ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνικές Δρομολόγησης Εργασιών σε Σύγχρονα Περιβάλλοντα Υπολογιστικών Νεφών.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αντώνιος Εμμ. Γιαννόπουλος

Επιβλέπων : Νεκτάριος Κοζύρης

Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2014

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνικές Δρομολόγησης Εργασιών σε Σύγχρονα Περιβάλλοντα Υπολογιστικών Νεφών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αντώνιος Εμμ. Γιαννόπουλος

Επιβλέπων : Νεκτάριος Κοζύρης

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 25η Ιουλίου 2014

.....	.....	.....
Νεκτάριος Κοζύρης	Νικόλαος Παπασπύρου	Γεώργιος Γκούμας
Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

.....

Αντώνιος Εμμ. Γιαννόπουλος

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

Περίληψη

Σκοπός αυτής της διπλωματικής εργασίας είναι η διερεύνηση και μελέτη διαφόρων τεχνικών με τις οποίες μπορούμε να αναγνωρίσουμε τις απαιτήσεις μιας υπολογιστικής εργασίας, ώστε να την εντάξουμε κατά τρόπο βέλτιστο σε ένα υπάρχον υπολογιστικό νέφος. Το νέφος αυτό θεωρούμε ότι διαθέτει ετερογενή συστήματα, στα οποία ήδη λειτουργούν άλλες εργασίες, που καταναλώνουν μέρος των διαθέσιμων πόρων. Οι τεχνικές που μελετούμε προσπαθούν να εξασφαλίσουν για την εργασία ότι θα δρομολογηθεί σε ένα τμήμα του υπολογιστικού νέφους που θα είναι επαρκές να καλύψει τις απαιτήσεις της εργασίας, χωρίς δυσμενή επίπτωση στη λειτουργία της, ενώ ταυτόχρονα να εξασφαλίσουν για το υπολογιστικό νέφος ότι η εργασία θα ανατεθεί με τρόπο που θα μεγιστοποιεί την εξοικονόμηση πόρων.

Η λεπτή αυτή ισορροπία απαιτεί, όχι μόνον να εξεταστούν οι απαιτήσεις της εργασίας σε σχέση με τους υπολογιστικούς πόρους που είναι διαθέσιμοι, αλλά να ληφθεί υπόψη και η συμπεριφορά της εργασίας σε σχέση με άλλες εργασίες που ήδη εκτελούνται στο ίδιο υπολογιστικό νέφος και μπορεί να της προκαλέσουν παρεμβολές. Επομένως, πριν την αναζήτηση και δέσμευση υπολογιστικών πόρων, πρέπει να προηγηθεί ένα βήμα αξιολόγησης της εργασίας και των απαιτήσεών της κάτω από διάφορες συνθήκες λειτουργίας. Η αξιολόγηση αυτή μπορεί να γίνει από το ίδιο το σύστημα που διαχειρίζεται τους εικονικούς πόρους, ή από άλλο σύστημα που διενεργεί ανεξάρτητους ελέγχους, πριν τροφοδοτήσει το σύστημα διαχείρισης του νέφους. Για αυτό ακριβώς το βήμα αξιολόγησης, εξετάζουμε διάφορες τεχνικές που έχουν προταθεί, αξιολογούμε την επάρκεια και αποτελεσματικότητά τους, μελετούμε το κόστος λειτουργίας τους και τις συγκρίνουμε ως προς το πραγματικό τους όφελος, τόσο για την απαίτηση (υπολογιστική εργασία) όσο και για το σύστημα εξυπηρέτησης της απαίτησης (υπολογιστικό νέφος).

Λέξεις-Κλειδιά: Υπολογιστικό νέφος, εργασία, κατηγοριοποίηση, ταξινόμηση, