαρακτηρίζεται από πολλά πλεονεκτήματα μερικά από τα ποία είναι τα εξής : οι υπηρεσίες προσφέρονται στους χρήστες με ασφάλεια, ελαστικότητα και συνεχή διαθεσιμότητα, χαρακτηρίζεται από μεγάλη ευελιξία λόγω της άμεσης διάθεσης υπηρεσιών και η χρέωση αφορά μόνο τις υπηρεσίες που θα χρησιμοποιηθούν.[3]

Υβριδικό Νέφος

Η υποδομή του υβριδικού νέφους ουσιαστικά αποτελεί ένα συνδυασμό του ιδιωτικού και κοινόχρηστου νέφους που έχει σαν στόχο να εκμεταλλευτεί τα πλεονεκτήματά τους. Βασικό χαρακτηριστικό του υβριδικού νέφους είναι το χαμηλότερο κόστος του σε σχέση με το ιδιωτικό, ενώ μπορεί να αποθηκεύσει δεδομένα ταυτόχρονα και στα δύο νέφη. Τα σημαντικά και απόρρητα δεδομένα αποθηκεύονται στο ιδιωτικό μέρος του υπολογιστικού νέφους, ενώ τα προσωπικά δεδομένα που είναι λιγότερο σημαντικά αποθηκεύονται στο δημόσιο μέρος του υπολογιστικού νέφους.[3]



Εικόνα 1.2 - Κατηγορίες Υπολογιστικού Νέφους (Πηγή www.actividgetechnologies.com/PublishingImages/integrated-security-solutions-for-hybrid-cloud-deployments1/hybrid%20cloud.jpg

Κοινοτικό Νέφος

Το συγκεκριμένο μοντέλο διαθέτει υποδομή η οποία είναι διαμοιρασμένη από πολλούς οργανισμούς και εξυπηρετεί συγκεκριμένη κοινότητα. Η κοινότητα αυτή έχει ως κοινό τόπο κάποιο συγκεκριμένο στόχο ή ενδιαφέρον. Αυτό το μοντέλο έχει σαν χαρακτηριστικό να μπορεί να το διαχειρίζεται ένας οργανισμός ή την εποπτεία του να την έχει ένας τρίτος οργανισμός ή επιχείρηση.[3]

Πίνακας 1. Σύγκριση κατηγοριών Νέφους

Χαρακτηριστικά	**Δημόσιο Νέφος**	**Ιδιωτικό Νέφος**	**Υβριδικό Νέφος**	**Κοινοτικό Νέφος**
Κόστος κατασκευής κέντρου δεδομένων στον χρήστη	Κανένα	Υψηλό	Μεσαίο	Ποικίλει ανάλογα τις κοινότητες
Κόστος συντήρησης και λειτουργίας στον πάροχο	Χαμηλότερο	Υψηλότερο	Μικρότερο από το ιδιωτικό νέφος	Παρόμοιο με το ιδιωτικό νέφος
Μέγεθος κέντρου δεδομένων	Πολύ μεγάλο	Μεγάλο	Μικρότερο από το ιδιωτικό νέφος	Μεγαλύτερο από το ιδιωτικό νέφος αλλά μικρότερο από το δημόσιο νέφος
Έλεγχος και προσαρμοστικότητα υποδομής	Περιορισμένη	Πλήρης έλεγχος	Πλήρης έλεγχος κατά το δημόσιο νέφος, περιορισμένος έλεγχος κατά το ιδιωτικό νέφος	Υψηλός αλλά περιορισμένος από την πολιτική των κοινοτήτων
Επίπεδο	Χαμηλότερο	Υψηλότερο	Μεσαίο	Υψηλό

εμπιστοσύνης				
Τοποθεσία υποδομής	Εκτός κτηρίου	Εντός κτηρίου	Εντός και εκτός κτηρίου	Εντός της εγκατάστασης της κοινότητας
Κάτοχος υποδομής	Παρόχου	Πελάτης	Ο πάροχος κατέχει το δημόσιο μέρος και ο πελάτης το ιδιωτικό	Διαμοιράζεται μεταξύ των κοινοτήτων

1.2.2 Μοντέλα και υπηρεσίες

Σύμφωνα με το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST), οι οντότητες που εμπλέκονται στην υπολογιστική νέφους είναι ο καταναλωτής, ο οποίος διατηρεί επαγγελματική σχέση με τον πάροχο, ο πάροχος όπου προσφέρει υπηρεσίες υπολογιστικού νέφους στους καταναλωτές, ο φορέας ο οποίος είναι ο ενδιάμεσος κρίκος μεταξύ του παρόχου και του καταναλωτή και προσφέρει συνδεσιμότητα και την μεταφορά τον υπηρεσιών, ο μεσάζων όπου διαχειρίζεται τις υπηρεσίες νέφους και τις σχέσεις μεταξύ καταναλωτή και παρόχου και τέλος ο ελεγκτής όπου πραγματοποιεί ελέγχους σχετικά με την ποιότητα, την απόδοση και την ασφάλεια των υπηρεσιών νέφους [5]. Η αντιστοίχηση υπηρεσίας και υποδομής παρουσιάζεται στην Εικόνα 1.3.

Software as a Service (SaaS)

Το μοντέλο SaaS επιτρέπει στους χρήστες να νοικιάζουν ή να δανείζονται υπηρεσίες λογισμικού διαδικτυακά χωρίς να χρειάζεται η αγορά και η εγκατάστασή τους στον προσωπικό υπολογιστή. Παρέχει ένα ολοκληρωμένο περιβάλλον με εγκατεστημένες εφαρμογές και οι χρήστες μπορούν να έχουν πρόσβαση σε στα αρχεία τους μέσω ενός περιηγητή ιστοσελίδων. Το μεγαλύτερο πλεονέκτημα του μοντέλου αυτού είναι το μειωμένο κόστος. Οι προμηθευτές λογισμικού συντηρούν το λογισμικό διαδικτυακά αντί να υποστηρίζουν τους χρήστες ξεχωριστά μέσω τηλεφώνου. Από την άλλη πλευρά, οι χρήστες δεν είναι υποχρεωμένοι να πληρώσουν όλο το ποσό της αγοράς του λογισμικού, αντ'αυτού πληρώνουν μόνο ένα συγκεκριμένο ποσό για το διάστημα που θέλουν να το χρησιμοποιήσουν.[4]

Πλεονεκτήματα Μοντέλου SaaS:

- Μειώνουν ή εξαλείφουν την ανάγκη ενός κέντρου δεδομένων εντός κτιριακών εγκαταστάσεων
- Εξαλείφουν την ανάγκη για διαχείριση των εφαρμογών
- Επιτρέπουν στους χρήστες να πληρώσουν ανάλογα με την ζήτηση
- Προσφέρουν επεκτασιμότητα χώρου δεδομένων και υπολογιστικής ισχύ
- Προσφέρουν πρόσβαση σε εφαρμογές ανεξαρτήτου συσκευής χρήστη
- Αυξάνουν τις πιθανότητες ανάκτησης του συστήματος σε περίπτωση βλάβης

Platform as a Service (PaaS)

Το μοντέλο PaaS παρέχει ένα σύνολο από πόρους υλικού και λογισμικού προς τους προγραμματιστές με σκοπό την δημιουργία εφαρμογών εντός του υπολογιστικού νέφους. Ανάλογα με τις ανάγκες, οι πόροι αυτοί μπορούν να είναι σε πλατφόρμα Windows ή Linux. Χρησιμοποιώντας το μοντέλο PaaS, οι προγραμματιστές καταργούν την ανάγκη αγοράς και συντήρησης υλικού, την εγκατάσταση λειτουργικού συστήματος καθώς και άλλου λογισμικού (π.χ. βάσεις δεδομένων). Έτσι, οι πόροι δεν βρίσκονται τοπικά αλλά στις εγκαταστάσεις του παρόχου. Αυτό προσφέρει την δυνατότητα στους χρήστες να αυξήσουν ή να μειώσουν τους

πόρους ανάλογα με τις απαιτήσεις της εφαρμογής, ενώ η χρέωση για τους πόρους εξαρτάται από την χρήση τους.[4]

Πλεονεκτήματα Μοντέλου PaaS:

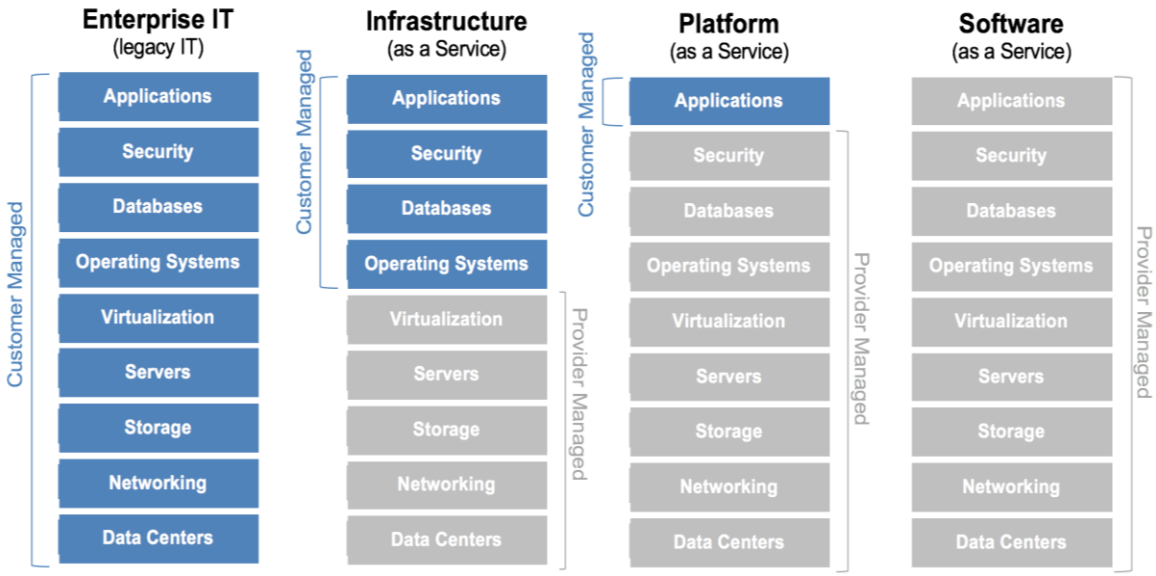
- Μειώνουν το συνολικό κόστος ιδιοκτησίας καθώς οι εταιρείες δεν χρειάζεται να αποκτήσουν και να συντηρήσουν υλικό για διακομιστές, εγκαταστάσεις ρεύματος και αποθήκευσης δεδομένων
- Μειώνουν το βάρος της διαχείρισης της υπηρεσίας καθώς αναλαμβάνεται από τους υπαλλήλους του παρόχου.
- Επιτρέπει στους υπαλλήλους της εταιρείας να επικεντρωθούν στην ανάπτυξη νέων εφαρμογών αντί της συντήρησης της υπηρεσίας νέφους
- Επεκτασιμότητα πόρων ανάλογα τις απαιτήσεις της εφαρμογής. Οι εταιρείες πληρώνουν μόνο ότι «καταναλώνουν»

Infrastructure as a Service (IaaS)

Το μοντέλο IaaS παρέχει κυρίως πόρους υλικού όπως αποθηκευτικούς χώρους, δικτυακή σύνδεση και υπολογιστική ισχύ σε χρήστες, οι οποίοι από την πλευρά τους είναι υπεύθυνοι για την εγκατάσταση και συντήρηση των εφαρμογών που θα χρησιμοποιήσουν. Οι υπολογιστικοί πόροι που παρέχονται είναι εικονικοιποιημένοι (virtual machines) και οι χρήστες θα πρέπει να προσαρμόσουν τους συγκεκριμένους πόρους για το εκάστοτε σύστημα που θέλουν να δημιουργήσουν.[4]

Πλεονεκτήματα Μοντέλου IaaS:

- Κατάργηση του τοπικού κέντρου δεδομένων
- Ευκολία στην επέκταση των πόρων υλικού
- Μειωμένα έξοδα για την απόκτηση υλικού
- Πληρωμή ανάλογα με την χρήση των πόρων
- Μειωμένο προσωπικό για την συντήρηση της υποδομής
- Πλήρης διαχείριση του συστήματος



Εικόνα 1.3 - Παροχές Μοντέλων (Πηγή mycloudblog7.wordpress.com/2013/06/19/who-manages-cloud-iaas-paas-and-saas-services/)

1.2.3 Χαρακτηριστικά και Ιδιότητες

Παρακάτω παρουσιάζονται τα χαρακτηριστικά της υπολογιστικής νέφους.

- Εξυπηρέτηση κατά απαίτηση (On Demand Self Service)

Ένας καταναλωτής μπορεί να δεσμεύσει από μόνος του τους υπολογιστικούς πόρους που χρειάζεται, όπως χρόνο στον server και αποθηκευτικό χώρο στο δίκτυο, ανάλογα με τις ανάγκες του αυτόματα, χωρίς να απαιτείται ανθρώπινη αλληλεπίδραση με το φορέα παροχής κάθε υπηρεσίας.

- Ευρεία πρόσβαση στο Δίκτυο (Broad Network Access)

Οι δυνατότητες είναι διαθέσιμες μέσω του δικτύου και προσβάσιμες μέσω τυποποιημένων μηχανισμών που προωθούν την χρήση από ετερογενείς thin ή thick client πλατφόρμες (π.χ. κινητά τηλέφωνα, φορητούς υπολογιστές και PDA's).

- Ανάθεση Πόρων

Οι υπολογιστικοί πόροι του παρόχου χρησιμοποιούνται για να εξυπηρετήσουν πολλαπλούς καταναλωτές με τη χρήση του μοντέλου πολλαπλών μισθωτών (multi-tenant), με τους διάφορους φυσικούς και εικονικούς πόρους να ανατίθενται δυναμικά και εκ νέου ανάλογα με τη ζήτηση των καταναλωτών. Υπάρχει μια αίσθηση ανεξαρτησίας από τον τόπο στο γεγονός ότι ο πελάτης δεν έχει γενικά κανέναν έλεγχο ή γνώση σχετικά με την ακριβή τοποθεσία των παρεχόμενων πόρων, αλλά μπορεί να είναι σε θέση να προσδιορίζει την τοποθεσία σε ένα υψηλότερο επίπεδο αφαίρεσης (π.χ. χώρα, κράτος, ή datacenter). Παραδείγματα πόρων αποτελούν οι αποθηκευτικοί χώροι, η επεξεργασία, η μνήμη, το εύρος ζώνης του δικτύου, καθώς και οι εικονικές μηχανές.

- Ελαστικότητα

Οι πόροι μπορούν να δεσμευτούν προς χρήση γρήγορα και ελαστικά, σε ορισμένες περιπτώσεις αυτόματα, έτσι ώστε να εμφανιστούν άμεσα ως μη διαθέσιμοι (scale out) και επίσης να αποδεσμευτούν γρήγορα για να εμφανιστούν ξανά ως διαθέσιμοι (scale in). Για τον καταναλωτή, οι διαθέσιμες δυνατότητες για δέσμευση και χρήση συχνά φαίνεται να είναι απεριόριστες και μπορούν να αγοραστούν ανά πάσα στιγμή και σε οποιαδήποτε ποσότητα.

- Έλεγχος Υπηρεσιών

Τα συστήματα cloud ελέγχουν και βελτιστοποιούν αυτόματα τη χρήση των πόρων, αξιοποιώντας μια δυνατότητα μέτρησης σε κάποιο επίπεδο αφαίρεσης που είναι κατάλληλο για το είδος της υπηρεσίας (π.χ. αποθήκευση, επεξεργασία, bandwidth, ενεργοί λογαριασμοί χρηστών). Η χρήση των πόρων μπορεί να παρακολουθείται, να ελέγχεται, και να παρουσιάζεται με τη μορφή reports, παρέχοντας διαφάνεια τόσο για τον πάροχο όσο και για τον καταναλωτή της χρησιμοποιούμενης υπηρεσίας.

- Ανάκτηση

Σε περίπτωση αποτυχίας της εφαρμογής θα πρέπει να υπάρχει δυνατότητα άμεσης ανάκτησης χωρίς διακοπή της υπηρεσίας. Αυτό σημαίνει ότι όταν υπάρχει πολιτική ανάκτησης, τότε σε περίπτωση σφάλματος η δευτερεύουσα (backup) υπηρεσία θα ενεργοποιείται και παράλληλα θα δημιουργείται καινούργια δευτερεύουσα υπηρεσία.[4]

1.2.4 Τρέχουσες Πλατφόρμες

Τα τελευταία χρόνια έχει αναπτυχθεί ένας αριθμός από πλατφόρμες, οι οποίες περιλαμβάνουν πολλαπλές υπηρεσίες σε κατανεμημένα συστήματα. Λύσεις υπηρεσιών νέφους έχουν εμφανισθεί στο εμπόριο και ενισχύουν τις επιχειρήσεις να μεταβούν προς το μοντέλο όπου η υπηρεσίες ενοικιάζονται ανάλογα την χρήση, αντί του παλαιότερου μοντέλου όπου οι υπηρεσίες έπρεπε να αγορασθούν εξ' ολοκλήρου. Παρακάτω παρατίθενται μερικές από τις πλατφόρμες αυτές.[7]

Amazon EC2

Η συγκεκριμένη πλατφόρμα επιτρέπει στους χρήστες να δημιουργήσουν εικονικές μηχανές (virtual machines) δίνοντας ένα όνομα στην συγκεκριμένη οντότητα. Η οντότητα αυτή

λειτουργεί σαν ένας εικονικός διακομιστής όπου περιέχει το επιθυμητό υλικό και λογισμικό. Προσφέρει ελαστικότητα στους πόρους (αύξηση, μείωση), πλήρη έλεγχο των οντοτήτων, δημιουργία πολλαπλών οντοτήτων καθώς και δικτυακή ασφάλεια μέσω IPsec VPN.

Amazon S3

Η πλατφόρμα Amazon S3 παρέχει υποδομές για αποθήκευση και ανάκτηση δεδομένων μέσω διαδικτύου. Αυτή η υπηρεσία είναι χρήσιμη για τους προγραμματιστές καθώς μπορούν να αποθηκεύουν δεδομένα σε πολλαπλές συσκευές με ξεχωριστό αριθμό καταχώρησης (version control) ώστε να γίνεται ανάκτηση σε περίπτωση σφαλμάτων.

Google App Engine

Κυκλοφόρησε το 2008 και παρέχει στους χρήστες την δυνατότητα ανάπτυξης και φιλοξενίας διαδικτυακών εφαρμογών σε πολλαπλούς διακομιστές και κέντρα δεδομένων. Στην περίπτωση που χρησιμοποιείται μοντέλο PaaS, η πλατφόρμα υποστηρίζει τις γλώσσες προγραμματισμού Python και Java. Σε σύγκριση με την EC2 υστερεί ως προς ευρεία χρήση, καθώς δεν επιτρέπει ευελιξία στην υποδομή του συστήματος.

MapReduce

Η πλατφόρμα Map Reduce αρχικά αναπτύχθηκε από την Google, και αποτελεί ένα προγραμματιστικό μοντέλο το οποίο υποστηρίζει την επεξεργασία πολύ μεγάλων όγκων δεδομένων για κατανεμημένα συστήματα τα οποία είναι οργανωμένα σε συστοιχίες. Χρησιμοποιείται για την ανάπτυξη εφαρμογών οι οποίες επεξεργάζονται γρήγορα και παράλληλα τεράστιες ποσότητες δεδομένων. Τα τελευταία χρόνια η πλατφόρμα χρησιμοποιείται το μοντέλο ανοιχτού κώδικα Hadoop.

Dryad

Το Dryad αναπτύχθηκε από την Microsoft ως ένα μοντέλο περιγραφικού προγραμματισμού (π.χ. SQL) με σκοπό να εφαρμόζεται πάνω σε υποδομές υπολογιστικής και αποθήκευσης. Η γλώσσα DryadLINQ επιτρέπει την ανάπτυξη εφαρμογών μεγάλης κλίμακας δεδομένων σε συστοιχίες.

1.2.5 Ασφάλεια στο Υπολογιστικό Νέφος

Η ασφάλεια στο Υπολογιστικό νέφος είναι ένα από τα σημαντικότερα ζητήματα και έχει αποτελέσει το αντικείμενο συζήτησης τόσο των σχετικών με την ασφάλεια επιστημόνων, όσο και των στελεχών των επιχειρήσεων. Έτσι, είναι πλέον ένα κομμάτι του νέφους που το χαρακτηρίζει και δεν θα μπορούσε να λείπει μια περαιτέρω ανάλυση πάνω στο ζήτημα αυτό.

Η ασφάλεια είναι δύσκολο να επιτευχθεί για ποικίλους λόγους. Τα πιθανά σενάρια ποικίλλουν από πιθανές επιθέσεις κακόβουλων χρηστών (π.χ. hackers) σε υποδομές Δημόσιου νέφους, μέχρι θέματα διαρροής δεδομένων, διαγραφής τους και διάφορα θέματα ιδιωτικότητας (περιπτώσεις ανεπαρκούς προστασίας των δεδομένων από το σύστημα) . Από τη στιγμή που οι χρήστες αποθηκεύουν τα δεδομένα τους κάπου που δεν έχουν τον έλεγχο της προστασίας τους οι ίδιοι και δεν διασφαλίζεται απόλυτα η διαγραφή τους σε περίπτωση που χρειαστεί, υπάρχει αίσθηση ανασφάλειας και διστακτικότητα υιοθέτησης του συστήματος.

Οι πάροχοι υπηρεσιών υπολογιστικού νέφους παρέχουν κρυπτογράφηση των δεδομένων και σύστημα ταυτοποίησης για πρόσβαση στα δεδομένα, αλλά το γεγονός ότι τα δεδομένα είναι προσβάσιμα μέσω του διαδικτύου, τα καθιστά πιθανό στόχο τρίτων, καθώς τίποτα στο διαδίκτυο δεν είναι ποτέ απόλυτα ασφαλές. Έτσι πολλοί υποψήφιοι χρήστες φοβούνται πως πάντα θα υπάρχει η πιθανότητα υποκλοπής - αλλοίωσης των δεδομένων τους, ειδικά σε μεγάλες επιχειρήσεις όπου υπάρχει και πιο μεγάλο (οικονομικό συνήθως) κίνητρο. Τέλος, πολλές επιχειρήσεις αποφεύγουν τη χρήση των υπηρεσιών υπολογιστικού νέφους επειδή και ανταγωνιστές τους μπορεί να χρησιμοποιήσουν τον ίδιο πωλητή – πάροχο και μια ευπάθεια του συστήματος ίσως θέσει σε κίνδυνο ολόκληρο το νέφος.

Για τα δεδομένα που είναι αποθηκευμένα στο Υπολογιστικό νέφος, αναφερόμαστε στο μοντέλο IaaS και όχι στα δεδομένα που σχετίζονται με την εφαρμογή που τρέχουν στα επίπεδα PaaS ή SaaS. Τα έξι στοιχεία που ενδιαφερόμαστε όσον αφορά την αποθήκευση των

δεδομένων είναι: η εμπιστευτικότητα, η ακεραιότητα, η διαθεσιμότητα, ο έλεγχος, η εξουσιοδότηση και ο έλεγχος ταυτότητας.

Εμπιστευτικότητα (Confidentiality)

Να μην μπορεί να έχει πρόσβαση στα δεδομένα ενός χρήστη υπηρεσιών Νέφους, κάποιος τρίτος, χωρίς την άδεια του χρήστη.

Ακεραιότητα (Integrity)

Να μην μπορεί να τροποποιήσει τα δεδομένα ενός χρήστη υπηρεσιών Νέφους κάποιος τρίτος, χωρίς την άδεια του χρήστη.

Διαθεσιμότητα (Availability)

Πόσο συχνά είναι προσβάσιμα (μέσω διαδικτύου) τα δεδομένα και οι εφαρμογές ενός χρήστη του νέφους.

Έλεγχος Ταυτότητας (Authentication)

Όταν κάποιος έχει πρόσβαση σε δεδομένα και εφαρμογές μέσω υπηρεσιών νέφους, να διασφαλίζεται πως είναι αυτός που ισχυρίζεται πως είναι.

Εξουσιοδότηση (Authorization)

Να διασφαλίζεται πως ο κάθε χρήστης υπηρεσιών νέφους έχει την πρόσβαση στα δεδομένα και εφαρμογές, που του επιτρέπεται από το σύστημα.

Έλεγχος (Auditing)

Να διασφαλίζεται συνεχώς πως τα περιεχόμενα του νέφους είναι συνεπή.

2 Εικονικοποίηση

Ο όρος εικονικοποίηση περιγράφει ευρέως την τμηματοποίηση των πόρων του υποκείμενου φυσικού υλικού, ώστε να καλυφθούν αιτήματα υπηρεσίας προς το συγκεκριμένο υλικό. Με την εικονική μνήμη για παράδειγμα ο υπολογιστής μπορεί να πάρει πρόσβαση σε περισσότερη μνήμη από αυτή που στην πραγματικότητα είναι εγκατεστημένη στο σύστημα. Αυτό είναι δυνατό μέσω της μεταφοράς σελίδων της μνήμης στην εικονική μνήμη (swap) στο σκληρό δίσκο.

Επιπρόσθετα, τεχνικές εικονικοποίησης μπορεί να εφαρμοστούν σε όλα τα επίπεδα των υπολογιστικών υποδομών – δίκτυα, αποθήκευσης, λειτουργικά συστήματα και τις εφαρμογές που τρέχουν σε αυτά. Αυτό το μείγμα τεχνολογιών παρέχει ένα επίπεδο αφαίρεσης ανάμεσα στο υλικό που γίνεται η διαδικασία εκτέλεσης, αποθήκευσης, και οι διαδικασίες δικτύου και στις εφαρμογές που τρέχουν σε αυτό. Η εφαρμογή εικονικών υποδομών δεν προκαλεί καμία διαταραχή καθώς ο χρήστης δεν μπορεί να παρατηρήσει παρά μόνο πολύ μικρές μεταβολές στην εμπειρία του από το σύστημα.

Ωστόσο οι εικονικές υποδομές παρέχουν στους διαχειριστές του συστήματος τη δυνατότητα διαχείρισης των πόρων από ένα «κοινό ταμείο» (pool) σε όλη την έκταση της υποδομής, συνθήκη που επιτρέπει την γρήγορη ανταπόκριση σε δυναμικές αλλαγές και στην καλύτερη ανάπτυξη νέων υποδομών.[9]

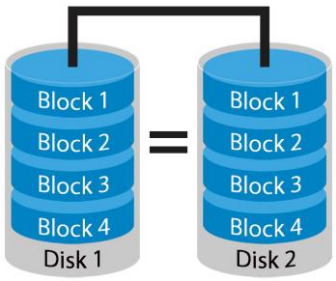
- Προσθέτουν ένα επίπεδο αφαιρετικότητας (abstraction) ανάμεσα στην εκάστοτε εφαρμογή και το υλικό
- Μειώνουν το κόστος και την πολυπλοκότητα.
- Προσφέρουν απομόνωση των πόρων των υπολογιστών για αυξημένη αξιοπιστία και ασφάλεια
- Βελτιώνουν τα επίπεδα υπηρεσιών και την ποιότητα της υπηρεσίας
- Συγχρονίζουν-ευθυγραμμίζουν τις IT διαδικασίες με τους επιχειρηματικούς στόχους
- Εξαλείφουν την αποδέσμευση και μεγιστοποιούν το ποσοστό χρήσης των υποδομών IT

2.1 Τύποι Εικονικοποίησης

Πολλοί θεωρούν την εικονικοποίηση σαν τον διαμερισμό απλά των υπολογιστικών πόρων, ενώ στην πραγματικότητα η εικονικοποίηση μπορεί να σημαίνει και ακριβώς το αντίθετο, δηλαδή την συνένωση πολλών οντοτήτων σε μια εικονική ενιαία, οδηγώντας έτσι στο “κρύψιμο” ή στη μεταμφίεση των πραγματικών υποδομών που υπάρχουν στην πραγματικότητα. Η εφαρμογή της εικονικοποίησης μπορεί να γίνει σε επίπεδο δικτύου, αποθηκευτικού χώρου, υπολογιστικών πόρων διακομιστή και εφαρμογών.

2.1.1 Εικονικοποίηση Αποθήκευσης

Με τον όρο εικονικοποίηση αποθήκευσης εννοείται η λογική παρουσίαση φυσικών πολλαπλών αποθηκευτικών συσκευών. Η τεχνολογία RAID άνοιξε νέες προοπτικές στην εικονικοποίηση της αποθήκευσης αλλά η διαχείριση εκατοντάδων gigabyte ακόμη και terabyte αποθηκευτικού χώρου αποδεικνύεται προβληματική. Η τεχνολογία RAID δίνει τη δυνατότητα να ρυθμίσουμε δύο ή παραπάνω φυσικούς δίσκους να εμφανίζονται ως ένας λογικός δίσκος. Για το λειτουργικό σύστημα, η παρουσίαση φυσικών δίσκων ως ένας λογικός δίσκος αποτελεί μία απλή λύση εικονικοιποιημένης αποθήκευσης. Στην Εικόνα 2.1, βλέπουμε το μοντέλο εικονικοιποιημένης αποθήκευσης RAID 1, όπου τα δεδομένα του δίσκου 1, αντιγράφονται στον δίσκο 2 όπου αναλαμβάνει ρόλο εφεδρικού δίσκου.



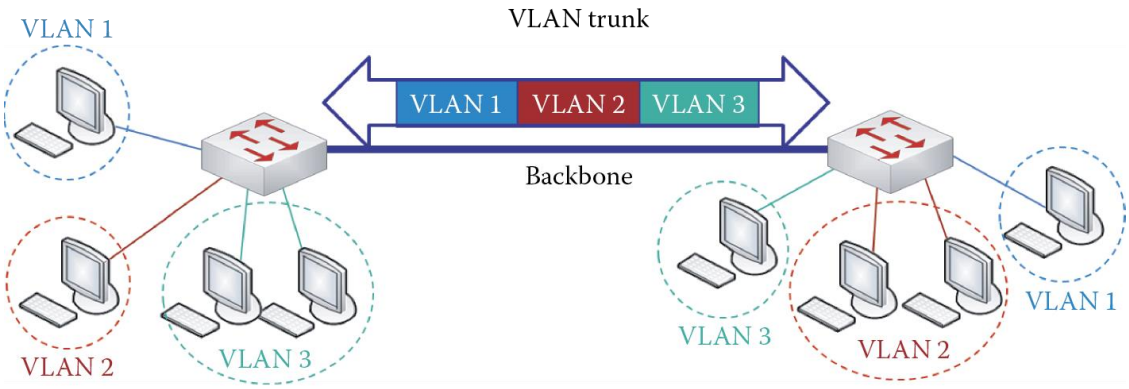
Εικόνα 2.1 - Εικονικοποίηση Αποθήκευσης (Πηγή www.flexraid.com/wp-content/uploads/2013/10/DataProtection_02_RAID11.jpg)

2.1.2 Εικονικοποίηση Δικτύου

Με τον όρο εικονικοποίηση δικτύου εννοούμε τον συνδυασμό πόρων του δικτύου υπολογιστών σε μία ενιαία πλατφόρμα, που είναι γνωστή με την ονομασία εικονικό δίκτυο (Virtual network). Αυτό επιτυγχάνεται με την βοήθεια λογισμικού και υπηρεσιών που επιτρέπουν την κοινή χρήση δικτυακών πόρων. Η τεχνολογία αυτή χρησιμοποιεί μία μέθοδο παρόμοια με τη διαδικασία εικονικοποίησης που χρησιμοποιείται για την προσομοίωση εικονικών μηχανών μέσα σε φυσικούς υπολογιστές. Ένα εικονικό δίκτυο θεωρεί όλο το υλικό και το λογισμικό στο δίκτυο ως μια ενιαία συλλογή πόρων, η οποία μπορεί να προσεγγιστεί ανεξάρτητα από τα φυσικά όρια του. Με άλλα λόγια η εικονικοποίηση δικτύου επιτρέπει σε κάθε εξουσιοδοτημένο χρήστη να μοιραστεί δικτυακούς πόρους από ένα μόνο υπολογιστή. Τα εικονικά δίκτυα χωρίζονται σε δύο μεγάλες κατηγορίες :

Εξωτερικά εικονικά δίκτυα

Τα εξωτερικά εικονικά δίκτυα αποτελούνται από ένα ή περισσότερα τοπικά δίκτυα τα οποία συνδυάζονται ή υποδιαιρούνται σε εικονικά δίκτυα, με στόχο τη βελτίωση της αποτελεσματικότητας ενός μεγαλύτερου δικτύου ή κέντρου δεδομένων. Τα βασικά στοιχεία ενός εξωτερικού εικονικού δικτύου, είναι τα εικονικά δίκτυα (Virtual Local Area Networks - VLANs) και οι μεταγωγής δικτύου (switches). Όπως φαίνεται στην Εικόνα 2.3, με την χρήση των VLAN και της τεχνολογίας μεταγωγέων, ο διαχειριστής, μπορεί να διαμορφώσει συστήματα τα οποία είναι συνδεδεμένα με φυσικό τρόπο στο ίδιο τοπικό δίκτυο, έτσι ώστε να ανήκουν σε διαφορετικά μεταξύ τους εικονικά δίκτυα.[9]



Εικόνα 2.2 - Εικονικοποίηση Δικτύου (Πηγή [9])

Εσωτερικά εικονικά δίκτυα

Η εσωτερική εικονικοποίηση δικτύου παρέχει ολόκληρο το σύστημα διαμοιρασμού καθώς και άλλες λειτουργίες του δικτύου στους περιέκτες (containers) λογισμικού, οι οποίοι λειτουργούν σε φιλόξενα περιβάλλοντα για τα στοιχεία του λογισμικού του δικτύου σε ένα μοναδικό φυσικό σύστημα.

2.1.3 Εικονικοποίηση Διακομιστών

Ένας από τους πιο σπουδαίους τύπους εικονικοποίησης με την μεγαλύτερη εφαρμογή, είναι η εικονικοποίηση διακομιστών. Οι οργανισμοί πληροφορικής επιδιώκουν με κάθε τρόπο την εικονικοποίηση των φυσικών διακομιστών τους προσπαθώντας με αυτό τον τρόπο να μειώσουν το κόστος ενέργειας-ψύξης, τον χώρο που απαιτείται για την τοποθέτησή τους καθώς και την καλύτερη διαχείριση του υλικού τους. Στην εικονικοποίηση διακομιστών ένας φυσικός διακομιστής μετατρέπεται σε έναν εικονικό.

Ο όρος φυσικός διακομιστής αναφέρεται στο υλικό που πραγματοποιεί την υπολογιστική επεξεργασία που επιβάλλεται από το λογισμικό, όπως το λειτουργικό σύστημα και οι εφαρμογές. Ένας εικονικός διακομιστής δεν μπορεί να λειτουργήσει χωρίς έναν φυσικό. Με την εικονικοποίηση διακομιστών μπορούμε πολλούς φυσικούς διακομιστές να τους μετατρέψουμε σε εικονικούς και να τους τοποθετήσουμε σε ένα φυσικό διακομιστή ο οποίος ονομάζεται και οικοδεσπότης (host). Αντίθετα, οι εικονικοί διακομιστές ονομάζονται φιλοξενούμενοι (guests). Στην Εικόνα 2.3, διακρίνεται η διαφορά της παραδοσιακής αρχιτεκτονικής διακομιστή από αυτή του εικονικού διακομιστή.[9]

Εικόνα 2.3 - Εικονικοποίηση Διακομιστών (Πηγή robertjrgraham.com/wp-content/uploads/2011/07/traditional_vs_virtual.jpg)

2.2 Επιμελητής Εικονικών Μηχανών (Virtual Machine Monitor – Hypervisor)

Ο Επιμελητής Εικονικών Μηχανών (ΕΜΜ) είναι ένα πρόγραμμα το οποίο διαχωρίζει με ασφάλεια τους πόρους ενός υπολογιστικού συστήματος σε μια ή περισσότερες εικονικές μηχανές. Ένα Φιλοξενούμενο Λειτουργικό Σύστημα (ΦΛΣ) εκτελείται υπό την επίβλεψη ενός ΕΕΜ αντί να εκτελείται απευθείας στους φυσικούς πόρους – υλικό του υπολογιστικού συστήματος. Ο ΕΕΜ εκτελείται στον πυρήνα (kernel) του συστήματος.

Οι ΕΕΜ επιτρέπουν πολλών τύπων λειτουργικά συστήματα (π.χ.Windows, Linux) να τρέχουν ταυτόχρονα στους ίδιους φυσικούς πόρους, προσφέροντας ανεξαρτησία και απομόνωση μεταξύ των λειτουργικών συστημάτων ενισχύοντας την ασφάλειά τους. Ο ΕΕΜ ελέγχει κατά πώς ένα ΦΛΣ χρησιμοποιεί τους πόρους και τα γεγονότα που συμβαίνουν σε μια εικονική μηχανή δεν επηρεάζουν τις άλλες. Ο ΕΕΜ επιτρέπει:

- Διαμοιρασμό πολλαπλών υπηρεσιών κάτω από την ίδια πλατφόρμα

- Την μεταφορά μια εικονική μηχανής από μία πλατφόρμα σε μία άλλη
- Συμβατότητα σε προηγούμενες εκδόσεις μία εικονικής μηχανής για λόγους συντήρησης

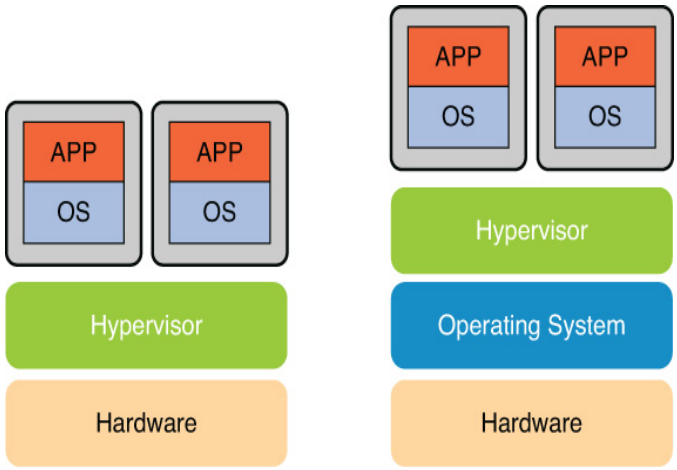
Όταν ένα ΦΛΣ εκτελεί κάποιες εντολές, ο ΕΕΜ εντοπίζει την εντολή αυτή και εγγυάται την σωστή εκτέλεσή της, ενώ παράλληλα επιβλέπει το σύστημα ως προς την απόδοσή του και εκτελεί διορθωτικές ενέργειες ώστε να αποφευχθεί η μείωση αυτής. Επιπρόσθετα, ο ΕΕΜ εικονικοποιεί την υπολογιστική ισχύ και την μνήμη μίας εικονικής μηχανής. Όπως φαίνεται στην Εικόνα 2.4 ένας ΕΕΜ μπορεί να υλοποιηθεί κυρίως σε δύο τύπους. Παρακάτω παρουσιάζονται οι τύποι και οι αρχιτεκτονικές τους:

ΕΕΜ Τύπου 1

Το λογισμικό εκτελείται απ' ευθείας στο υλικό, ενώ τα φιλοξενούμενα λειτουργικά τρέχουν σε ένα επίπεδο υψηλότερο από το πραγματικό υλικό επιτρέποντας έτσι πραγματική απομόνωση κάθε εικονικής μηχανής. Τα πιο γνωστά προϊόντα ΕΕΜ του τύπου 1 είναι το VMWare, το Xen και το Hyper-V της Microsoft.

ΕΕΜ Τύπου 2

Το λογισμικό τρέχει μέσα στο λειτουργικό σύστημα. Αν και αυτή η προσέγγιση προσφέρει κάποια πλεονεκτήματα, συνήθως υπόκειται σε περιορισμούς απόδοσης επειδή οι εντολές προς το φυσικό υλικό πρέπει να μετακινηθούν ανάμεσα σε πολλά διαφορετικά επίπεδα μέχρι να υλοποιηθούν και να επιστρέψουν τελικά στο ΦΛΣ. Τα πιο γνωστά λογισμικά ΕΕΜ του τύπου 2 είναι τα KVM. Qemu, VMWare, Virtual Box και το Hyper-V της Microsoft [9].



Εικόνα 2.4 – Αρχιτεκτονική των δύο τύπων ΕΕΜ (Πηγή ptgmedia.pearsoncmg.com/images/chap1_9780133570182/elementLinks/01fig02_alt.jpg

2.3 Ενορχηστρωτής (Orchestrator)

Σαν γενική έννοια, ο ενορχηστρωτής περιγράφει την αυτοματοποιημένη οργάνωση, συντονισμό και διαχείριση περίπλοκων υπολογιστικών συστημάτων και υπηρεσιών. Για το περιβάλλον εικονικοποίησης, ο ενορχηστρωτής διαχειρίζεται την δημιουργία και εγκατάσταση μιας εικονικής μηχανής για την ανάπτυξη εφαρμογών. Ο χρήστης καθορίζει τους υπολογιστικούς πόρους για κάθε εφαρμογή και ο ενορχηστρωτής αυτοματοποιεί την διαχείριση της εικονικής μηχανής. Ο ενορχηστρωτής βρίσκεται ένα λογικό επίπεδο πάνω από τον ΕΕΜ. Μπορεί να ορίσει και να δημιουργήσει μία εικονική μηχανή. Παρ'όλα αυτά, θα πρέπει να επικοινωνήσει με τον ΕΕΜ όταν μια εικονική μηχανή πρέπει να ξεκινήσει ή να σταματήσει. Παρακάτω παρουσιάζονται μερικά χαρακτηριστικά στοιχεία:

- Ένας ενορχηστρωτής πάντα χρησιμοποιεί της υπηρεσίες που προσφέρει ο ΕΕΜ. Με άλλα λόγια ο ενορχηστρωτής είναι το «μυαλό» και ο ΕΕΜ το «σώμα».

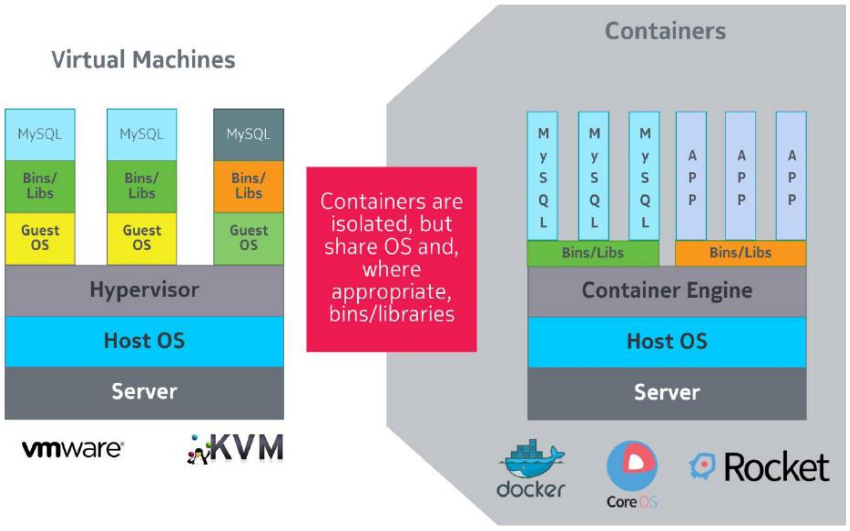
- Είναι απαραίτητος μόνο σε περίπλοκα περιβάλλοντα. Για απλές χρήσεις χρησιμοποιείται μονό ο EEM

Η ενορχήστρωση είναι μια ιδέα που είναι κατανοητή στο γενικό πλαίσιο της Υπολογιστικής Νέφους, της Διαχείρισης Νέφους ή όταν μία γενική εφαρμογή αποτελείται από πολλαπλές εικονικές μηχανές που απαιτούν μια κοινή διαχείριση.[11]

2.4 Εικονικοποίηση με Περιέκτες (Container-Based Virtualization)

Υπάρχει και μια νέα μορφή εικονικοποίησης που γνωρίζει άνθιση τον τελευταίο καιρό. Πρόκειται για την εικονικοποίηση με περιέκτες, όπου δεν υπάρχει η έννοια του φιλοξενούμενου λειτουργικού συστήματος. Οι εφαρμογές μπορεί να οργανωθούν σε περιέκτες. Αυτό σημαίνει ότι ένας περιέκτης θα περιέχει μία η περισσότερες εφαρμογές και αυτές θα γίνονται διαθέσιμες κάθε φορά που εκτελείται. Η εκτέλεσή τους μπορεί να γίνει σε διαφορετικά λειτουργικά συστήματα και διακομιστές. Στην Εικόνα 2.5 βλέπουμε την διαφορά αρχιτεκτονικής μεταξύ της εικονικοποίησης διακομιστών και εικονικοποίησης με περιέκτες.

Η εικονικοποίηση με περιέκτες αποτελεί μια πιο απλή και «ελαφριά» μορφή εικονικοποίησης καθώς δεν χρησιμοποιεί το ήδη υπάρχον λειτουργικό σύστημα, μπορεί να εκτελεσθεί γρήγορα, καθώς και να μεταφερθεί εύκολα μεταξύ των συστημάτων. Παρ' όλα αυτά, η εικονικοποίηση με περιέκτες έχει και αρνητικές πτυχές. Μια από αυτές είναι ότι η χρήση τους περιορίζεται κυρίες σε διαδικτυακές εφαρμογές και υπηρεσίες. Παράλληλα, η ασφάλεια είναι «αποδυναμωμένη» καθώς όλοι οι περιέκτες μοιράζονται το ίδιο λειτουργικό σύστημα. Τα πιο γνωστά συστήματα περιεκτών είναι το Docker και CoreOS Rocket.[11]



Εικόνα 2.5 - Αρχιτεκτονική Περιεκτών (Πηγή [11])

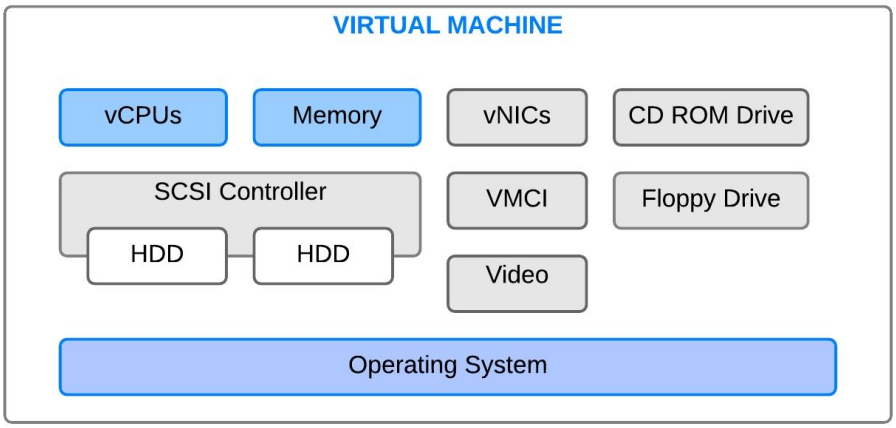
2.5 Εικονικές Μηχανές

Μία εικονική μηχανή έχει πολλά κοινά χαρακτηριστικά με ένα φυσικό υπολογιστή. Υποστηρίζει την φιλοξενία λειτουργικού συστήματος και χρησιμοποιεί ένα μέρος από τους διαθέσιμους υπολογιστικούς πόρους ώστε να εκτελέσει τις όποιες εφαρμογές. Σε αντίθεση με ένα φυσικό υπολογιστή, πολλές εικονικές μηχανές μπορούν να τρέξουν παράλληλα, εκτελώντας διαφορετικά λειτουργικά συστήματα και εφαρμογές.

Τα κύρια στοιχεία τα οποία συντεθούν μια εικονική μηχανή, είναι τα αρχεία ρυθμίσεων (configuration files) και τα αρχεία εικονικών δίσκων (virtual disk files). Τα αρχεία ρυθμίσεων περιγράφουν τους πόρους που μια εικονική μηχανή θα χρησιμοποιήσει κατά την αρχικοποίηση. Παραδείγματος χάρη, αν σκεφτούμε ότι η εικονική μηχανή είναι μία μητρική κάρτα υπολογιστή, τότε τα αρχεία ρυθμίσεων καταγράφουν την κεντρική μονάδα επεξεργασίας (CPU), το μέγεθος

της μνήμης, το μέγεθος του αποθηκευτικού χώρου, το δίκτυο και τον οπτικό δίσκο που θα συνδέονταν πάνω στην μητρική κάρτα.

Οι εικονικές μηχανές στην ουσία έχουν πρόσβαση σε ποικίλους πόρους του υλικού, αλλά από την δική τους οπτική γωνία, δεν γνωρίζουν ότι αυτοί οι πόροι υφίστανται (Εικόνα 2.6). Η εικόνα που προκύπτει είναι σαν να πρόκειται για φυσικούς πόρους. Από την πλευρά του λειτουργικού συστήματος και των εφαρμογών που εκτελούνται σε αυτό, η μνήμη, ο χώρος αποθήκευσης και η υπολογιστική ισχύ είναι όλα διαθέσιμα προς χρήση.[10]



Εικόνα 2.6 - Αρχιτεκτονική Υλικού Εικονικής Μηχανής (Πηγή [10])

Ξεχωρίζουμε δύο τύπους εικονικών μηχανών: συστήματος και διεργασίας. Η εικονική μηχανή επεξεργασίας είναι μια πλατφόρμα η οποία δημιουργείται με σκοπό μόνο να τρέξει συγκεκριμένες διεργασίες. Μετά το πέρας της εκτέλεσης η εικονική μηχανή καταστρέφεται. Από την άλλη πλευρά η εικονική μηχανή συστήματος, υποστηρίζει ένα λειτουργικό σύστημα μαζί με πολλές διεργασίες. Ο συγκεκριμένος τύπος παρέχει ένα σύστημα το οποίο είναι ανεξάρτητο από την πλατφόρμα που εκτελείται.[6]

Συνοψίζοντας με τα παραπάνω, παρατηρείται ότι τα πλεονεκτήματα των εικονικών μηχανών είναι:

Συμβατότητα

Οι εικονικές μηχανές είναι συμβατές με όλους τους τυποποιημένους x86 υπολογιστές, λειτουργικά συστήματα, εφαρμογές και οδηγούς συσκευών.

Απομόνωση

Ενώ οι εικονικές μηχανές μπορούν να μοιραστούν τους φυσικούς πόρους ενός ενιαίου υπολογιστή, παραμένουν εντελώς απομονωμένες η μία από την άλλη σαν να ήταν χωριστές φυσικές μηχανές. Η απομόνωση είναι ένας σημαντικός λόγος για τον οποίο η διαθεσιμότητα και η ασφάλεια των εφαρμογών που τρέχουν σε ένα εικονικό περιβάλλον είναι σαφώς ανώτερες από τις εφαρμογές που τρέχουν σε ένα παραδοσιακό σύστημα.

Φορητότητα

Μπορούμε να μετακινήσουμε και να αντιγράψουμε μια εικονική μηχανή από μια θέση σε μια άλλη ακριβώς όπως οποιαδήποτε άλλο αρχείο λογισμικού ή να σώσουμε μια εικονική μηχανή σε οποιοδήποτε τυποποιημένο μέσο αποθήκευσης.

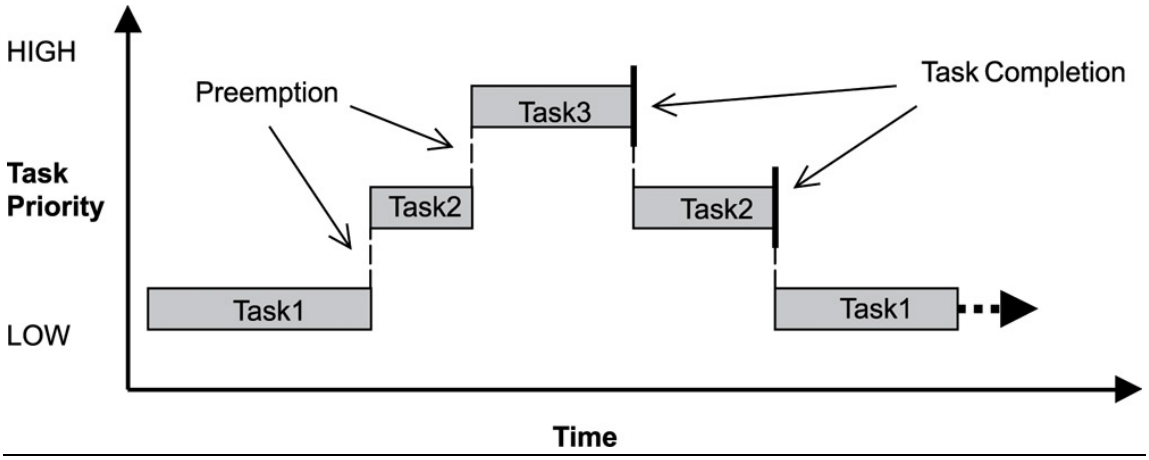
Ανεξαρτητοποιημένο Υλικό

Οι εικονικές μηχανές είναι απολύτως ανεξάρτητες από το φυσικό υλικό τους. Όταν δε αυτό συνδυάζεται με τις ιδιότητες της φορητότητας και της συμβατότητας, η ανεξαρτησία υλικού δίνει την δυνατότητα της μεταφοράς μιας εικονικής μηχανής από έναν τύπο x86 υπολογιστή προς άλλο χωρίς καμία απολύτως αλλαγή στους οδηγούς συσκευών, το λειτουργικό σύστημα, ή τις εφαρμογές. Η ανεξαρτησία υλικού επίσης σημαίνει ότι μπορούμε να τρέξουμε ένα ετερογενές μίγμα λειτουργικών συστημάτων και εφαρμογών σε έναν ενιαίο φυσικό υπολογιστή.[6]

3 Χρονοπρογραμματισμός

Η έννοια του χρονοπρογραμματισμού έχει αντίκτυπο σε πολλές πτυχές της καθημερινότητας. Από την προσωπική οργάνωση του χρόνου, μέχρι και την οργάνωση των εργασιών μια επιχείρησης. Σαν γενική έννοια μπορούμε να πούμε ότι ο χρονοπρογραμματισμός είναι η διαδικασία ανάθεσης ενός συνόλου από εργασίες στη διάρκεια του χρόνου, ώστε να βελτιστοποιείται κάποιο κριτήριο ικανοποιώντας ένα σύνολο από περιορισμούς. Η ικανοποίηση των περιορισμών έχει να κάνει με την ανάθεση κατάλληλων τιμών σε ένα σύνολο από μεταβλητές ώστε να επιτυγχάνεται ένας στόχος. Στόχοι του χρονοπρογραμματισμού είναι η ελαχιστοποίηση του χρόνου υλοποίησης, η ελαχιστοποίηση του χρόνου ή του κόστους των πόρων και η μεγιστοποίηση της χρησιμοποίησης των πόρων και όλα αυτά στο λιγότερο δυνατόν κόστος.

Στον κλάδο της πληροφορικής, ο χρονοπρογραμματισμός είναι η ανάθεση διαφόρων διεργασιών σε ένα σύνολο πόρων όπως υπολογιστική ισχύ, μνήμη, αποθήκευση, όπου οι πόροι αυτοί μπορούν να εκτελέσουν τις διεργασίες αυτές εκπληρώνοντας οποιεσδήποτε από τις απαιτήσεις. Υπάρχουν πολλοί παράγοντες που μπορούν να επηρεάσουν τις αποφάσεις του χρονοπρογραμματισμού όπως ο χρόνος καθυστέρησης ανάθεσης μιας διεργασίας προς εκτέλεση, η αργή μετάδοση των δεδομένων καθώς και τα πιθανά ποσοστά λάθους. Για τον λόγο αυτό οι διεργασίες θα πρέπει να οργανώνονται κατά προτεραιότητα (υψηλής ή χαμηλής σημασίας) σύμφωνα με τα κριτήρια που έχουν ορισθεί.[6] Στην Εικόνα 3.1 βλέπουμε ένα παράδειγμα όπου η διεργασία 3, έχοντας την υψηλότερη προτεραιότητα, παρεμβάλλεται στην εκτέλεση της διεργασίας 2. Αντίστοιχα κατά την εκτέλεση της διεργασίας 1, παρεμβάλλεται η διεργασία 2, καθώς έχει υψηλότερη προτεραιότητα.



Εικόνα 3.1 - Προτεραιότητα διεργασιών (Πηγή http://www.embeddedlinux.org.cn/rtconforembsys/5107final/images/other-0405_0.jpg)

Όσο ο φόρτος εργασίας σε ένα υπολογιστικό σύστημα αλλάζει, η απαίτηση για πόρους από τις εφαρμογές ποικίλει με αποτέλεσμα κάποιες από τις διεργασίες να χρειάζονται επαναπρογραμματισμό. Επιπρόσθετα, νέες διεργασίες μπορούν να γίνουν διαθέσιμες προς εκτέλεση, πράγμα το οποίο απαιτεί νέες αποφάσεις χρονοπρογραμματισμού.

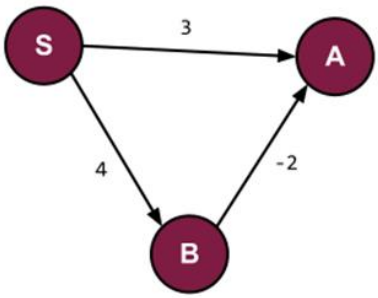
3.1 Το πρόβλημα του χρονοπρογραμματισμού

Η υπολογιστική νέφους, μπορούμε να πούμε ότι είναι ένα εικονικό καλάθι από πόρους οι οποίοι διατίθενται στους χρήστες μέσω του διαδικτύου. Όπως προαναφέρθηκε στα προηγούμενα κεφάλαια, επιτρέπει στους χρήστες να χρησιμοποιούν τους πόρους αυτούς και να χρεώνονται ανάλογα με την ζήτηση. Αντίθετα, οι πάροχοι έχουν ως στόχο να μεγιστοποιήσουν τα κέρδη τους μέσω αυτών των υπηρεσιών. Αυτό έχει ως αποτέλεσμα να απαιτείται χρονοπρογραμματισμός διεργασιών, βοηθώντας τους παρόχους σε καλύτερη διαχείριση και εξοικονόμηση των πόρων αυτών. Ο χρονοπρογραμματισμός αυτών των διεργασιών αποτελεί πρόκληση για τους παρόχους καθώς και για τον κλάδο της πληροφορικής γενικότερα.

Το πρόβλημα του χρονοπρογραμματισμού έγκειται στο πρόβλημα της ένωσης διαφόρων στοιχείων από διαφορετικά σύνολα, το οποίο εκφράζεται και ως (**Π, Λ, Σ**) όπου:

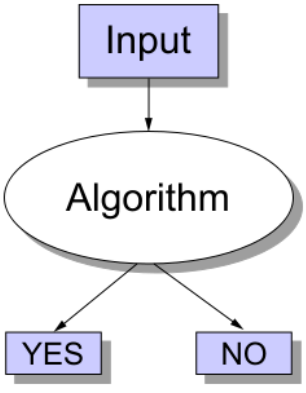
- **Π** είναι ένα σύνολο από παραδείγματα, όπου κάθε παράδειγμα αποτελεί και ένα πρόβλημα.
- **Λ** είναι ένα σύνολο από πιθανές λύσεις για το πρόβλημα.
- **Α** είναι το αντικείμενο του προβλήματος.

Μπορούν να προκύψουν δύο κατηγορίες προβλημάτων χρονοπρογραμματισμού, το πρόβλημα απόφασης και το πρόβλημα βελτιστοποίησης. Το πρόβλημα βελτιστοποίησης απαιτεί την αναζήτηση της καλύτερης λύσης ανάμεσα σε ένα σύνολο πιθανών λύσεων του Λ. Για παράδειγμα οι αλγόριθμοι δρομολόγησης προσπαθούν να επιλύσουν προβλήματα βελτιστοποίησης, βρίσκοντας την «καλύτερη» διαδρομή για ένα πακέτο δεδομένων, βάσει συγκεκριμένων κριτηρίων απόφασης (πχ. Bellman Ford).[14]



Εικόνα 3.2 - Δρομολόγηση μεταξύ κόμβων δικτύου

Από την άλλη πλευρά, ο στόχος του προβλήματος απόφασης είναι πιο απλός. Για μία πιθανή λύση λ η οποία ανήκει στο σύνολο λύσεων Λ, το πρόβλημα απαιτεί μία αρνητική ή θετική απάντηση για το αν το αντικείμενο του προβλήματος Α μπορεί να επιλυθεί. Είναι ξεκάθαρο ότι το πρόβλημα βελτιστοποίησης είναι ευκολότερο, καθώς, μια λύση συγκρίνεται από ένα συγκεκριμένο κριτήριο, σε αντίθεση με το πρόβλημα απόφασης όπου πρέπει να συγκριθούν όλες οι λύσεις.



Εικόνα 3.3 - Διάγραμμα προβλήματος απόφασης (Πηγή https://upload.wikimedia.org/wikipedia/commons/0/06/Decision_Problem.svg)

Ένας αλγόριθμος είναι ένα σύνολο απλών εντολών που στοχεύουν στην λύση του προβλήματος. Αποτελείται από τρία μέρη: την είσοδο, την μέθοδο και την έξοδο. Η είσοδος είναι ένα σύνολο από δεδομένα που πρόκειται να χρησιμοποιηθούν, η μέθοδος είναι ένα σύνολο

ελεγχόμενων και επαναλαμβανόμενων διαδικασιών που στοχεύουν στην λύση χρησιμοποιώντας δεδομένα εισόδου, ενώ η έξοδος είναι το αποτέλεσμα που προκύπτει. Ιδιαίτερα για τον χρονοπρογραμματισμό, ο αλγόριθμος αποτελεί την μέθοδο όπου οι διεργασίες αποκτούν πρόσβαση, αντιστοιχίζονται και ανατίθενται στους υπολογιστικούς πόρους. Στην πραγματικότητα δεν υπάρχει κάποιος «τέλειος» αλγόριθμος καθώς οι τελικοί στόχοι διαφέρουν ανά περίπτωση.[17]

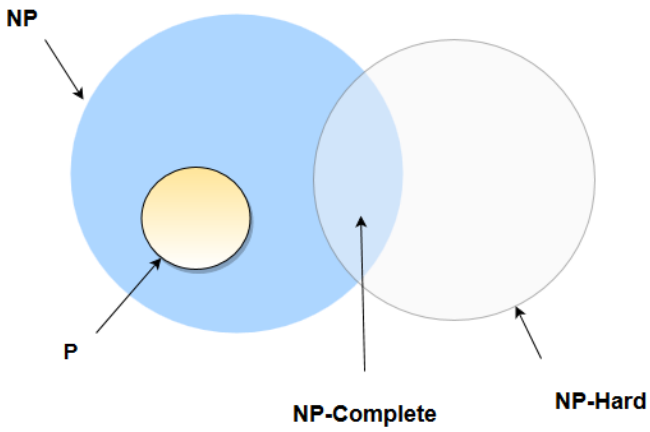
Ένα πρόβλημα μπορεί να επιλυθεί σε δευτερόλεπτα, ώρες ακόμα και χρόνια. Η διάρκεια εξαρτάται από τον αλγόριθμο που εφαρμόζεται. Ένας αλγόριθμος θεωρείται αποδοτικός από την ποσότητα του χρόνου που απαιτείται για να εκτελεσθεί. Ο χρόνος εκτέλεσης ενός αλγόριθμου ορίζεται ως η συνάρτηση της πολυπλοκότητας του χρόνου σε σχέση με τον αριθμό των διαδικασιών που εκτελούνται. Υπάρχουν διάφοροι αλγόριθμοι πολυπλοκότητας:

- Συνεχής αλγόριθμος A(1), όπου ο μέγιστος χρόνος εκτέλεσης ορίζεται από μια τιμή η οποία δεν εξαρτάται από το μέγεθος εισόδου
- Γραμμικός αλγόριθμος A(n), όπου ο μέγιστος χρόνος εκτέλεσης αυξάνεται γραμμικά με το μέγεθος εισόδου
- Πολυωνυμικός αλγόριθμος $A(n^c)$, όπου ο μέγιστος χρόνος εκτέλεσης εξαρτάται από την πολυωνυμική συνάρτηση του μεγέθους εξόδου
- Εκθετικός αλγόριθμος $A(2^{nc})$, όπου ο μέγιστος χρόνος εκτέλεσης εξαρτάται από την εκθετική συνάρτηση του μεγέθους εξόδου

Εάν ένα πρόβλημα απαιτεί αλγόριθμο πολυωνυμικού χρόνου, τότε είναι προσιτό, εφικτό και αποδοτικό να εκτελεστεί σε μια υπολογιστική μηχανή. Στην θεωρία της πολυπλοκότητας, τα σύνολα των προβλημάτων μπορεί να διαχωριστούν σε κλάσεις:

- Κλάση P, είναι ένα σύνολο από προβλήματα αποφάσεων τα οποία επιλύονται σε πολυωνυμικό χρόνο από μία ντετερμινιστική μηχανή Turing, και η λύση του προβλήματος μπορεί να αποφασιστεί γρήγορα από έναν πολυωνυμικό αλγόριθμο.
- Κλάση NP, είναι ένα σύνολο από προβλήματα αποφάσεων τα οποία επιλύονται σε πολυωνυμικό χρόνο από μία μη- ντετερμινιστική μηχανή Turing, και η πιθανή λύση του προβλήματος μπορεί να επαληθευτεί γρήγορα από έναν πολυωνυμικό αλγόριθμο.
- Κλάση NP-πλήρης, είναι ένα σύνολο από προβλήματα αποφάσεων, όπου όλα τα προβλήματα μπορούν να μετασχηματιστούν σε προβλήματα κλάσης NP. Αν για ένα οποιοδήποτε NP-πλήρες πρόβλημα βρεθεί πολυωνυμικός αλγόριθμος τότε P=NP. Τα NP-πλήρη προβλήματα είναι πιο δύσκολα σε σύγκριση με τα NP προβλήματα.
- Κλάση NP-δύσκολη, είναι ένα σύνολο από προβλήματα βελτιστοποίησης, όπου όλα τα προβλήματα μπορούν να μετασχηματιστούν αλλά ένα NP-δύσκολο πρόβλημα δεν είναι απαραίτητο ότι θα ανήκει στο σύνολο της κλάσης NP.

Στην Εικόνα 3.4 βλέπουμε ένα διάγραμμα της περιοχή που καλύπτει η κάθε κλάση.



Εικόνα 3.4 - Κλάσεις προβλημάτων σε διάγραμμα Venn

Αν και τα περισσότερα προβλήματα κλάσης NP-πλήρης είναι δύσκολα υπολογιστικά, κάποια από αυτά μπορούν να επιλυθούν αποδοτικά σε επαρκείς χρόνους. Υπάρχουν αλγόριθμοι όπου ο χρόνος εκτέλεσής τους δεν εξαρτάται μόνο από το μέγεθος εισόδου αλλά και από το μέγιστο αριθμό των εισόδων. Αυτοί οι αλγόριθμοι έχουν ψευδο-πολυωνυμική πολυπλοκότητα χρόνου. Για ένα πρόβλημα, αν τα μεγέθη εισόδου δεν είναι πολλά σε αριθμό, μπορούν να επιλυθούν γρήγορα. Αυτό που προκύπτει είναι ότι όταν P≠NP, ένα NP-πλήρες πρόβλημα αν δεν μπορεί να λυθεί με ψευδο-πολυωνυμικούς αλγορίθμους αποτελεί ένα NP-δύσκολο πρόβλημα.[15]

3.2 Τύποι χρονοπρογραμματισμού

Ο βέλτιστος χρονοπρογραμματισμός διεργασιών στην υπολογιστική νέφους θα πρέπει να αποφασίζει τον σωστό αριθμό πόρων που απαιτούνται ώστε να μειώνεται το κόστος και να μην παραβιάζονται οι συμφωνίες πελάτη-παρόχου. Η υπολογιστική νέφους είναι δυναμική, πράγμα που κάνει την κατανομή πόρων να επιδέχεται συνεχώς βελτιστοποιήσεις. Για τον λόγο αυτό, έχουν δημιουργηθεί διάφοροι τύποι αλγορίθμων.[16]

Χρονοπρογραμματισμός υπηρεσιών νέφους

Ο χρονοπρογραμματισμός νέφους κατηγοριοποιείται σε δύο επίπεδα, στο επίπεδο χρήστη και στο επίπεδο συστήματος. Στο επίπεδο χρήστη, ο χρονοπρογραμματισμός χρησιμοποιείται σε προβλήματα που έχουν να κάνουν μεταξύ πελατών και παρόχου. Αναλαμβάνει την ανάθεση πόρων μεταξύ των πελατών, με βάση τις συμφωνίες πελάτη-παρόχου. Στο επίπεδο συστήματος ο χρονοπρογραμματισμός χειρίζεται θέματα διαχείρισης πόρων και ανάθεσης διεργασιών εντός του κέντρου δεδομένων.

Στατικός και δυναμικός χρονοπρογραμματισμός

Ο στατικός χρονοπρογραμματισμός «προσκομίζει» τα απαιτούμενα δεδομένα ως μια συστάδα και τα θέτει προς εκτέλεση, γνωρίζοντας εξαρχής τους χρόνους εκτέλεσης της κάθε διεργασίας και έχοντας προαποφασίσει τους πόρους ανάθεσης. Ο στατικός χρονοπρογραμματισμός μειώνει τον φόρτο εκτέλεσης. Αντίθετα, στον δυναμικό προγραμματισμό ο χρόνος εκτέλεσης δεν είναι προϋπολογισμένος και η ανάθεση των διεργασιών στους πόρους γίνεται εκείνη την στιγμή.

Ευρετικός χρονοπρογραμματισμός

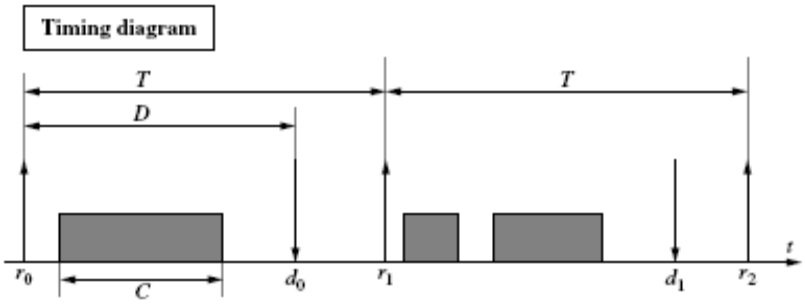
Τα προβλήματα βελτιστοποίησης είναι της κλάσης NP-δύσκολη, και μπορούν να επιλυθούν με την απαριθμητική μέθοδο, την ευρετική μέθοδο ή την μέθοδο προσέγγισης. Στην απαριθμητική μέθοδο, η καλύτερη λύση θα επιλεγεί εάν όλες οι πιθανές λύσεις αριθμηθούν και συγκριθούν μία προς μία. Η συγκεκριμένη μέθοδος παρουσιάζει αστάθεια όταν το μέγεθος των λύσεων είναι μεγάλο. Σε αυτή την περίπτωση χρησιμοποιείται η ευρετική μέθοδος ώστε να βρει καλές λύσει σε ένα σχετικά μικρό χρονικό διάστημα. Η προσεγγιστική μέθοδος χρησιμοποιείται ώστε να βρεί μια λύση η οποία είναι εγγυημένα «κοντά» στη βέλτιστη λύση.

Χρονοπρογραμματισμός πραγματικού χρόνου

Στα συστήματα πραγματικού χρόνου, το πρώτιστο μέλημα δεν είναι η υψηλή ακρίβεια ή η υψηλή πιστότητα, αλλά η αύξηση της μετάδοσης δεδομένων καθώς και η βελτιστοποίηση του χρόνου εκτέλεσης. Οι διεργασίες πραγματικού χρόνου είναι οι βασικές οντότητες οι οποίες χρονοδρομολογούνται. Οι διεργασίες μπορεί να είναι περιοδικές ή σποραδικές. Ένα μοντέλο διεργασίας προσδιορίζεται με χρονικά χαρακτηριστικά που περιλαμβάνουν βασικές και δυναμικές παραμέτρους. Οι παράμετροι της διεργασίας είναι (Εικόνα 3.5):

- **r**, χρόνος άφιξης της διεργασίας που προσδιορίζεται από τη χρονική στιγμή που γίνεται αίτηση για την εκτέλεσή της.
- **C**, χρόνος εκτέλεσης της διεργασίας όταν ο επεξεργαστής εξυπηρετεί μόνον αυτή.

- **D**, σχετική χρονική προθεσμία εκτέλεσης της διεργασίας που προσδιορίζει τη μέγιστη επιτρεπτή καθυστέρηση ολοκλήρωσής της.
- **T**, περίοδος της διεργασίας μόνο για περιοδικές διεργασίες.
- **s**, χρόνος έναρξης της εκτέλεσης της διεργασίας.
- **e**, διάρκεια χρόνου της εκτέλεσης της διεργασίας.
- **D(t) = d-t**, χρόνος που υπολείπεται από το χρονικό σημείο t που βρίσκεται η διεργασία μέχρι το χρόνο πέρατος της σχετικής της προθεσμίας. Ισχύει: 0 < D(t) < D.
- **C(t)**, χρόνος που απομένει από το χρονικό σημείο που βρίσκεται η διεργασία μέχρι το χρονικό σημείο περάτωσής της. Ισχύει: 0 < C(t) < C.

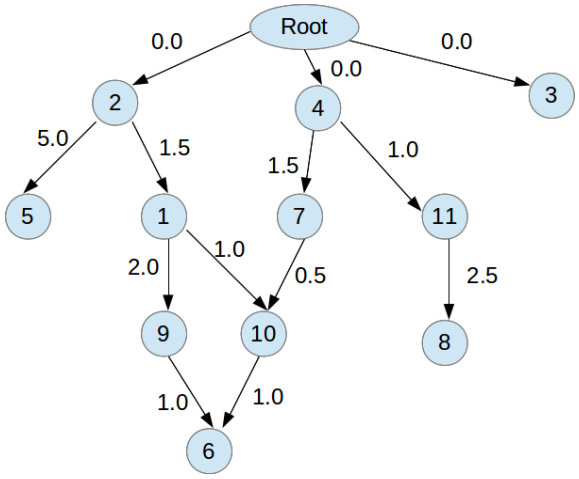


Εικόνα 3.5 - Διάγραμμα παραμέτρων διεργασίας (Πηγή [30])

Κάθε φορά που μία διεργασία είναι έτοιμη, κάνει μια αίτηση εκτέλεσης. Οι διαδοχικοί χρόνοι που απευθύνει αυτήν την αίτηση είναι rk = r0 + kT, όπου r0 είναι ο πρώτος χρόνος άφιξης. Οι διαδοχικοί απόλυτοι χρόνοι θα είναι dk = rk +D. Αν ισχύει D=T, η περιοδική διεργασία έχει σχετική χρονική προθεσμία εκτέλεσης ίση με την περίοδό της. Μια διεργασία είναι καλά δομημένη εάν ισχύει 0<C<D<T. Η ποιότητα της χρονοδρομολόγησης εξαρτάται από την ακρίβεια των παραπάνω παραμέτρων και έτσι ο λεπτομερής καθορισμός τους είναι πολύ σημαντικό ζήτημα για το σχεδιασμό των συστημάτων πραγματικού χρόνου.

Χρονοπρογραμματισμός ροής εργασίας

Ο χρονοπρογραμματισμός ροής εργασίας είναι από τα σημαντικά στοιχεία της διαχείρισης εκτέλεσης ροής εργασιών και θα μας απασχολήσει αρκετά στην συνέχεια της διπλωματικής εργασίας. Στην Εικόνα 3.6 βλέπουμε ότι η ροής εργασίας δομεί τις διεργασίες σε μορφή άκυκλου κατευθυνόμενου γράφου (Directed Acyclic Graph - DAG), όπου κάθε κόμβος αναπαριστά μια διεργασία και τα άκρα διασύνδεσης αναπαριστούν την εξάρτηση μεταξύ δύο κόμβων. Η ροή εργασίας αποτελείται από ένα σύνολο διεργασιών όπου επικοινωνούν-εξαρτώνται με άλλους κόμβους του γράφου.

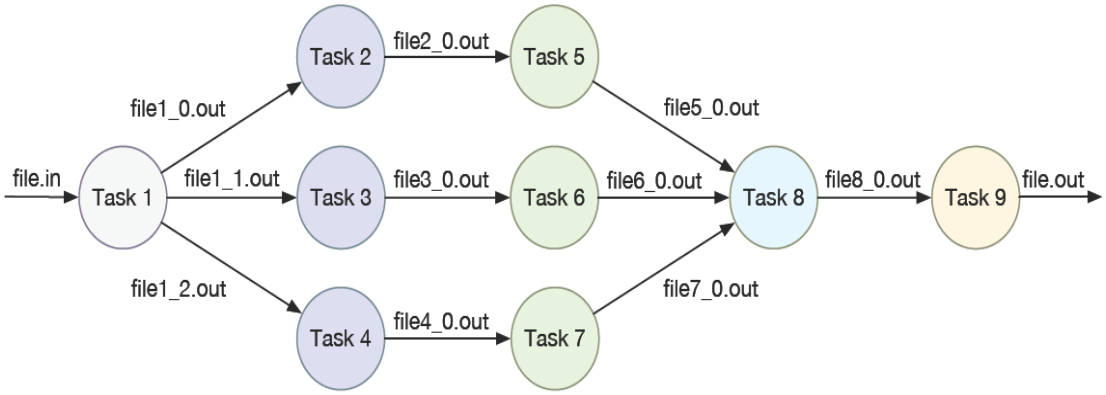


Εικόνα 3.6 - Η αναπαράσταση μιας ροής εργασίας σε γράφο (Πηγή https://i.stack.imgur.com/s1kwR.png)

3.3 Ροές Εργασίας (Scientific Workflows)

Η ιδέα των ροών εργασίας έχει τις ρίζες της από τις εμπορικές επιχειρήσεις, σαν ένα εργαλείο μοντελοποίησης των διεργασιών της εταιρείας. Αυτές οι επιχειρηματικές ροές εργασιών, οι οποίες πρωτοεμφανίστηκαν το 1999, στοχεύουν στην αυτοματοποίηση και βελτιστοποίηση των εργασιών. Όπως αναφέρθηκε στην προηγούμενη ενότητα, μια ροή εργασίας αποτελείται από ένα σύνολο διεργασιών όπου εξαρτώνται μεταξύ τους. Στις επιστημονικές εφαρμογές, οι εξαρτήσεις αυτές συχνά μεταφράζονται σαν ροή δεδομένων από την μία διεργασία στην άλλη· τα παραγόμενα δεδομένα γίνονται δεδομένα εισόδου για την επόμενη.

Η Εικόνα 3.7, παρουσιάζει ένα παράδειγμα μιας ροής δεδομένων με εννέα διεργασίες. Η διεργασία 1 παράγει δεδομένα εισόδου προς τις διεργασίες 2,3 και 4. Τα δεδομένα αυτά αποτελούν την εξάρτηση μεταξύ της διεργασίας 1 και των 2,3 και 4. Οι διεργασίες αυτές δεν θα ξεκινήσουν αν δεν ληφθούν τα δεδομένα από την διεργασία 1. Αντίστοιχη λογική ακολουθείται και για τις υπόλοιπες διεργασίες.



Εικόνα 3.7 - Ροή εργασίας με δεδομένα εισόδου-εξόδου

Οι διεργασίες, μπορεί να απαιτούν την χρήση υπολογιστικής ισχύος, την χρήση μνήμης, την χρήση του καναλιού εισόδου/εξόδου (I/O) ή και των τριών μαζι, ανάλογα την φύση του προβλήματος που προσπαθούν να επιλύσουν. Στις ροές υπολογιστικής ισχύος, οι περισσότερες διεργασίες εκτελούν υπολογισμούς. Στις ροές μνήμης οι διεργασίες απαιτούν την

χρήση μεγάλου μεγέθους της φυσικής μνήμης του συστήματος. Τέλος στις ροές εισόδου/εξόδου, οι διεργασίες απαιτούν και παράγουν μεγάλου μεγέθους δεδομένα.

Πολλές επιστημονικές περιοχές προτιμούν τις ροές εργασιών για την παρουσίαση πολύπλοκων υπολογιστικών προβλημάτων όπου μπορούν να επιλυθούν στα κατανεμημένα συστήματα. Για παράδειγμα μια συγκεκριμένη ροή εργασίας με το όνομα Montage, οπού θα μας απασχολήσει στην συνέχεια, χρησιμοποιείται στην αστρονομίας και βοηθά τους επιστήμονες στην δημιουργία σύνθετης εικόνας μιας περιοχής του νυχτερινού ουρανού η οποία είναι δύσκολο να παραχθεί από αστρονομικές κάμερες.

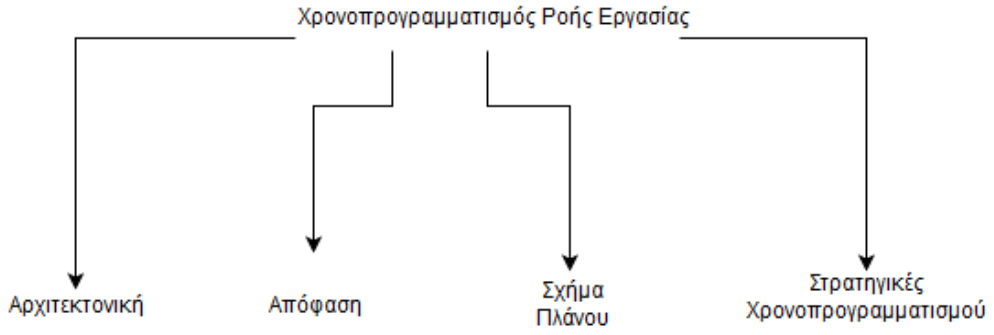
Ο σκοπός της μεταπτυχιακής διατριβής περιορίζεται σε ροές εργασιών οι οποίες αναπαριστώνται ως Άκυκλοι Κατευθυνόμενοι Γράφοι. Τα πιο γνωστά συστήματα διαχείρισης ροής εργασιών είναι τα Pegasus, Cloudbus WfMS, ASKALON και DAGMan. Ο γράφος παρουσιάζει μια ροη εργασιών W=(T,E) όπου W είναι η ροή εργασίας, T είναι το σύνολο των διεργασιών $T=t_1,t_2,t_3…t_n$ και E είναι το σύνολο των συνδέσεων $E=e_1,e_2,e_3…e_n$ μεταξύ των διεργασιών. Μια σύνδεση e_{ij}, δημιουργείται όταν υπάρχει εξάρτηση δεδομένων μεταξύ δύο διεργασιών t_i με t_j.[18]

3.4 Χρονοπρογραμματισμός Ροών Εργασίας

Η ανάθεση και η διαχείριση των διεργασιών μιας ροής εργασίας πραγματοποιείται με την βοήθεια του χρονοπρογραμματισμού αυτής της ροής, καθώς βρίσκει την σωστή ακολουθία εκτέλεσης των διεργασιών με σκοπό να ικανοποιηθούν οι απαιτήσεις και περιορισμοί που υπάρχουν από την συμφωνία πελάτη-παρόχου. Τα στοιχεία του χρονοπρογραμματισμού ροής εργασίας (Εικόνα 3.8) είναι τα έξης:

Αρχιτεκτονική: Είναι απαραίτητη η δημιουργία μιας δομής η οποία θα βελτιώνει την απόδοση και την ποιότητα του συστήματος. Οι αποφάσεις για την ανάθεση των διεργασιών λαμβάνονται από ένα κεντρικό χρονοπρογραμματιστή. Υπάρχουν περιπτώσεις όπου ο φόρτος διαμοιράζεται σε περισσότερους από έναν χρονοπρογραμματιστές υπό την προϋπόθεση ότι υπάρχει επικοινωνία μεταξύ τους.

Απόφαση: Είναι οι αποφάσεις που λαμβάνονται από τον χρονοπρογραμματιστή για την οργάνωση και ανάθεση των διεργασιών ως προς εκτέλεση.



Εικόνα 3.8 - Στοιχεία χρονοπρογραμματισμού

Σχήμα Πλάνου: Υπάρχουν δύο σχήματα για το πλάνο των διεργασιών, το στατικό και το δυναμικό. Στην περίπτωση του στατικού πλάνου, η αντιστοίχιση διεργασίας και πόρων γίνεται πριν την έναρξη εκτέλεσης της ροής εργασίας. Τα κριτήρια για την αντιστοίχιση επηρεάζονται την απόδοση καθώς και της απαιτήσεις του χρήστη. Υπάρχουν περιπτώσεις όπου πριν την έναρξη της ροής εργασίας πραγματοποιείται προσομοίωση της εκτέλεσης των διεργασιών στους πόρους. Στο δυναμικό πλάνο εφαρμόζεται η τεχνική της πρόβλεψης σύμφωνα με την οποία οι αντιστοίχιση γίνεται σύμφωνα με την προβλεπόμενη απόδοση των διεργασιών. Έτσι υπάρχει πιθανότητα πριν την εκτέλεση μιας διεργασίας το πλάνο να αλλάξει «μορφή».

Η στρατηγική αγοράς στοχεύει στην διαθεσιμότητα των πόρων, στη μείωση του κόστους χρησιμοποίησης των πόρων καθώς υπό την προϋπόθεση πάντα ότι οι στόχοι (deadlines) θα επιτυγχάνουν. Με άλλα λόγια η στρατηγική αυτή χρησιμοποιείται στο χρονοπρογραμματισμό μιας ροής εργασίας με στον καλύτερα δυνατό πόρο όπου θα προσφέρει χαμηλότερο συνολικό κόστος.

Η στρατηγική εμπιστοσύνης στοχεύει στην ασφάλεια. Αυτό σημαίνει ότι οι διεργασίες αντιστοιχούνται σε πόρους οπού προσφέρουν ασφάλεια ως προς το ποσοστό επιτυχίας εκτέλεσης μειώνοντας την πιθανότητα λαθών.

4 Αλγόριθμοι Χρονοπρογραμματισμού

Η συγκεκριμένη μεταπτυχιακή διατριβή στοχεύει στην παρουσίαση και την ανάλυση διαφόρων αλγορίθμων που χρησιμοποιούνται σε περιβάλλον Υπολογιστικού Νέφους και στοχεύουν στην βέλτιστη απόδοση του συστήματος. Οι αλγόριθμοι αυτοί χρησιμοποιούνται από τον χρονοπρογραμματιστή και εφαρμόζονται σε μια ροή εργασίας. Σκοπός της χρησιμοποίησης του αλγορίθμου είναι η μείωση του χρόνου εκτέλεσης σύμφωνα πάντα με τις απαιτήσεις του συστήματος.

Αρχικά οι αλγόριθμοι χρονοπρογραμματισμού υλοποιούνταν και εφαρμόζονταν σε περιβάλλον πλέγματος. Λόγω της μειωμένης απόδοσής τους, άρχισαν να χρησιμοποιούνται και σε περιβάλλοντα υπολογιστικού νέφους. Το πρωταρχικό πλεονέκτημα είναι ότι η επεκτασιμότητα που προσφέρεται συμβάλλει σε καλύτερη ποιότητα υπηρεσιών. Στις μέρες μας, αναπτύσσονται πολλοί αλγόριθμοι χρονοπρογραμματισμού οι οποίοι στοχεύουν στην βελτιστοποίηση συγκεκριμένων χαρακτηριστικών, όπως η ποιότητα υπηρεσιών (Quality of Service), ο χρόνος εκτέλεσης, η αξιοπιστία εκτέλεσης, η απόδοση, η κατανομή των πόρων καθώς και το κέρδος.

4.1 First Come First Served (FCFS)

Αποτελεί τον απλούστερο τρόπο χρονοπρογραμματισμού. Σε αυτόν τον αλγόριθμο οι διεργασίες συγκρίνονται σύμφωνα με τον χρόνο άφιξής τους, και η εργασία που φθάνει πρώτη στην ουρά εκτέλεσης εξυπηρετείται. Η μεθοδολογία που ακολουθείται είναι η First In First Out (FIFO). Το πλεονέκτημα του αλγορίθμου αυτού είναι απλότητα και η ταχύτητα εκτέλεσης καθώς δεν προσθέτει επιπλέον υπολογιστικό φόρτο. [19]

Το κύριο μειονέκτημά του έχει να κάνει με την εκτέλεση διεργασιών όπου απαιτούν μεγάλο χρόνο εκτέλεσης, καθώς η ουρά είναι μοναδική και όταν εκτελείται μια τέτοια διεργασία, οι διεργασίες που απαιτούν λιγότερο χρόνο εκτέλεσης χρειάζεται να παραμένουν στην ουρά περιμένοντας να εξυπηρετηθούν. Παράλληλα ο μέσος χρόνος αναμονής προς εκτέλεση εξαρτάται από την σειρά εκτέλεσης των διεργασιών. Λόγω αυτού του προβλήματος αυξάνεται ο συνολικός χρόνος εκτέλεσης ενώ παράλληλα μειώνεται η απόδοση του συστήματος.

Πίνακας 2. Δεδομένα FCFS

Διεργασία	**Διάρκεια Εκτέλεσης**	**Σειρά**	**Χρόνος Άφιξης**
Δ1	*24*	*1*	*0*
Δ2	*3*	*2*	*0*
Δ3	*4*	*3*	*0*

Παράδειγμα υπολογισμού:

χρόνος αναμονής Δ1: 0 μέσος χρόνος αναμονής: (0+24+27)/3 = **17**

χρόνος αναμονής Δ2: 24

χρόνος αναμονής Δ3: 27

Πίνακας 3. Δεδομένα FCFS με διαφορετική σειρά άφιξης

Διεργασία	**Διάρκεια Εκτέλεσης**	**Σειρά**	**Χρόνος Άφιξης**
Δ1	*24*	*3*	*0*
Δ2	*3*	*2*	*0*
Δ3	*4*	*1*	*0*

Παράδειγμα υπολογισμού:

χρόνος αναμονής Δ3: 0 μέσος χρόνος αναμονής: (0+4+7)/3 = **3,6**

χρόνος αναμονής Δ2: 4

χρόνος αναμονής Δ1: 7

4.2 Min-Min

Ο συγκεκριμένος αλγόριθμος είναι σχετικά απλός στην λογική του και αποτελεί έναν από τους βασικούς αλγορίθμους για υπολογιστικό νέφος ακόμα και σήμερα. Αρχικά ορίζει ένα σύνολο Σ από όλες τις διεργασίες που δεν έχουν αντιστοιχηθεί σε κάποιο πόρο προς εκτέλεση. Στην συνέχεια βρίσκεται ο πόρος Π ο οποίος μπορεί να προσφέρει τον μικρότερο χρόνο ολοκλήρωσης (ΜΧΟ) για όλες τις διεργασίες. Στην συνέχεια η διεργασία Δ με το μικρότερο μέγεθος επιλέγεται και ανατίθεται στον αντίστοιχο πόρο Π. Στην συνέχεια η διεργασία Δ αφαιρείται από το σύνολο και η παραπάνω διαδικασία επαναλαμβάνεται μέχρι να ανατεθούν όλες οι διεργασίες του συνόλου Σ.

Είναι ένας γρήγορος αλγόριθμος, ο οποίος μπορεί να βελτιώσει την απόδοση και να μειώσει το συνολικό χρόνο ολοκλήρωσης (makespan). Η ανάθεση όμως τον διεργασιών με τον μικρότερο χρόνο είναι και το μειονέκτημά του καθώς θα χρησιμοποιούνται συνήθως οι «καλύτεροι» πόροι, το οποίο έχει σαν αποτέλεσμα την μη κατανεμημένη διαχείρισή τους.[20]

1. **for** all tasks T_i in meta-task M_v
2. **for** all resources R_j
3. $C_{ij}=E_{ij}+r_j$
4. **do** until all tasks in M_v are mapped
5. for each task in M_v find the earliest completion time and the resource that obtines it
6. find the task T_k with the *minimum* earliest completion time
7. assigne task T_k to the resource R_l that gives the earliest completion time
8. delete task T_k from M_v
9. update r_l
10. update C_{il} for all i
11. **end do**

Εικόνας 4.1 - Αλγόριθμος Min-Min (Πηγή [20])

Ο ψευδοκώδικας της Εικόνας 4.1 υποθέτει ότι υπάρχει ένα σύνολο διεργασιών T1,T2,T3…Ti, όπου πρέπει να ανατεθούν στους διαθέσιμους πόρους P1,P2,P3…Pj. Ο Αναμενόμενος Χρόνος Ολοκλήρωσης (ΑΧΟ) για την διεργασία i στον πόρο j, ορίζεται ως Ctij και υπολογίζεται ως Ctij = Etij + rtj , όπου rj είναι ο χρόνος που χρειάζεται ο πόρος Rj να είναι έτοιμος για εκτέλεση και Etij είναι ο εκτιμώμενος χρόνος εκτέλεσης της διεργασίας Ti στον πόρο Rj. Έστω οι διεργασίες (Πίνακας 4) με ΑΧΟ στους πόρους T1,T2,T3

Πίνακας 4. ΑΧΟ για τις διεργασίες ABC στους πόρους T1,T2,T3

	T1	**T2**	**T3**
A	**3**	6	7
B	4	12	8
C	6	20	18

Από τις παραπάνω διεργασίες επιλέγεται πρώτη ως προς εκτέλεση η Α καθώς προσφέρει τον μικρότερο χρόνο ολοκλήρωσης, στο πόρο Τ1, με αποτέλεσμα ο Τ1 να γίνεται διαθέσιμος έπειτα από τον απαιτούμενο χρόνο εκτέλεσης της Α.

Πίνακας 5. ΑΧΟ για τις διεργασίες ΒC στους πόρους Τ1,Τ2,Τ3

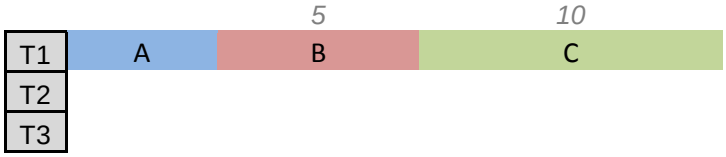
	T1	**T2**	**T3**
B	4+*3*=***7***	12	8
C	6+*3*=*9*	20	18

Στο επόμενο στάδιο η διεργασία Α αφαιρείται και ο πίνακας διεργασιών ανανεώνεται με τις νέες τιμές ΑΧΟ. Παρατηρούμε ότι πάλι ο πόρος Τ1 προσφέρει τον μικρότερο χρόνο ολοκλήρωσης. Έτσι επιλέγεται η διεργασία Β να εκτελεστεί στον πόρο Τ1 και επαναλαμβάνεται η ίδια διαδικασία για την διεργασία C, όπου και σε αυτή την περίπτωση επιλέγεται πάλι ο πόρος Τ1.

Πίνακας 6. ΑΧΟ για την διεργασία C στους πόρους Τ1,Τ2,Τ3

	T1	**T2**	**T3**
C	9+*4*=***13***	20	18

Η χρήση των πόρων παρουσιάζεται στο παρακάτω διάγραμμα:



4.3 Max-Min

Ο αλγόριθμος Max-Min, οπού παρουσιάζεται στην Εικόνα 4.2, ακολουθεί παρόμοια μεθοδολογία με τον Min-Min αλλά με μία βασική διαφορά. Οι διεργασίες με το μεγαλύτερο μέγεθος έχουν την προτεραιότητα εκτέλεσης αντί των μικρών, με σκοπό οι μικρές διεργασίες να καθυστερούν για αρκετό χρονικό διάστημα. Συγκεκριμένα, υπολογίζει τον εκτιμώμενο χρόνο εκτέλεσης και τον εκτιμώμενο χρόνο ολοκλήρωσης της κάθε διεργασίας σε κάθε ένα από τους διαθέσιμους πόρους.

Στην συνέχεια επιλέγει την διεργασία με τον μέγιστο εκτιμώμενο χρόνο εκτέλεσης και την αναθέτει στον πόρο με τον μικρότερο χρόνο εκτέλεσης. Στην συνέχεια η διεργασία αφαιρείται από το σύνολο και η διαδικασία επαναλαμβάνεται μέχρι να ανατεθούν όλες οι διεργασίες του συνόλου.Ο αλγόριθμος είναι αποδοτικότερος όταν οι διεργασίες με μικρό χρόνο εκτέλεσης είναι περισσότερες από αυτές με μεγάλο χρόνο εκτέλεσης. Για παράδειγμα, όταν υπάρχει μια μόνο μεγάλη διεργασία, τότε εκτελούνται ταυτόχρονα πολλές μικρές.[21]

1. **for** all tasks T_i in meta-task M_v
2. **for** all resources R_j
3. $C_{ij}=E_{ij}+r_j$
4. **do** until all tasks in M_v are mapped
5. for each task in M_v find the earliest completion time and the resource that obtines it
6. find the task T_k with the maximum earliest completion time
7. assigne task T_k to the resource R_l that gives the earliest completion time
8. delete task T_k from M_v
9. update r_l
10. update C_{il} for all i
11. **end do**

Εικόνα 4.2 - Αλγόριθμος Max-Min (Πηγή [20])

.

	T1	**T2**	**T3**
A	3	6	7
B	4	12	8
C	**6**	20	18

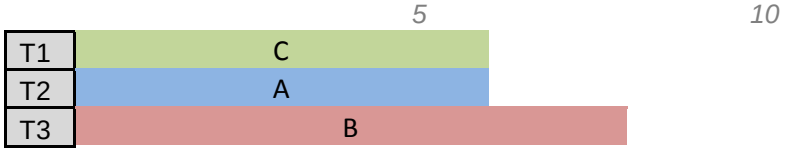
Χρησιμοποιώντας τον Πίνακα 4, παρατηρούμε ότι αρχικά αναζητείται η διεργασία με τον μέγιστο ΑΧΟ. Η διεργασία C με τιμή 20 επιλέγεται προς εκτέλεση και ανατίθεται στον πόρο T1, καθώς προσφέρει τον μικρότερο ΑΧΟ.

	T1	**T2**	**T3**
A	3+6= 9	6	7
B	4+6=10	12	**8**

Στο επόμενο στάδιο επιλέγεται η διεργασία B καθώς έχει τον μέγιστο ΑΧΟ και ανατίθεται στον πόρο T3. Παρόμοια, η διαδικασία A ανατίθεται στον πόρο T2.

	T1	**T2**	**T3**
A	9	**6**	7+8=15

Η χρήση των πόρων παρουσιάζεται στο παρακάτω διάγραμμα:



4.4 Minimum Completion Time (MCT)

Ο αλγόριθμος MCT, είναι μέρος των παραπάνω αλγορίθμων. Συγκεκριμένα, αντιστοιχεί κάθε διεργασία σε τυχαία σειρά στον πόρο με το μικρότερο χρόνο ολοκλήρωσης. Αυτό έχει σαν αποτέλεσμα κάποιες από τις διεργασίες να αντιστοιχούνται σε πόρους που δεν έχουν το ελάχιστο χρόνο εκτέλεσης για αυτές. Η συμπεριφορά αυτή μας θυμίζει την περίπτωση του Min-Min με την διαφορά ότι ο Min-Min κατά την διάρκεια της αντιστοίχησης εξετάζει όλες τις διεργασίες του συνόλου.[21]

4.5 RASA

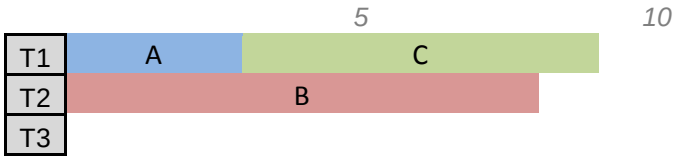
Ο συγκεκριμένος αλγόριθμος που παρουσιάζεται στην Εικόνα 4.3, έχει υλοποιηθεί με βάση τους προαναφερθέντες αλγορίθμους Min-Min και Max-Min. Ο στόχος του είναι να εκμεταλλευτεί τα πλεονεκτήματά τους και να «κρύψει» τα μειονεκτήματα τους. Για να το επιτύχει αρχικά χτίζει ένα πίνακα C όπου Cij είναι ο χρόνος ολοκλήρωσης της διεργασίας Ti στον πόρο Rj. Εάν ο αριθμός των διαθέσιμων πόρων είναι περιττός τότε εφαρμόζεται ο αλγόριθμος Min-Min, ενώ αν είναι άρτιος εφαρμόζεται ο Max-Min.Η διαδικασία αυτή ακολουθείται για τις υπόλοιπες διεργασίες.

Για παράδειγμα, αν η μια διεργασία αντιστοιχιστεί σε ένα πόρο με τον αλγόριθμο Min-Min τότε η επόμενη θα αντιστοιχιστεί με τον αλγόριθμο Max-Min. Έτσι ο αλγόριθμος RASA χρησιμοποιεί τον Min-Min για την εκτέλεση των μικρών διεργασιών, ενώ τον Max-Min για την εκτέλεση των μεγάλων διεργασιών, με αποτέλεσμα την αποφυγή καθυστερήσεων από αυτές. Τα πλεονεκτήματα του αλγορίθμου είναι η βελτίωση του χρόνου ολοκλήρωσης σε σύγκριση με τους Min-Min και Max-Min. Παράλληλα βελτιώνει την ροή δεδομένων και προσφέρει καλύτερη κατανομή πόρων. Ο αλγόριθμος είναι αποδοτικότερος όταν ο όγκος δεδομένων είναι μεγάλος.[21]

```
for all tasks Ti in meta-task Mv
    for all resources Rj
        Cij=Ej+rj
do until all tasks in Mv are mapped
    if the number of resources is even then
        for each task in Mv find the earliest completion
            time and the resource that obtines it
        find the task Tk with the maximum earliest
            completion time
        assigne task Tk to the resource Rl that gives the
            earliest completion time
        delete task Tk from Mv
        update rl
        update Cil for all i
    else
        for each task in Mv find the earliest completion
            time and the resource that obtines it
        find the task Tk with the minimum earliest
            completion time
        assigne task Tk to the resource Rl that gives the
            earliest completion time
        delete task Tk from Mv
        update rl
        update Cil for all i
    end if
end do
```

Εικόνα 4.3 - Αλγόριθμος RASA (Πηγή [21])

Για το παράδειγμά μας, χρησιμοποιούμε τα δεδομένα του Πίνακα 4. Εφόσον ο αριθμός διεργασιών είναι περιττός, αρχικά θα εφαρμοσθεί ο αλγόριθμος Min-Min και στην συνέχεια θα πραγματοποιηθεί εναλλαγή αλγορίθμων από Max-Min σε Min-Min.



4.6 Heterogeneous Earlier Finish Timer (HEFT)

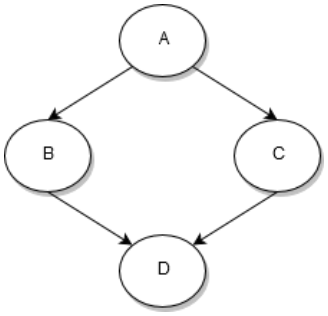
Ο αλγόριθμος HEFT, σε αντίθεση με αυτούς που παρουσιάστηκαν προηγουμένως, αποτελεί ένα δυναμικό αλγόριθμο που χρησιμοποιείται στα ετερογενή υπολογιστικά συστήματα και εφαρμόζεται κυρίως σε ροές εργασιών. Με την ανάθεση μιας τιμής-βάρους σε κάθε κόμβο (διεργασία) και με τη διασύνδεση του γράφου δίνεται προτεραιότητα σε αυτούς ικανοποιούν καλύτερα την συνθήκη που προκύπτει από τα δύο παραπάνω βάρη. Οι φάσεις που αποτελούν τον συγκεκριμένο αλγόριθμο είναι τρεις:

1. Η ανάθεση «βάρους» στους κόμβους και τις διασυνδέσεις της ροής εργασίας.
2. Οι διεργασίες ταξινομούνται σε σειρά κατάταξης σύμφωνα με την σειρά εκτέλεσης τους
3. Αντιστοίχιση των διεργασιών στους πόρους.

Τα βάρη που τίθεντο στους κόμβους υπολογίζονται σύμφωνα με τον εκτιμώμενο χρόνο εκτέλεσης, ενώ τα βάρη που τίθεντο στις διασυνδέσεις υπολογίζονται σύμφωνα με τον εκτιμώμενο ρυθμό μεταφοράς των δεδομένων από τον ένα κόμβο στον επόμενο. Στα ομοιογενή περιβάλλοντα τα βάρη έχουν την ίδια τιμή. Αντίθετα, στα ετερογενή περιβάλλοντα, τα βάρη θα πρέπει προσεγγιστούν λαμβάνοντας υπόψη διαφορετικούς χρόνους εκτέλεσης σε διαφορετικούς πόρους και για διαφορετικούς ρυθμούς δεδομένων σε διαφορετικές διασυνδέσεις.[22]

Η διαδικασία δημιουργίας της σειράς κατάταξης πραγματοποιείται διαβαίνοντας την ροή εργασίας **από το τέλος προς την αρχή** και θέτοντας μια τιμή κατάταξης σε κάθε κόμβο. Αρχικά πρέπει να υπολογισθούν τα βάρη για κάθε κόμβο και διασύνδεση. Όπως αναφέρθηκε και προηγουμένως, οι διεργασίες εκφράζονται ως κόμβοι. Στην περίπτωση που υπάρχουν περισσότεροι του ενός πόρου, η τιμή του βάρους του κόμβου υπολογίζεται ως ο μέσος όρος του χρόνο εκτέλεσης μιας διεργασίας σε όλους τους διαθέσιμους πόρους. Αντιστοίχως τα βάρη της διασύνδεσης υπολογίζονται από το μέσω ρυθμό μετάδοσης των δεδομένων από τον ένα κόμβο προς τον επόμενο διασυνδεδεμένο. Έτσι η τιμή κατάταξης ισούται με το τελικό βάρος του κόμβου συν το χρόνο εκτέλεσης των διαδόχων τους, συν το βάρος του ρυθμού μετάδοσης της διασύνδεσης.

Τέλος, δημιουργείται η σειρά κατάταξης και η εκτέλεση πραγματοποιείται από την αρχή προς το τέλος του κόμβου και η προτεραιότητα δύνεται στον κόμβο με την μεγαλύτερη τιμή κατάταξης, όποτε η τιμές είναι από την μεγαλύτερη προς την μικρότερη. Κατά την ανάθεση του επιλεγόμενου κόμβου η διεργασία εκτελείται στον πόρο που προσφέρει τον ελάχιστο χρόνο εκτέλεσης. Έστω η ροή εργασίας της Εικόνας 4.4:



Εικόνα 4.4 - Ροή εργασίας 1 με τέσσερις κόμβους

Για τη ροή αυτή ισχύουν οι παρακάτω τιμές:

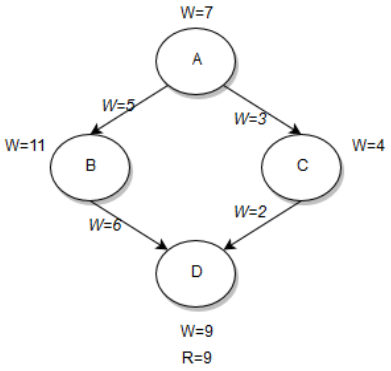
Πίνακας 7. Ρυθμός δεδομένων για την ροή εργασίας 1

	T1 -> T2	**T1 ->T3**	**T2->T3**	**M.O.**
A->B	6	4	5	*5*
A->C	4	2	3	*3*
B->D	7	4	7	*6*
C->D	1	1	4	*2*

Πίνακας 8. Ρυθμός χρόνου εκτέλεσης διεργασιών για την ροή εργασίας 1

	T1	**T2**	**T3**	**M.O**
A	5	8	8	7
B	13	9	11	11
C	3	4	5	4
D	10	10	7	9

Σύμφωνα με τον αλγόριθμο αρχικά θα πρέπει να υπολογισθεί ο μέσος όρος του βάρους για κάθε κόμβο και διασύνδεση. Η διαδικασία δημιουργίας της σειράς κατάταξης ξεκινάει από τον τελευταίο κόμβο. Έστω R η τιμή κατάταξης και W η τιμή βάρους, τότε για τον τελευταίο κόμβο D ισχύει ότι $R_D = W_D$. Στην Εικόνα 4.5 βλέπουμε τα βάρη για κάθε κόμβο και διασύνδεση.



Εικόνα 4.5 - Βάρη της ροής εργασίας 1

Στην συνέχεια ακολουθώντας τον κανόνα του αλγορίθμου υπολογίζουμε τις αντίστοιχες τιμές κατάταξης, ως εξής:

$R_C = R_D + W_{CD} + W_C =$ **15**

$R_B = R_D + W_{BD} + W_B =$ **26**

$R_A = R_B + W_{AB} + W_A =$ **38**

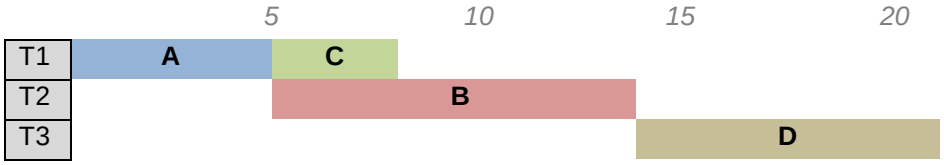
$R_A = R_C + W_{AC} + W_A = 25$

Για την περίπτωση του κόμβου A , επειδή υπάρχουν δύο μονοπάτια προς αυτόν, θα χρησιμοποιηθεί αυτό που δίνει την μεγαλύτερη τιμή κατάταξης. Έτσι η σειρά κατάταξης έχει ως εξής:

Πίνακας 9. Συγκεντρωτικός Πίνακας Κατάταξης ροής εργασίας 1

	T1	**T2**	**T3**	**Κατάταξη**
A	**5**	8	8	*38*
B	13	**9**	11	*26*
C	**3**	4	5	*15*
D	10	10	**7**	*9*

Η χρήση των πόρων παρουσιάζεται στο παρακάτω διάγραμμα:



Τα πλεονεκτήματα του αλγορίθμου HEFT είναι η ταχύτητα εκτέλεσης και η απλή δομή του σε σύγκριση με άλλους δυναμικούς αλγορίθμους.

4.7 Αλγόριθμος Triplet

Ο συγκεκριμένος αλγόριθμος εφαρμόζεται σε ετερογενή συστήματα και έχει σαν στόχο να βελτιστοποιήσει τον χρόνο εκτέλεσης μειώνοντας τον χρόνο επικοινωνίας μεταξύ των διεργασιών. Αυτό επιτυγχάνεται μέσο της ομαδοποίησης, και εφαρμόζεται σε ροές δεδομένων. Η βασική ιδέα του αλγορίθμου είναι ότι οι διεργασίες και οι πόροι του συστήματος ομαδοποιούνται σε συστάδες όπου μια συστάδα διεργασιών ανατίθεται σε μία συστάδα πόρων. Οι διεργασίες ομαδοποιούνται με βάση την διασύνδεσή τους με σκοπό να μειωθεί ο φόρτος επικοινωνίας, ενώ οι πόροι ομαδοποιούνται με βάση τα κοινά τους χαρακτηριστικά όπως το δίκτυο ή την υπολογιστική ισχύ. Οι φάσεις που αποτελούν τον συγκεκριμένο αλγόριθμο είναι τρεις:

1. Ομαδοποίηση διεργασιών

Κάθε κόμβος-διεργασία, αρχικά, αναπαριστάται και ως συστάδα, η οποία αποτελείται από ένα κόμβο. Για να μειωθεί ο χρόνος επικοινωνίας πραγματοποιείται συγχώνευση των κόμβων. Οι κόμβοι πού ανήκουν σε μονοπάτι μεγέθους δύο, όπου το μονοπάτι σχηματίζεται από τον επιλεγμένο κόμβο και δύο συνδεδεμένους κόμβους μετά, ομαδοποιούνται σε μια νέα συστάδα με σύνολο τριών κόμβων η οποία ονομάζεται triplet. Όταν πραγματοποιηθεί η συγκεκριμένη ομαδοποίηση τότε οι συστάδες ταξινομούνται με πρώτο κριτήριο τον αριθμό των εντολών προς εκτέλεση και δεύτερο τον ρυθμό μετάδοσης δεδομένων.

2. Ομαδοποίηση πόρων

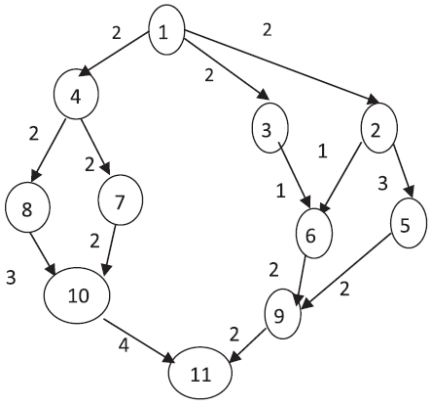
Οι πόροι όπου έχουν τα ίδια χαρακτηριστικά ομαδοποιούνται σε συστάδες ώστε να επιτευχθεί ομοιογένεια στο σύστημα. Στην περίπτωση που κάποιοι από τους πόρους δεν έχουν τα ίδια χαρακτηριστικά τότε ο καθένας σχηματίζει μια υποθετική συστάδα.

3. Αντιστοίχιση

Εδώ γίνεται η αντιστοίχιση των διεργασιών στους πόρους. Οι συστάδες των πόρων και των διεργασιών ταξινομούνται βάσει των ελαχίστων ρυθμών μετάδοσης και της μέγιστης υπολογιστικής ισχύος. Η συστάδα διεργασιών αντιστοιχείται με σκοπό το σύστημα μας να έχει καλύτερο χρόνο ολοκλήρωσης με προϋπόθεση να μην υπερβαίνει το όριο του ρυθμού μετάδοσης. Αυτή η αντιστοίχηση διασφαλίζει ότι οι διεργασίες που απαιτούν μεγάλο ρυθμό

μετάδοσης εξυπηρετούνται από τις καλύτερες διασυνδέσεις και κάθε συστάδα αυτών ολοκληρώνεται σχεδόν στον ίδιο χρόνο.[23]

Στην Εικόνα 4.6 βλέπουμε μια ροή εργασίας έντεκα κόμβων και τεσσάρων πόρων, στην οποία εφαρμόζουμε τον αλγόριθμο Triplet.



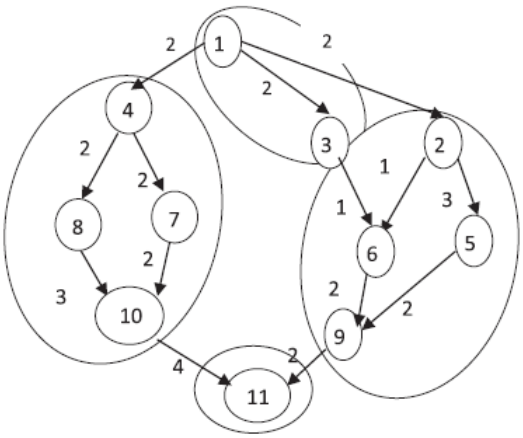
Εικόνα 4.6 - Ροή εργασίας 2 για τον αλγόριθμο Triplet (Πηγή [23])

Έστω ότι έχουμε τις τιμές υπολογιστικού κόστους κάθε διεργασίας. Με την μεθοδολογία του HEFT αλγορίθμου υπολογίζουμε τις τιμές κατάταξης, όπως εμφανίζονται στον πίνακα που ακολουθεί.

Πίνακας 10. Συγκεντρωτικός Πίνακας Κατάταξης ροής εργασίας 2

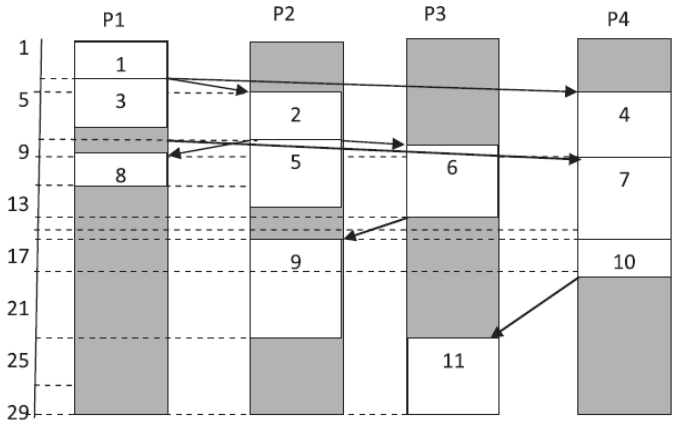
Διεργασία	**T1**	**T2**	**T3**	**T4**	**M.O**	**Κατάταξη**
1	4	4	4	4	*4*	*34*
2	5	5	5	5	*5*	*28*
3	4	6	4	7	*5.25*	*26.5*
4	3	3	3	3	*3*	*27.5*
5	3	5	3	4	*3.75*	*20*
6	3	7	2	2	*3.5*	*20.25*
7	5	8	5	5	*5,75*	*22.5*
8	2	4	5	3	*3.5*	*21.25*
9	5	6	7	5	*5.75*	*14.25*
10	3	7	5	2	*4.25*	*14.75*
11	5	6	7	8	*6.5*	*6.5*

Το επόμενο βήμα είναι η δημιουργία των συστάδων. Όπως είπαμε προηγουμένως, οι συστάδες δημιουργούνται με σκοπό την μείωση του χρόνου επικοινωνίας μεταξύ των διεργασιών. Η ομαδοποίηση που προκύπτει είναι:



Εικόνα 4.7 - Ροή εργασίας 2 σε συστάδες (Πηγή [23])

Στην Εικόνα 4.7 παρατηρούμε ότι υπάρχουν δύο συστάδες με τέσσερις κόμβους, οι οποίες έχουν μέγιστο μονοπάτι τριών κόμβων. Κατά την ανάθεση των διεργασιών στους πόρους ο αλγόριθμος αντιστοιχεί μια διεργασία στον πόρο όπου προσφέρει τον ελάχιστο χρόνο ολοκλήρωσης. Στην Εικόνα 4.8 παρουσιάζεται η αντιστοίχιση κατά την εκτέλεση. Παρατηρούμε ότι οι διεργασίες που σχηματίζουν μονοπάτι τριών κόμβων 4→7→10 και 2→5→9 χρησιμοποιούν τον ίδιο πόρο.



Εικόνα 4.8 - Εκτέλεση διεργασιών ροής εργασίας 2 (Πηγή [23])

4.8 CSAAC

Ο CSAAC είναι ένας δυναμικός αλγόριθμος, όπου έχει βασιστεί στον αλγόριθμο CASA (Community-Aware Scheduling algorithm)[25], και έχει σαν στόχο την βελτίωση της επίδοσης του συστήματος καθώς και την μείωση της κατανάλωσης ενέργειας των πόρων. Το επιτυγχάνει με την μετατόπιση των διεργασιών σε όσο το δυνατόν λιγότερους πόρους , με αποτέλεσμα οι ελεύθεροι πόροι να εισέρχονται σε κατάσταση «αδράνειας» ελαχιστοποιώντας κατά πολύ την κατανάλωσή τους.

Οι πόροι ανήκουν στον ίδιο σύστημα και υπάρχει διασύνδεση μεταξύ τους. Σύμφωνα με τον CASA κάθε φορά που σε κάποιο πόρο i ανατίθεται μια διεργασία k, ο πόρος αυτός αναλαμβάνει τον ρόλο του αιτούντα. Συγκεκριμένα, παράγει ένα μήνυμα $req_{i,k}$ προς όλους τους συνδεδεμένους πόρους ζητώντας να αναλάβει κάποιος την εκτέλεση της διεργασίας. Το μήνυμα αυτό περιέχει πληροφορίες όπως εκτιμώμενος χρόνος ολοκλήρωσης, την απαιτούμενη ενέργεια καθώς και το φορτίο του πόρου.

Κάθε πόρος όπου λαμβάνει αυτό το μήνυμα, υπολογίζει τον εκτιμώμενο χρόνο εκτέλεσης σύμφωνα με τον χρόνο των τρεχουσών διεργασιών που εκτελεί, καθώς και την

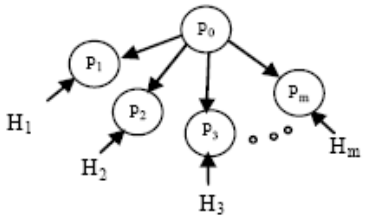
απαιτούμενη ενέργεια. Αν ο πόρος μπορεί να φιλοξενήσει την διεργασία τότε απαντά με ένα μήνυμα αποδοχής τύπου ACCEPT. Όταν υπάρχουν περισσότεροι του ενός πόρου που αποδέχονται την διεργασία τότε ο αλγόριθμος αποφασίζει με διάφορα κριτήρια τον καλύτερο υποψήφιο για την εκτέλεση της.

Τα κριτήρια αυτά βασίζονται σε διάφορες παραμέτρους, όπως ο χρόνος εκτέλεσης, τον φόρτο επικοινωνίας μεταξύ των πόρων και τέλος, η κατανάλωση ενέργειας. Παράλληλα, κατά την διάρκεια της εκτέλεσης, ο αλγόριθμος ελέγχει τον φόρτο εργασίας των πόρων και αποφασίζει αν είναι ικανοί να φιλοξενήσουν νέες διεργασίες. Αυτό επιτυγχάνεται με την χρήση δύο κατωφλίων. Μια τιμή κατωφλίου ***M*** που υποδεικνύει ότι ο πόρος είναι υπερφορτωμένος, και μια τιμή ***E*** που υποδεικνύει ότι έχει ελάχιστο φόρτο. Ο επιλεγόμενος πόρος είναι αυτός με τον ελάχιστο χρόνο εκτέλεσης, την ελάχιστη κατανάλωση και τον καλύτερο δυνατό φόρτο επικοινωνίας.

Σαν αποτέλεσμα έχουμε την ανάθεση των διεργασιών σε πόρους που έχουν τιμή κατωφλίου μεγαλύτερη από Ε και οριακά μικρότερη από Μ. Έτσι πολλές διεργασίες ανατίθενται σε λιγότερους πόρους και οι πόροι όπου δεν εξυπηρετούν κάποια διεργασία τίθενται σε κατάσταση αδράνειας. Ο αλγόριθμος CSAAC καταναλώνει λιγότερη ενέργεια από τον CASA όταν ο αριθμός των διεργασιών είναι Τ > 300, ενώ παράλληλα κατορθώνει να μειώσει την κατανάλωση ενέργειας από 5 – 80 % όταν ο αριθμός τον διεργασιών κυμαίνεται από 300 έως 1700.[26]

4.9 Reliable Scheduling Distributed in Cloud computing - RSDC

Ο αλγόριθμος RSDC προσπαθεί να επιτύχει καλύτερη διανομή των διεργασιών στους πόρους με σκοπό να μην υπάρχει υπερφόρτωση των πόρων και σαν αποτέλεσμα να μειώνεται ο συνολικός χρόνος εκτέλεσης, ενώ ο χρόνος επικοινωνίας μεταξύ τον πόρων λαμβάνεται ως ο χαμηλότερος δυνατόν και θεωρείται αμελητέος. Όπως βλέπουμε στην Εικόνα 4.9, οι πόροι είναι «λογικά» οργανωμένοι σε ιεραρχική μορφή δέντρου. Δέντρο είναι ένα σύνολο κόμβων (ίδιου τύπου) και ακμών, που συνδέουν τους κόμβους, με βάση κάποια σχέση που δημιουργεί την ιεραρχική δομή των κόμβων, ενώ ένας από τους κόμβους αποτελεί και την ρίζα του δέντρου. Για τον αλγόριθμο υπάρχει ένα σύνολο από πόρους P και η ρίζα του $\mathbf{P_0}$ εκτελεί χρέη χρονοπρογραμματιστή. Η μεταβλητή $\mathbf{H_m}$ αναφέρεται στο σύνολο των διεργασιών Η που εκτελούνται.



Εικόνα 4.9 - Δέντρο ενός επιπέδου (Πηγή [31])

Ο αλγόριθμος χωρίζεται σε 2 φάσεις. Στην πρώτη φάση το σύνολο των διεργασιών συλλέγεται από τον πόρο $\mathbf{P_0}$, όπου λειτουργεί σαν δρομολογητής διεργασιών. Στόχος του αλγορίθμου είναι η ισομερής κατανομή των διεργασιών επιτυγχάνοντας μείωση του συνολικού χρόνου εκτέλεσης. Η στρατηγική που ακολουθείται είναι αντιστοίχηση όσο το δυνατόν περισσότερων διεργασιών στους πόρους με την υψηλότερη υπολογιστική ισχύ, και λιγότερων διεργασιών στους πόρους με την χαμηλότερη υπολογιστική ισχύ.

Στην συνέχεια υπολογίζεται ο χρόνος εκτέλεσης για κάθε πόρο σύμφωνα με το φορτίο του. Αν υπερβαίνει ένα συγκεκριμένο κατώφλι τότε θεωρείται υπερ-φορτωμένος. Από την άλλη πλευρά, οι πόροι που ικανοποιούν το κατώφλι θεωρούνται υπο-φορτωμένοι. Τέλος, στην δεύτερη φάση ο αλγόριθμος ενεργοποιεί την μεταφορά διεργασιών από τους υπερ-φορτωμένους προς τους υπο-φορτωμένους.[31]

4.10 Compromised-Time-Cost Scheduling Algorithm

Ο αλγόριθμος CTC (Compromised-Time-Cost) μπορούμε να πούμε ότι ικανοποιεί πολλά από τα χαρακτηριστικά του υπολογιστικού νέφους. Αυτό διότι δεν στοχεύει μόνο στην μείωση του χρόνου εκτέλεσης ή του ισομερούς διαμοιρασμού των πόρων, αλλά επίσης και στις απαιτήσεις του πελάτη καθώς και το κόστος εκτέλεσης των διεργασιών στους πόρους. Ο αλγόριθμος είναι δυναμικός και πέρα από τον χρονοπρογραμματισμό διεργασιών, στοχεύει και στην διαχείριση πολλών ροών εργασίας ομαδοποιημένες σε συστάδα. Εστιάζει στα εξής στοιχεία:

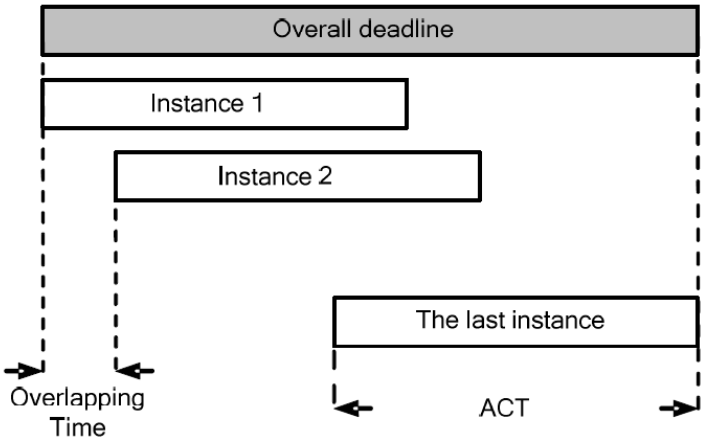
- Το κόστος, με στόχο την μείωση του, ικανοποιώντας τις απαιτήσεις του χρήστη στα απαιτούμενα χρονικά όρια.
- Στην προσαρμοστικότητα του αλγορίθμου σε νέες συνθήκες, βάση των απαιτήσεων του χρήστη την στιγμή που ζητούνται.
- Τον δυναμικό διαμοιρασμό των πόρων σε συνθήκες υψηλών απαιτήσεων, βάση του φόρτου εργασίας όπως αυτός φανερώνεται από τον υπολογισμό του χρόνου εκτέλεσης μιας διεργασίας σε όλους τους διαθέσιμους πόρους.

Αρχικά πραγματοποιείται έλεγχος του αριθμού των διεργασίες που δεν έχουν αντιστοιχηθεί σε κάποιο πόρο και τις εντάσσει σε ένα πίνακα. Οι διεργασίες αυτές δημιουργούν μια ή περισσότερες ροές εργασίας προς εκτέλεση. Σύμφωνα με τις απαιτήσεις του χρήστη, έχει οριστεί χρονική προθεσμία εκτέλεσης όλων των διεργασιών. Για να επιτευχθεί αυτή, ο αλγόριθμος ορίζει μία υπο-προθεσμία (sub-deadline) ως τον μέγιστο χρόνο ολοκλήρωσης μίας διεργασίας και στην συνέχεια υπολογίζει αυτό τον χρόνο για τις διεργασίες τις τελευταίας ροής εργασίας. Αφαιρετικά (όπως στον αλγόριθμο HEFT) υπολογίζονται οι υπο-προθεσμίες διεργασιών των προηγούμενων ροών εργασίας ως εξής:

$$ACT + (n-m)\cdot OT = OD, \text{ όπου}$$

- ACT, ορίζεται ως ο μέσος χρόνος ολοκλήρωσης μιας ροής εργασίας.
- OT, ορίζεται ως το χρονικό διάστημα μεταξύ της εκκίνησης δύο διεργασιών.
- OD, ορίζεται ως ο συνολικός χρόνος της προθεσμίας
- n, ορίζεται ως ο συνολικός αριθμός των ροών εργασίας
- m, ορίζεται ως ο τρέχων αριθμός ροής εργασίας

Ο χρόνος της τελευταίας διεργασίας ισούται με τον συνολικό χρόνο προθεσμίας OD. Στην Εικόνα 4.10 βλέπουμε πως κατανέμονται οι παραπάνω χρόνοι.



Εικόνα 4.10 - Συστάδα ροών εργασίας (Πηγή [27])

Έστω ότι έχουμε τρείς ροές εργασιών I_1, I_2, I_3, τρείς διεργασίες T_1, T_2, T_3 με εκτιμώμενο χρόνο εκτέλεσης 2,3 και 2 αντίστοιχα ενώ το χρονικό διάστημα μεταξύ της εκκίνησης δύο διεργασιών ισούται με 2. Ο χρόνος υπο-προθεσμίας της τελευταίας διεργασίας υπολογίζεται ως,

ACT=2+3+2=7, n=3 και OD=I_3T_3=7 +(3-1)·2=11. Ενώ για τον υπολογισμό των υπόλοιπων υπο-προθεσμιών ισχύει ότι ο χρόνος I_3T_2 ισούται με τον χρόνο υπο-προθεσμίας της I_3T_3 πλην τον εκτιμώμενο χρόνο εκτέλεσης της T_3. Οπότε έχουμε I_3T_2=11-2=9 και αντίστοιχα I_3T_1=9-3=6. Για τον υπολογισμό των υπο-προθεσμιών της ροής εργασίας 2, έχουμε I_2T_3=OD=11, I_2T_2= I_3T_2 – (3-2)·2 = 7, I_2T_1= I_3T_1 – (3-1)·2 = 2, κ.ο.κ.

Μόλις ολοκληρωθεί ο υπολογισμός των υπο-προθεσμιών, το επόμενο βήμα είναι ο υπολογισμός του συνολικού χρόνου εκτέλεσης σε συνάρτηση με το κόστος εκτέλεσης. Οι πόροι με την υψηλότερη επεξεργαστική ισχύ και οι διασυνδέσεις με την ταχύτερη μεταφορά δεδομένων είναι σαφώς και οι πιο ακριβοί. Για την κάθε υπηρεσία πόρων και διασυνδέσεων ορίζεται και το αντίστοιχο κόστος. Έστω οι παρακάτω πίνακες:

Πίνακας 11. Υπηρεσίες πόρου

Υπηρεσία	**Ταχύτητα εκτέλεσης**	**Κόστος (€/διεργασία)**
1	0,5	3
2	1	6
3	2	12

Πίνακας 12. Υπηρεσίες διασύνδεσης

Υπηρεσία	**Εύρος ζώνης (Mbps)**	**Κόστος (€/MB)**
1	100	1
2	1000	10
3	10000	100

Για κάθε υπηρεσία Υ, με ταχύτητα εκτέλεσης ΤΕ, φορτίο Φ, ο χρόνος εκτέλεσης υπολογίζεται ως ΧΕ=ΤΕ/Φ, ενώ ο χρόνος μεταφοράς δεδομένων ΧΜ με μέγεθος δεδομένων ΜΔ και εύρος ζώνης ΕΖ υπολογίζεται ως ΧΜ=ΜΔ/ΕΖ. Ο συνολικός χρόνος εκτέλεσης ισούται με το άθροισμα του χρόνου εκτέλεσης και του χρόνου μεταφοράς δεδομένων. Αντίστοιχα το συνολικό κόστος υπολογίζεται από το άθροισμα του κόστους των υπηρεσιών πόρων και διασυνδέσεων. Η παραπάνω διαδικασία εκτελείται για κάθε διεργασία

Στο επόμενο βήμα ο αλγόριθμος χρησιμοποιώντας την μέθοδο του FCFS, αντιστοιχεί κάθε διεργασία στην αντίστοιχη υπηρεσία η οποία προσφέρει χρόνο εκτέλεσης ο οποίος δεν ξεπερνά την προθεσμία του χρήστη καθώς και το επιτρεπόμενο κόστος. Τέλος προσφέρει στον χρήστη «ζωντανή» γραφική αναπαράσταση μεταξύ του χρόνου εκτέλεσης και του κόστους, δίνοντας την δυνατότητα να ορίσει τυχόν νέους ταχύτερους χρόνους εκτέλεσης, αυξάνοντας το κόστος ή το αντίστροφο. Σε περίπτωση που ο χρήστης ορίσει νέες συνθήκες τότε ο αλγόριθμος επαναλαμβάνει τα παραπάνω βήματα για τον υπολογισμό του συνολικού χρόνου εκτέλεσης και συνολικού κόστους.[27]

5 Προσομοίωση

Οι εφαρμογές υπολογιστικού νέφους υποστηρίζουν διαφορετικές απαιτήσεις και εφαρμογές. Ο χρονοπρογραμματισμός και η μέτρηση της απόδοσης αυτών των εφαρμογών σε πραγματικό περιβάλλον είναι κάτι που απαιτεί αρκετό ρίσκο, καθώς και επιπλέον κόστος. Για τον λόγο αυτό έχουν αναπτυχθεί εφαρμογές προσομοίωσης περιβάλλοντος νέφους, όπου βοηθούν στην επαλήθευση των απαιτήσεων με ένα πιο απλό τρόπο. Τα βασικά πλεονεκτήματα είναι:

- Η μείωση του κόστους καθώς δεν απαιτείται πραγματική υποδομή νέφους
- Αξιολόγηση της απόδοσης των εφαρμογών νέφους πριν την υλοποίησή τους σε πραγματικές υποδομές

Σε περίπτωση που αυτές οι εφαρμογές προσομοίωσης δεν χρησιμοποιούνται από τους παρόχους τότε θα πρέπει να στηριχθούν στις θεωρητικές μετρήσεις για την αξιολόγηση και επαλήθευση των απαιτήσεων. Στην συνέχεια θα παρουσιάσουμε την προσομοίωση προαναφερθέντων αλγορίθμων σε ένα εικονικό περιβάλλον νέφους.

5.1 Εργαλεία

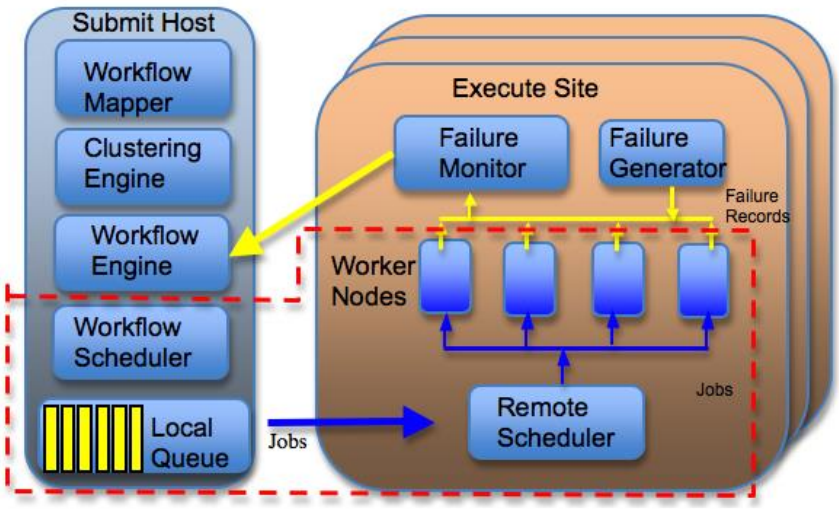
Για την προσομοίωση χρησιμοποιήσαμε την πλατφόρμα WorkflowSim η οποία αποτελεί επέκταση της πλατφόρμας CloudSim και επιτρέπει την μοντελοποίηση νέφους όπου περιλαμβάνουν εικονικές μηχανές, κέντρα δεδομένων καθώς και την υλοποίηση αλγορίθμων χρονοπρογραμματισμού. Μπορούμε να πούμε ότι το WorkflowSim αποτελεί ένα επιπλέον επίπεδο πάνω από το CloudSim.[24] Η αρχιτεκτονική του WorkflowSim παρουσίαζεται στην Εικόνα 5.1 και έχει ως εξής:

- Workflow Mapper

  Ο workflow mapper αντιστοιχεί την αφαιρετική μορφή μια ροής εργασίας σε μια πραγματική ροή, εισάγοντας αρχεία γράφου (DAG files). Τα αρχεία αυτά είναι σε μορφή XML και περιέχουν πληροφορίες σχετικά με την διασύνδεση των κόμβων καθώς και του χρόνου εκτέλεσης των διεργασιών. Από αυτή την πληροφορία δημιουργείται η ροή εργασίας και αντιστοιχεί τις διεργασία με τους διαθέσιμους πόρους.

- Clustering Engine

  Η συγκεκριμένη οντότητα ομαδοποιεί πολλές μικρές διεργασίες σε μια μεγάλη διεργασία με στόχο τον ευκολότερο χρονοπρογραμματισμό διεργασιών. Η συστάδα αυτή διεργασιών, ως προς τον χρήστη θα φαίνεται σαν μια εφαρμογή η οποία θα εκτελεί πολλές διεργασίες σειριακά ή παράλληλα.



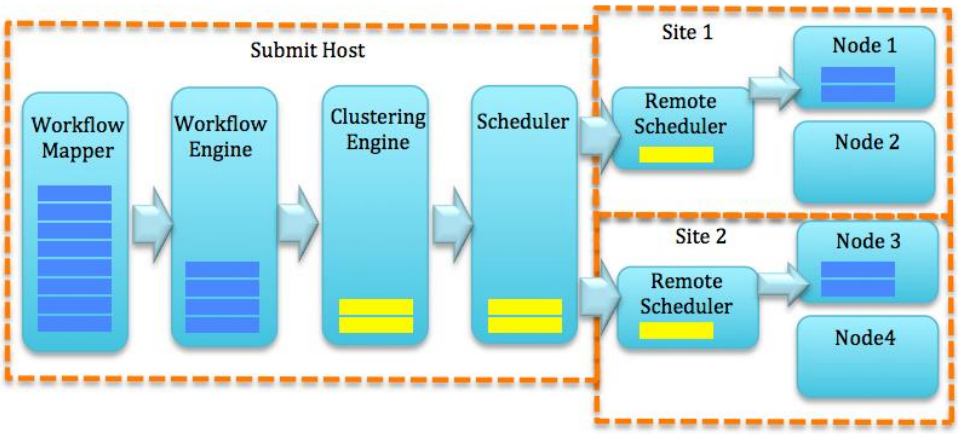
Εικόνα 5.1 - Αρχιτεκτονική της πλατφόρμας WorkflowSim (Πηγή [24])

- Workflow Engine

  Η οντότητα αυτή διαχειρίζεται τις σχέσεις δεδομένων μεταξύ των κόμβων της ροής εργασίας. Μια διεργασία θα μπορεί να εκτελεστεί μόνο όταν όλες οι προηγούμενες διασυνδεδεμένες διεργασίες του γράφου έχουν ολοκληρωθεί.

- Workflow Scheduler

  Η οντότητα αυτή χρησιμοποιείται για τον χρονοπρογραμματισμό των διεργασιών στους διαθέσιμους πόρους σύμφωνα με τα κριτήρια που έχουν οριστεί από τον χρήστη. Η πλατφόρμα WorfkflowSim προσφέρει δυναμικό χρονοπρογραμματισμό διεργασιών. Τα κριτήρια ελέγχονται από τον χρονοπρογραμματιστή, εφαρμόζοντας τους αντίστοιχους αλγορίθμους.



Εικόνα 5.2 - Επικοινωνία μεταξύ των οντοτήτων (Πηγή [24])

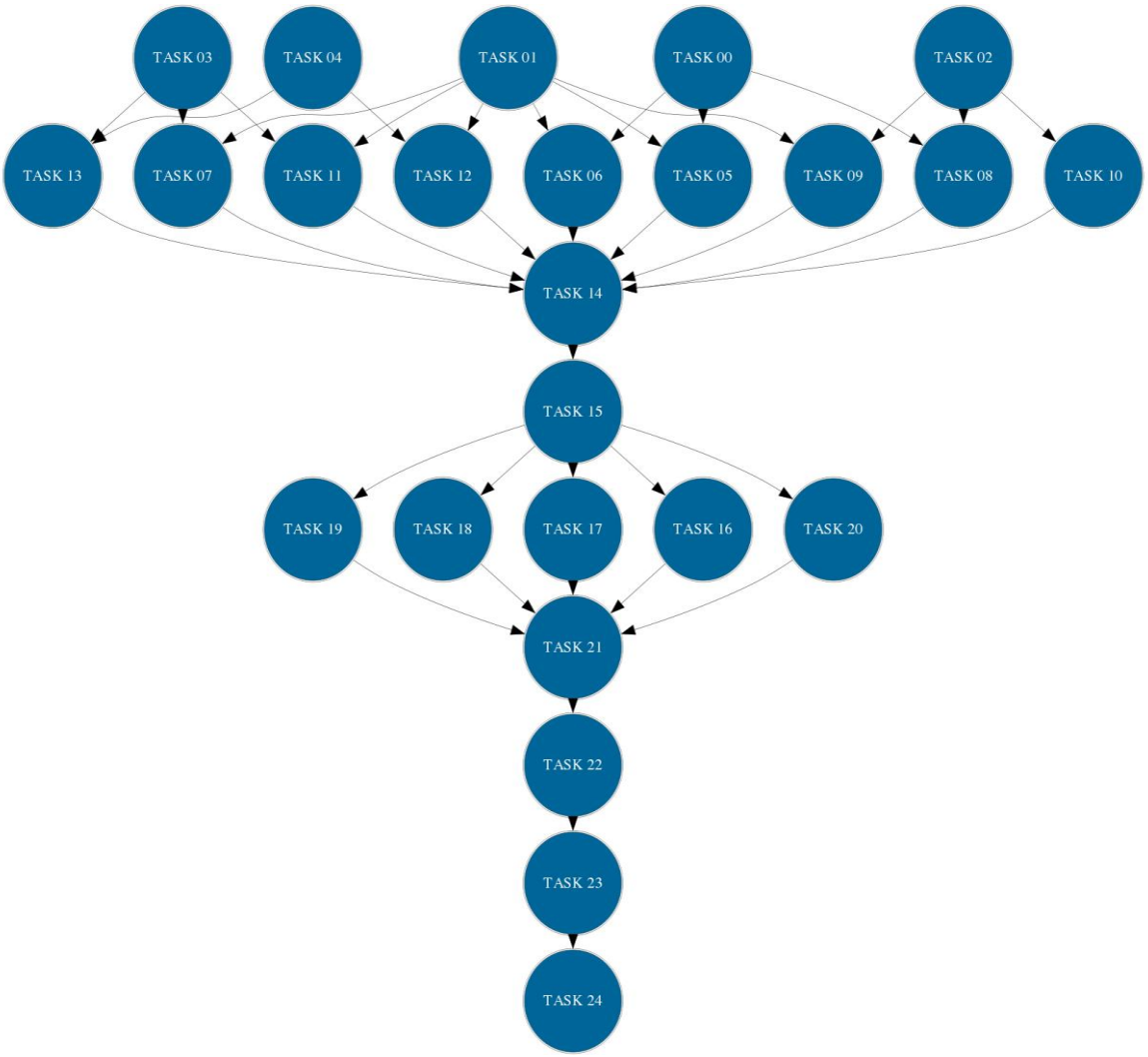
Η πλατφόρμα WorkflowSim υποστηρίζει σενάρια προσομοίωσης αποτυχίας κατά την εκτέλεση όπως καθυστέρηση μεταξύ ανάθεσης διεργασιών, καθυστέρηση σε ουρές αναμονής, καθυστέρηση στην μεταφορά δεδομένων καθώς και καθυστέρηση στον χρονοπρογραμματισμό συστάδων.

5.2 Πειραματικά Αποτελέσματα

Στην ενότητα αυτή παρουσιάζονται και συγκρίνονται τα χαρακτηριστικά των αλγορίθμων FCFS, MCT, Min-Min και Max-Min. Η προσομοίωση πραγματοποιήθηκε με την πλατφόρμα WorkflowSim για δύο διαφορετικές ροές εργασίας. Οι ροές εργασίας που χρησιμοποιήθηκαν είναι δύο ειδών άκυκλοι γράφοι γνωστοί και ως Montage και CyberShake και παράχθηκαν από την εφαρμογή Pegasus [28].

5.2.1 Ροη εργασίας Montage

Η ροή εργασίας της Εικόνας 5.3 αποτελείται από 25 διεργασίες με τους εξής χρόνους εκτέλεσης. Στην συνέχεια παρουσιάζεται ο συνολικός χρόνος εκτέλεσης της ροής εργασίας για τους παραπάνω αλγορίθμους στους



Εικόνα 5.3 - Σχήμα ροής εργασίας Montage

Πίνακας 13. Χρόνοι εκτέλεσης διεργασιών ροής εργασίας Montage

TASK00	*13.39*	**TASK13**	*10.37*
TASK01	*13.83*	**TASK14**	*0.72*
TASK02	*13.36*	**TASK15**	*1.42*
TASK03	*13.6*	**TASK16**	*10.39*
TASK04	*13.78*	**TASK17**	*10.64*
TASK05	*10.59*	**TASK18**	*10.83*
TASK06	*10.59*	**TASK19**	*10.93*
TASK07	*10.88*	**TASK20**	*10.76*
TASK08	*10.81*	**TASK21**	*1.39*
TASK09	*10.49*	**TASK22**	*3.03*
TASK10	*10.51*	**TASK23**	*3.86*
TASK11	*10.51*	**TASK24**	*0.45*
TASK12	*10.62*		

- **FCFS**

Πίνακας 14. Συνολικός χρόνος εκτέλεσης ροής εργασίας Montage για τον FCFS

Job_ID	Task_ID	STATUS	Data_cent	VM_ID	Time	Start_Tim	Finish_Tin
25		SUCCESS	2	0	0.05	0.1	0.15
2	3,	SUCCESS	2	2	6.68	0.15	6.84
0	1,	SUCCESS	2	0	6.7	0.15	6.85
3	4,	SUCCESS	2	3	6.81	0.15	6.96
1	2,	SUCCESS	2	1	6.92	0.15	7.07
10	11,	SUCCESS	2	0	5.26	6.85	12.12
5	6,	SUCCESS	2	1	5.3	7.07	12.38
8	9,	SUCCESS	2	3	5.52	6.96	12.49
4	5,	SUCCESS	2	2	6.89	6.84	13.73
6	7,	SUCCESS	2	0	5.3	12.12	17.42
9	10,	SUCCESS	2	3	5.25	12.49	17.74
7	8,	SUCCESS	2	1	5.45	12.38	17.82
11	12,	SUCCESS	2	2	5.27	13.73	19
12	13,	SUCCESS	2	0	5.32	17.42	22.74
13	14,	SUCCESS	2	3	5.19	17.74	22.93
14	15,	SUCCESS	2	0	0.36	22.93	23.29
15	16,	SUCCESS	2	0		23.29	24
16	17,	SUCCESS	2	0	5.2	24	29.2
17	18,	SUCCESS	2	1	5.32	24	29.32
18	19,	SUCCESS	2	2	5.41	24	29.42
19	20,	SUCCESS	2	3	5.46	24	29.47
20	21,	SUCCESS	2	0	5.38	29.2	34.58
21	22,	SUCCESS	2	0	0.72	34.58	35.3
22	23,	SUCCESS	2	0	1.52	35.3	36.81
23	24,	SUCCESS	2	0	1.93	36.81	38.74
24	25,	SUCCESS	2	0	0.23	38.74	38.97

- **MCT**

Πίνακας 15. Συνολικός χρόνος εκτέλεσης ροής εργασίας Montage για τον MCT

Job_ID	Task_ID	STATUS	Data_cent	VM_ID	Time	Start_Tim	Finish_Tin
25		SUCCESS	2	0	0.05	0.1	0.15
2	3,	SUCCESS	2	2	6.68	0.15	6.84
0	1,	SUCCESS	2	0	6.7	0.15	6.85
3	4,	SUCCESS	2	3	6.81	0.15	6.96
1	2,	SUCCESS	2	1	6.92	0.15	7.07
10	11,	SUCCESS	2	0	5.26	6.85	12.12
5	6,	SUCCESS	2	1	5.3	7.07	12.38
8	9,	SUCCESS	2	3	5.52	6.96	12.49
4	5,	SUCCESS	2	2	6.89	6.84	13.73
6	7,	SUCCESS	2	0	5.3	12.12	17.42
9	10,	SUCCESS	2	3	5.25	12.49	17.74
7	8,	SUCCESS	2	1	5.45	12.38	17.82
11	12,	SUCCESS	2	2	5.27	13.73	19
12	13,	SUCCESS	2	0	5.32	17.42	22.74
13	14,	SUCCESS	2	3	5.19	17.74	22.93
14	15,	SUCCESS	2	0	0.36	22.93	23.29
15	16,	SUCCESS	2	0		23.29	24
16	17,	SUCCESS	2	0	5.2	24	29.2
17	18,	SUCCESS	2	1	5.32	24	29.32
18	19,	SUCCESS	2	2	5.41	24	29.42
19	20,	SUCCESS	2	3	5.46	24	29.47
20	21,	SUCCESS	2	0	5.38	29.2	34.58
21	22,	SUCCESS	2	0	0.72	34.58	35.3
22	23,	SUCCESS	2	0	1.52	35.3	36.81
23	24,	SUCCESS	2	0	1.93	36.81	38.74
24	25,	SUCCESS	2	0	0.23	38.74	38.97

- **Min-Min**

Πίνακας 15. Συνολικός χρόνος εκτέλεσης ροής εργασίας Montage για τον Min-Min

Job_ID	Task_ID	STATUS	Data_center_ID	VM_ID	Time	Start_Time	Finish_Time
25	-	SUCCESS	2	0	0.05	0.1	0.15
2	3,	SUCCESS	2	0	6.68	0.15	6.84
0	1,	SUCCESS	2	1	6.7	0.15	6.85
3	4,	SUCCESS	2	2	6.81	0.15	6.96
4	5,	SUCCESS	2	3	6.89	0.15	7.05
10	11,	SUCCESS	2	1	5.26	6.85	12.12
13	14,	SUCCESS	2	3	5.19	7.05	12.24
8	9,	SUCCESS	2	2	5.42	6.96	12.39
1	2,	SUCCESS	2	0	6.92	6.84	13.76
9	10,	SUCCESS	2	0	5.25	13.76	19
11	12,	SUCCESS	2	1	5.27	13.76	19.03
5	6,	SUCCESS	2	2	5.3	13.76	19.06
6	7,	SUCCESS	2	3	5.36	13.76	19.11
12	13,	SUCCESS	2	0	5.32	19	24.32
7	8,	SUCCESS	2	1	5.44	19.03	24.47
14	15,	SUCCESS	2	0	0.36	24.47	24.83
15	16,	SUCCESS	2	0	0.71	24.83	25.54
16	17,	SUCCESS	2	0	5.2	25.54	30.74
17	18,	SUCCESS	2	1	5.32	25.54	30.86
20	21,	SUCCESS	2	2	5.39	25.54	30.93
18	19,	SUCCESS	2	3	5.42	25.54	30.96
19	20,	SUCCESS	2	0	5.47	30.74	36.22
21	22,	SUCCESS	2	0	0.72	36.22	36.94
22	23,	SUCCESS	2	0	1.52	36.94	38.45
23	24,	SUCCESS	2	0	1.93	38.45	40.38
24	25,	SUCCESS	2	0	0.23	40.38	40.61

- **Max-Min**

Πίνακας 16. Συνολικός χρόνος εκτέλεσης ροής εργασίας Montage για τον Min-Min

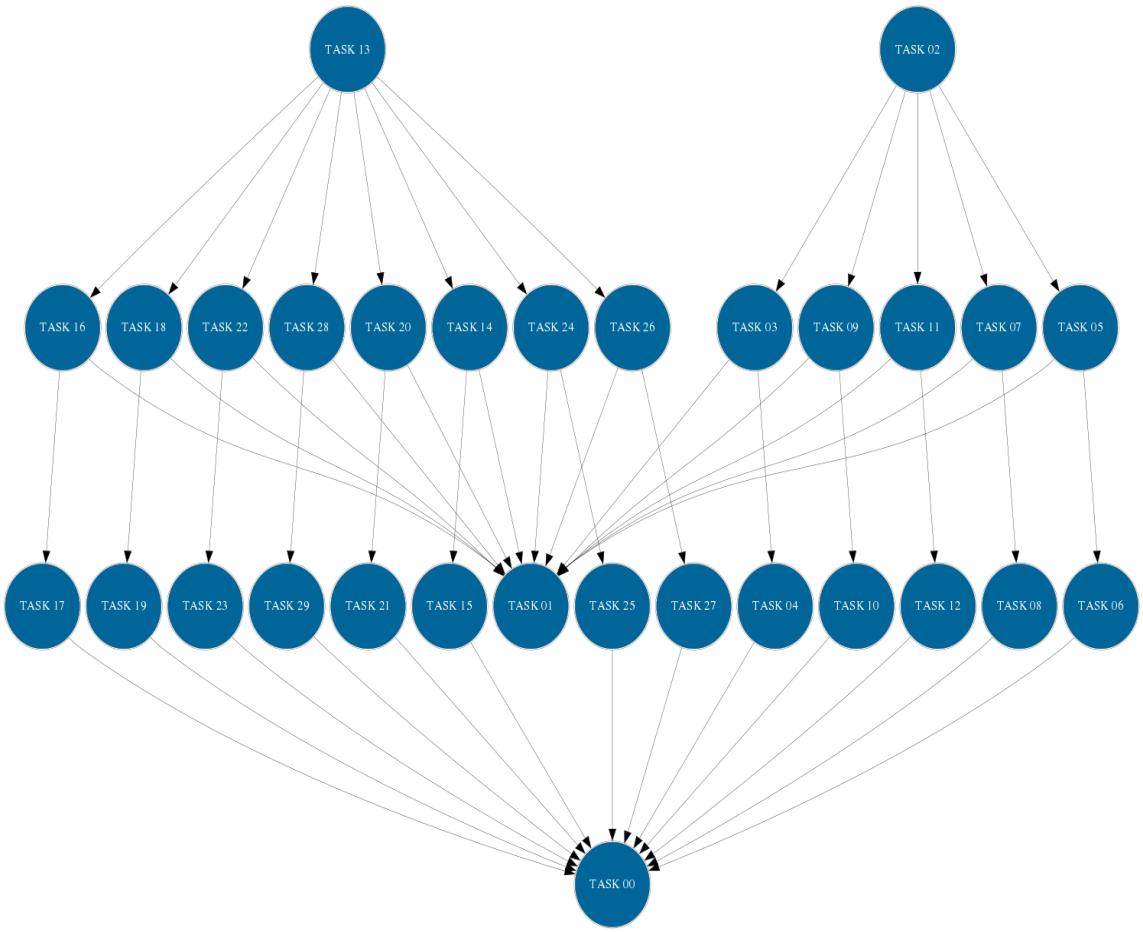
Job_ID	Task_ID	STATUS	Data_cen	VM_ID	Time	Start_Tim	Finish_Tin
25		SUCCESS	2	0	0.05	0.1	0.15
0	1,	SUCCESS	2	3	6.7	0.15	6.85
3	4,	SUCCESS	2	2	6.8	0.15	6.96
4	5,	SUCCESS	2	1	6.89	0.15	7.05
1	2,	SUCCESS	2	0	7	0.15	7.16
13	14,	SUCCESS	2	1	5.19	7.05	12.24
12	13,	SUCCESS	2	2	5.33	7.16	12.49
7	8,	SUCCESS	2	0	5.45	7.16	12.61
2	3,	SUCCESS	2	3	6.68	6.85	13.54
5	6,	SUCCESS	2	1	5.31	12.24	17.55
6	7,	SUCCESS	2	2	5.3	12.49	17.79
11	12,	SUCCESS	2	0	5.25	12.61	17.86
8	9,	SUCCESS	2	3	5.4	13.54	18.94
10	11,	SUCCESS	2	1	5.26	17.55	22.82
9	10,	SUCCESS	2	2	5.25	17.79	23.04
14	15,	SUCCESS	2	0	0.36	23.04	23.4
15	16,	SUCCESS	2	0	0.71	23.4	24.11
17	18,	SUCCESS	2	3	5.33	24.11	29.44
20	21,	SUCCESS	2	2	5.38	24.11	29.49
18	19,	SUCCESS	2	1	5.41	24.11	29.53
19	20,	SUCCESS	2	0	5.46	24.11	29.58
16	17,	SUCCESS	2	3	5.19	29.44	34.63
21	22,	SUCCESS	2	0	0.73	34.63	35.36
22	23,	SUCCESS	2	0	1.52	35.36	36.88
23	24,	SUCCESS	2	0	1.93	36.88	38.81
24	25,	SUCCESS	2	0	0.23	38.81	39.03

5.2.2 Ροή εργασίας Cybershake

Η συγκεκριμένη ροή εργασίας αποτελείται από 30 διεργασίες με τους εξής χρόνους εκτέλεσης. Στην συνέχεια παρουσιάζεται ο συνολικός χρόνος εκτέλεσης της ροής εργασίας

Πίνακας 17. Χρόνοι εκτέλεσης διεργασιών ροής εργασίας Montage

TASK00	*0.07*	**TASK10**	*0.74*	**TASK20**	*24.56*
TASK01	*0.19*	**TASK11**	*62.25*	**TASK21**	*1.36*
TASK02	*158.1*	**TASK12**	*1.42*	**TASK22**	*31.05*
TASK03	*39.06*	**TASK13**	*96.91*	**TASK23**	*1.43*
TASK04	*0.63*	**TASK14**	*47.44*	**TASK24**	*54.87*
TASK05	*38.49*	**TASK15**	*1.53*	**TASK25**	*0.78*
TASK06	*0.8*	**TASK16**	*45.6*	**TASK26**	*23.99*
TASK07	*36.27*	**TASK17**	*1.53*	**TASK27**	*1.1*
TASK08	*0.73*	**TASK18**	*28.67*	**TASK28**	*26.46*
TASK09	*32.29*	**TASK19**	*1.36*	**TASK29**	*0.85*



Εικόνα 5.4 - Σχήμα ροής εργασίας Cybershake

- **FCFS**

Πίνακας 18. Συνολικός χρόνος εκτέλεσης ροής εργασίας Cybershake για τον FCFS

Job_ID	Task_ID	STATUS	Data_cent	VM_ID	Time	Start_Tim	Finish_Time
30		SUCCESS	2	0	0.05	0.1	0.15
13	14,	SUCCESS	2	1	88.58	0.15	88.73
18	19,	SUCCESS	2	3	14.89	88.73	103.62
16	17,	SUCCESS	2	2	23.35	88.73	112.08
14	15,	SUCCESS	2	1	23.72	88.73	112.46
20	21,	SUCCESS	2	3	12.28	103.62	115.9
2	3,	SUCCESS	2	0	119.17	0.15	119.32
22	23,	SUCCESS	2	2	15.53	112.08	127.62
26	27,	SUCCESS	2	3	12	115.9	127.9
19	20,	SUCCESS	2	2	0.68	127.62	128.3
17	18,	SUCCESS	2	3	0.77	127.9	128.66
15	16,	SUCCESS	2	2	0.76	128.3	129.06
21	22,	SUCCESS	2	3	0.68	128.66	129.34
28	29,	SUCCESS	2	0	13.78	119.32	133.11
24	25,	SUCCESS	2	1	27.44	112.46	139.9
5	6,	SUCCESS	2	3	19.87	129.34	149.21
3	4,	SUCCESS	2	2	20.26	129.06	149.32
23	24,	SUCCESS	2	2	0.72	149.32	150.04
27	28,	SUCCESS	2	2	0.55	150.04	150.59
29	30,	SUCCESS	2	2	0.43	150.59	151.01
7	8,	SUCCESS	2	0	18.14	133.11	151.25
25	26,	SUCCESS	2	2	0.39	151.01	151.4
6	7,	SUCCESS	2	0	0.4	151.25	151.65
4	5,	SUCCESS	2	2	0.31	151.4	151.72
8	9,	SUCCESS	2	0	0.37	151.65	152.01
9	10,	SUCCESS	2	1	16.77	139.9	156.67
10	11,	SUCCESS	2	0	0.37	156.67	157.04
11	12,	SUCCESS	2	3	31.13	149.21	180.34
1	2,	SUCCESS	2	0	0.09	180.34	180.44
12	13,	SUCCESS	2	1	0.71	180.34	181.05
0	1,	SUCCESS	2	0	0.05	181.05	181.1

- **MCT**

Πίνακας 19. Συνολικός χρόνος εκτέλεσης ροής εργασίας Cybershake για τον MCT

Job_ID	Task_ID	STATUS	Data_cent	VM_ID	Time	Start_Tim	Finish_Time
30		SUCCESS	2	0	0.05	0.1	0.15
13	14,	SUCCESS	2	1	88.58	0.15	88.73
18	19,	SUCCESS	2	3	14.89	88.73	103.62
16	17,	SUCCESS	2	2	23.35	88.73	112.08
14	15,	SUCCESS	2	1	23.72	88.73	112.46
20	21,	SUCCESS	2	3	12.28	103.62	115.9
2	3,	SUCCESS	2	0	119.17	0.15	119.32
22	23,	SUCCESS	2	2	15.53	112.08	127.62
26	27,	SUCCESS	2	3	12	115.9	127.9
19	20,	SUCCESS	2	2	0.68	127.62	128.3
17	18,	SUCCESS	2	3	0.77	127.9	128.66
15	16,	SUCCESS	2	2	0.76	128.3	129.06
21	22,	SUCCESS	2	3	0.68	128.66	129.34
28	29,	SUCCESS	2	0	13.78	119.32	133.11
24	25,	SUCCESS	2	1	27.44	112.46	139.9
5	6,	SUCCESS	2	3	19.87	129.34	149.21
3	4,	SUCCESS	2	2	20.26	129.06	149.32
23	24,	SUCCESS	2	2	0.72	149.32	150.04
27	28,	SUCCESS	2	2	0.55	150.04	150.59
29	30,	SUCCESS	2	2	0.43	150.59	151.01
7	8,	SUCCESS	2	0	18.14	133.11	151.25
25	26,	SUCCESS	2	2	0.39	151.01	151.4
6	7,	SUCCESS	2	0	0.4	151.25	151.65
4	5,	SUCCESS	2	2	0.31	151.4	151.72
8	9,	SUCCESS	2	0	0.37	151.65	152.01
9	10,	SUCCESS	2	1	16.77	139.9	156.67
10	11,	SUCCESS	2	0	0.37	156.67	157.04
11	12,	SUCCESS	2	3	31.13	149.21	180.34
1	2,	SUCCESS	2	0	0.09	180.34	180.44
12	13,	SUCCESS	2	1	0.71	180.34	181.05
0	1,	SUCCESS	2	0	0.05	181.05	181.1

- **Min-Min**

Πίνακας 20. Συνολικός χρόνος εκτέλεσης ροής εργασίας Cybershake για τον Min-Min

Job_ID	Task_ID	STATUS	Data_cent	VM_ID	Time	Start_Time	Finish_Time
30		SUCCESS	2	0	0.05	0.1	0.15
13	14,	SUCCESS	2	0	88.58	0.15	88.73
26	27,	SUCCESS	2	0	12	88.73	100.73
20	21,	SUCCESS	2	2	12.83	88.73	101.56
27	28,	SUCCESS	2	2	0.55	101.56	102.11
28	29,	SUCCESS	2	3	13.78	88.73	102.52
21	22,	SUCCESS	2	2	0.68	102.11	102.79
29	30,	SUCCESS	2	2	0.42	102.79	103.22
18	19,	SUCCESS	2	0	14.34	100.73	115.07
22	23,	SUCCESS	2	3	15.53	102.52	118.05
19	20,	SUCCESS	2	3	0.68	118.05	118.73
2	3,	SUCCESS	2	1	119.17	0.15	119.32
23	24,	SUCCESS	2	3	0.72	118.73	119.44
16	17,	SUCCESS	2	2	22.8	103.22	126.02
9	10,	SUCCESS	2	3	16.77	119.44	136.21
17	18,	SUCCESS	2	3	0.76	136.21	136.98
10	11,	SUCCESS	2	3	0.37	136.98	137.35
14	15,	SUCCESS	2	0	23.72	115.07	138.79
7	8,	SUCCESS	2	2	18.76	126.02	144.78
15	16,	SUCCESS	2	2	0.76	144.78	145.55
8	9,	SUCCESS	2	2	0.37	145.55	145.91
24	25,	SUCCESS	2	1	27.99	119.32	147.31
25	26,	SUCCESS	2	1	0.39	147.31	147.7
5	6,	SUCCESS	2	3	19.25	137.35	156.59
6	7,	SUCCESS	2	1	0.4	156.59	156.99
3	4,	SUCCESS	2	0	20.16	138.79	158.95
4	5,	SUCCESS	2	0	0.31	158.95	159.27
11	12,	SUCCESS	2	2	31.13	145.91	177.04
1	2,	SUCCESS	2	0	0.09	177.04	177.14
12	13,	SUCCESS	2	1	0.71	177.04	177.75
0	1,	SUCCESS	2	0	0.05	177.75	177.8

- **Max-Min**

Πίνακας 21. Συνολικός χρόνος εκτέλεσης ροής εργασίας Cybershake για τον Max-Min

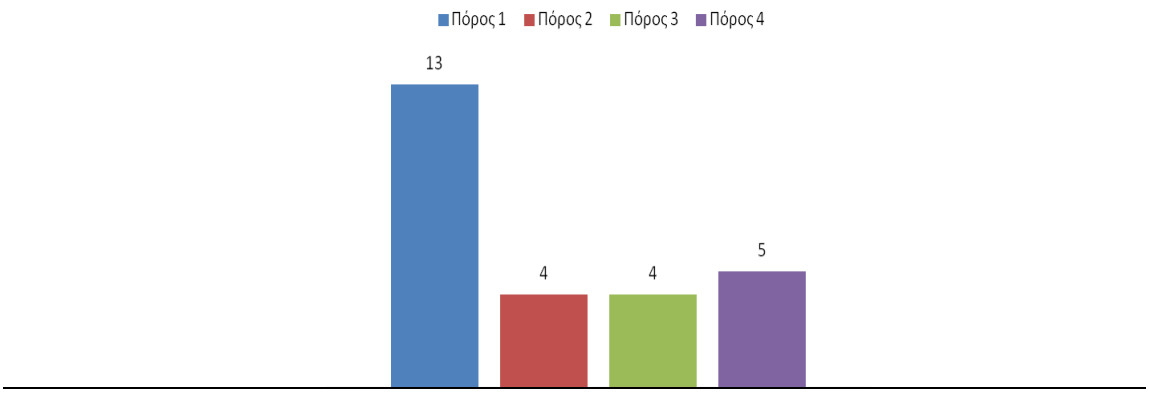
Job_ID	Task_ID	STATUS	Data_cent	VM_ID	Time	Start_Tim	Finish_Tin
30		SUCCESS	2	0	0.05	0.1	0.15
13	14,	SUCCESS	2	1	88.58	0.15	88.73
16	17,	SUCCESS	2	3	23.35	88.73	112.08
14	15,	SUCCESS	2	2	24.27	88.73	113.01
24	25,	SUCCESS	2	1	27.44	88.73	116.17
2	3,	SUCCESS	2	0	119.17	0.15	119.32
18	19,	SUCCESS	2	2	14.34	113.01	127.34
22	23,	SUCCESS	2	3	15.53	112.08	127.62
28	29,	SUCCESS	2	1	13.23	116.17	129.4
20	21,	SUCCESS	2	0	12.83	119.32	132.15
3	4,	SUCCESS	2	3	20.16	127.62	147.77
5	6,	SUCCESS	2	1	19.87	129.4	149.27
7	8,	SUCCESS	2	0	18.14	132.15	150.29
17	18,	SUCCESS	2	0	0.76	150.29	151.06
15	16,	SUCCESS	2	0	0.76	151.06	151.82
23	24,	SUCCESS	2	0	0.72	151.82	152.54
19	20,	SUCCESS	2	0	0.68	152.54	153.22
21	22,	SUCCESS	2	0	0.68	153.22	153.9
29	30,	SUCCESS	2	0	0.43	153.9	154.32
6	7,	SUCCESS	2	0	0.4	154.32	154.72
25	26,	SUCCESS	2	0	0.39	154.72	155.11
8	9,	SUCCESS	2	0	0.37	155.11	155.48
4	5,	SUCCESS	2	0	0.31	155.48	155.79
11	12,	SUCCESS	2	2	31.75	127.34	159.09
12	13,	SUCCESS	2	0	0.71	159.09	159.8
26	27,	SUCCESS	2	1	12	149.27	161.27
27	28,	SUCCESS	2	0	0.55	161.27	161.82
9	10,	SUCCESS	2	3	16.15	147.77	163.92
1	2,	SUCCESS	2	1	0.09	163.92	164.02
10	11,	SUCCESS	2	0	0.37	163.92	164.29
0	1,	SUCCESS	2	0	0.05	164.29	164.34

Στους παραπάνω πίνακες, η τελευταία γραμμή της τελευταίας στήλης δείχνει τον συνολικό χρόνο ολοκλήρωσης της ροής εργασίας. Παρατηρούμε ότι στην πρώτη ροή εργασίας Montage, ότι οι αλγόριθμοι FCFS και MCT προσφέρουν τον ίδιο χρόνο ολοκλήρωσης και παράλληλα τον μικρότερο. Από την άλλη πλευρά ο αλγόριθμος Max-Min προσφέρει μικρότερο χρόνο ολοκλήρωσης σε σύγκριση με τον αλγόριθμο Min-Min. Σχετικά με τον διαμοιρασμό διεργασιών τα καλύτερα αποτελέσματα τα προσφέρει ο αλγόριθμος Max-Min.

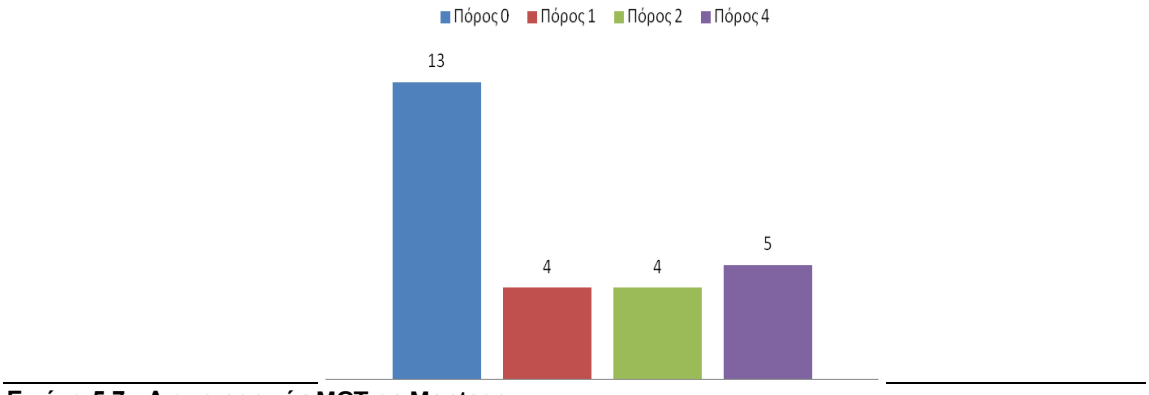
Από την άλλη μεριά, για την ροή εργασίας Cybershake η οποία έχει περισσότερες διασυνδέσεις και πολύπλοκη μορφή σε σχέση με την ροή εργασίας Montage, παρατηρούμε ότι ο αλγόριθμος Max-Min προσφέρει τον μικρότερο χρόνο ολοκλήρωσης από τους υπόλοιπους, ενώ ο Min-Min είναι ταχύτερος από τους MCT και FCFS. Σχετικά με τον διαμοιρασμό διεργασιών τα καλύτερα αποτελέσματα τα προσφέρει Min-Min. Στην συνέχεια παρουσιάζονται, ένας συγκεντρωτικός πίνακας με τους συνολικού χρόνους ολοκλήρωσης καθώς και τα διαγράμματα διαμοιρασμού των διεργασιών στους πόρους.

Πίνακας 22. Συγκεντρωτικός πίνακας συνολικού χρόνους ολοκλήρωσης

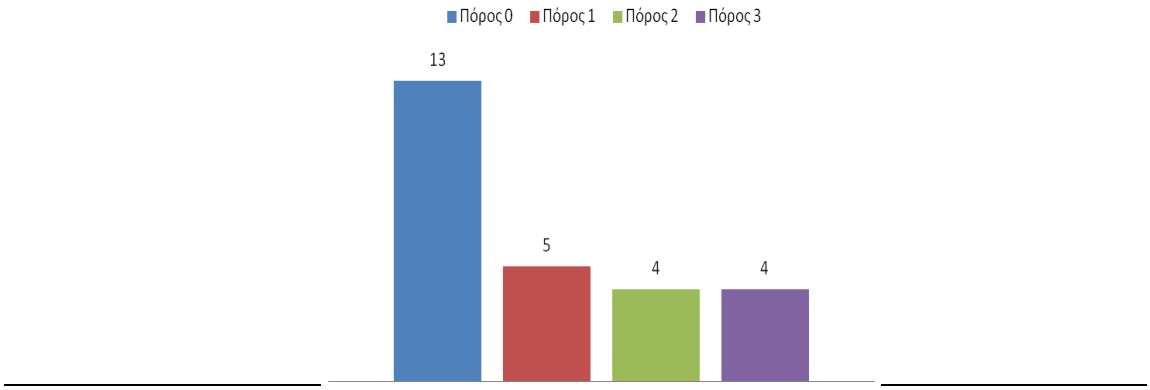
	FCFS	MCT	Min-Min	Max-Min
Montage	**38.97**	**38.97**	40.61	39.03
Cybershake	181.1	181.1	177.75	**164.34**



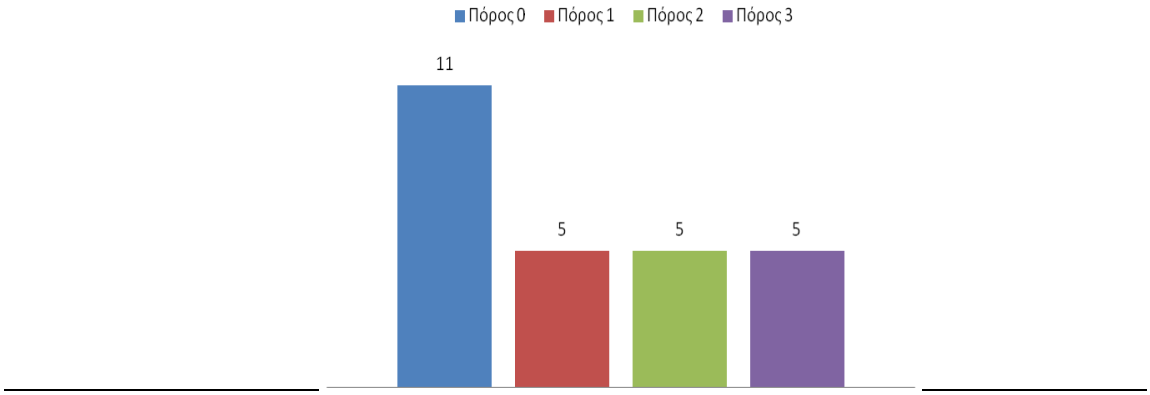
Εικόνα 5.6 - Διαμοιρασμός FCFS σε Montage



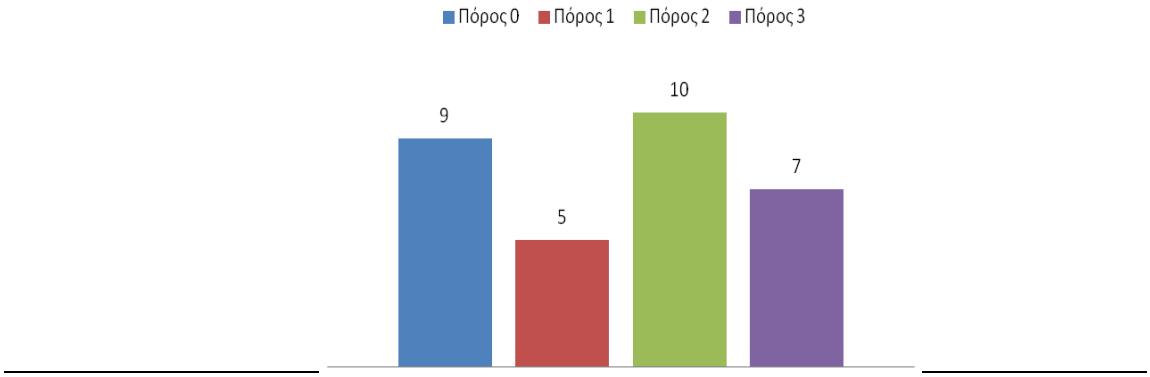
Εικόνα 5.7 - Διαμοιρασμός MCT σε Montage



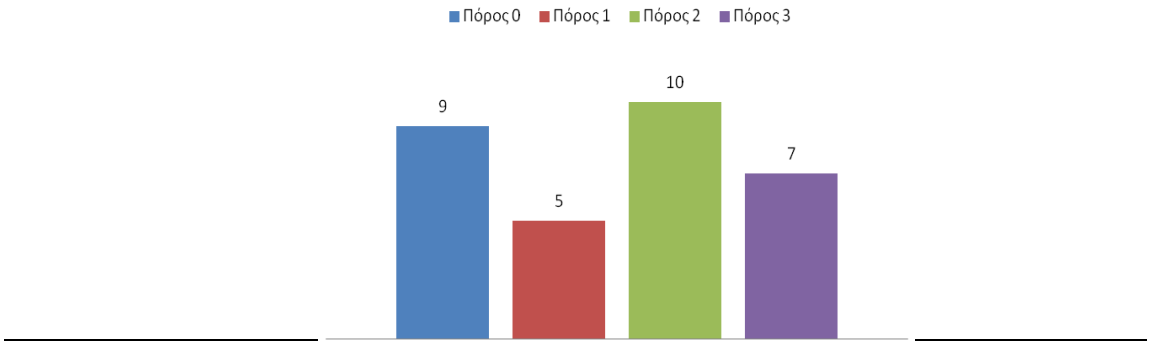
Εικόνα 5.8 - Διαμοιρασμός Min-Min σε Montage



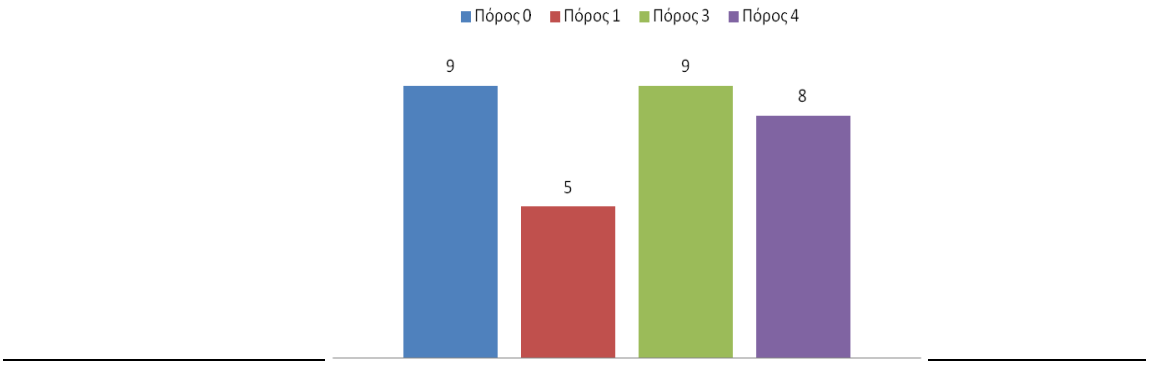
Εικόνα 5.9 - Διαμοιρασμός Max-Min σε Montage



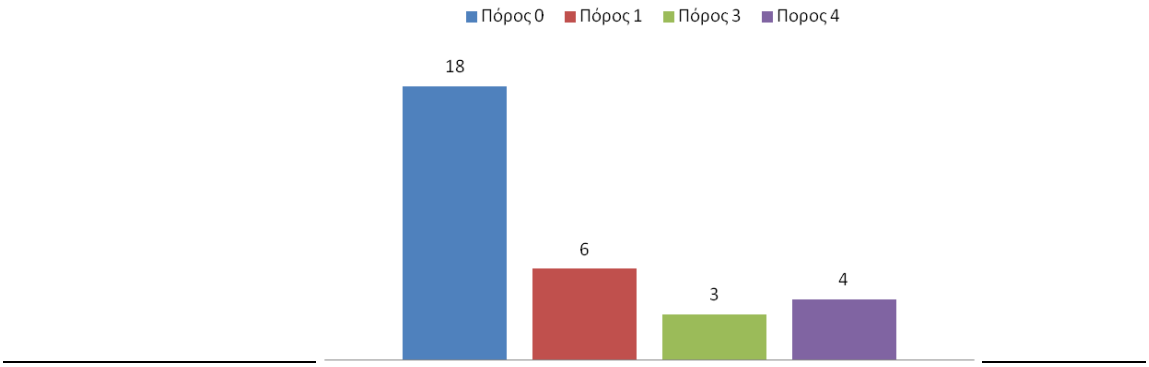
Εικόνα 5.10 - Διαμοιρασμός FCFS σε Cybershake



Εικόνα 5.11 - Διαμοιρασμός MCT σε Cybershake



Εικόνα 5.12 - Διαμοιρασμός Min-Min σε Cybershake



Εικόνα 5.13 - Διαμοιρασμός Max-Min σε Cybershake

6 Επίλογος

Υπάρχουν πολλοί αλγόριθμοι χρονοδρομολόγησης και η ζήτησή για νέες τεχνικές αυξάνεται όσο αυξάνονται και οι απαιτήσεις. Η εφαρμογή τους επηρεάζεται από τις συνθήκες του περιβάλλοντος καθώς και από την συμφωνία μεταξύ χρήστη και παρόχου. Δεν μπορούμε να πούμε ότι υπάρχει βέλτιστος αλγόριθμος, καθώς υπάρχουν διαφορετικά είδη αλγορίθμων. Κάποιοι προσφέρουν μειωμένο χρόνο εκτέλεσης, κάποιοι καλύτερη κατανομή διεργασιών στους πόρους και κάποιοι βελτιστοποιημένη απόδοση ή μείωση του κόστους.

Από τα παραπάνω πειραματικά αποτελέσματα παρατηρούμε ότι οι αλγόριθμοι επηρεάζονται από τις τρέχουσες συνθήκες. Για την ροή εργασίας Montage, είδαμε ότι ο Max-Min προσέφερε καλύτερη κατανομή διεργασιών στους πόρους, ενώ οι FCFS και MCT προσφέρουν καλύτερο χρόνο εκτέλεσης, σε σχέση με τους υπόλοιπους. Από την άλλη πλευρά, όταν εφαρμόσαμε μια πιο περίπλοκη ροή εργασίας (Cybershake), παρατηρήσαμε ότι η συμπεριφορά άλλαξε. Ο Min-Min προσέφερε καλύτερη κατανομή διεργασιών στους πόρους έναντι του Max-Min, και ο Max-Min προσέφερε καλύτερο χρόνο εκτέλεσης έναντι των υπολοίπων.

Βιβλιογραφία

[1] http://www.computerweekly.com/feature/A-history-of-cloud-computing

[2] I. Faynberg, Hui-Lan Lu, D.Skuler. Cloud Computing. John Wiley & Sons, January 19, 2016

[3] D. Rountree, Ileana Castrillo. The Basics of Cloud Computing, Syngress, September 3, 2013

[4] Z. Mahmood and R.Hill. Cloud Computing for enterprise architectures, Springer, 2011.

[5] NIST. The NIST Definition of Cloud Computing,National Institute of Standards and Technology, September 2011

[6] D.C. Marinescu. Cloud ComputingTheory and Practice, MK publications, 2013

[7] Information Resources Management Association. Cloud Technology: Concepts, Methodologies, Tools, and Applications, IGI Global, October 31, 2014

[8] J.R. Vacca. Cloud Computing Security, CRC Press, September 19, 2016

[9] VMware. Inc. «VMware Infrastructure 3 TCO Methodology", November 2006

[10] M. Portnoy. Virtualization Essentials, 2nd Edition, John Wiley & Sons, August 29, 2016

[11] R. Smith, Docker Orchestration. Packt Publishing, January 24, 2017

[12] Rajkumar Kannan, Raihan Ur Rasool, Hai Jin, S.R. Balasundaram. Managing and Processing Big Data in Cloud Computing, IGI Global, January 7, 2016

[13] D. Mishchenko. VMware ESXi: Planning, Implementation, and Security Cengage, Learning PTR 1 edition 2010

[14] Yong Dr. Zhao, Wenhong Dr. Tian. Optimized Cloud Resource Management and Scheduling, Morgan Kaufmann, October 15, 2014

[15] R. Stephens. Essential Algorithms: A Practical Approach to Computer Algorithms, John Wiley & Sons, August 12, 2013

[16] Y. Chawla and M. Bhonsle. A Study on Scheduling Methods in Cloud Computing, Pune University, October 2012

[17] J. Blazewicz. Scheduling in Computer and Manufacturing Systems. Springer-Verlag New York,Inc., Secaucus, NJ, USA, 1996.

[18] M. Wieczorek, R. Prodan and T. Fahringer, "Scheduling of scientific workflows in the ASKALON grid environment," SIGMOD Record, 2005.

[19] https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm

[20] Min-You Wu and Wei Shu. Segmented Min-Min: A Static Mapping Algorithm for Meta-tasks on Heterogeneous Computing Systems, University of New Mexico

[21] S. Parsa and R. Entezari-Maleki. RASA: A New Task Scheduling Algorithm in Grid Environment, IDOSI Publications, 2009

[22] N. Chopra, S. Singh. HEFT based Workflow Scheduling Algorithm for Cost Optimization within Deadline in Hybrid Clouds, 4th ICCCNT 2013

[23] B. Cirou, E. Jeannot. Triplet : a Clustering Scheduling Algorithm for Heterogeneous Platforms, HAL Id, 19 May 2006

[24] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," E-Science (e-Science), 2012 IEEE 8th International Conference on. IEEE, 2012.

[25] Y. Huang et al, Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm, Future Generation Computer System (2011)

[26] C. Thiam, G. Da Costa, JM. Pierson. Cooperative Scheduling Anti-load balancing Algorithm for Cloud : CSAAC, 2013 IEEE International Conference on Cloud Computing Technology and Science

[27] Ke Liu, Hai Jin, Jinjun Chen, Xiao Liu,Dong Yuan, Yun Yang. A Compromised-Time-Cost Scheduling Algorithm in SwinDeW-C for Instance-Intensive Cost-Constrained Workflows on Cloud Computing, Huazhong University of Science and Technology

Platform

[28] https://pegasus.isi.edu/

[29] W. Voorsluys, J. Broberg, and R. Buyya. "Introduction to cloud computing." Cloud computing: Principles and para-digms (2011):

[30] Chetto and Delacroix, Scheduling in Real-Time Systems, 1993

[31] A. G. Delavar, M. Javanmard , M. B. Shabestari, M. K. Talebi. RSDC (Reliable Scheduling Distributed in Cloud computing), Department of Computer, Payame Noor Universtiy, 2012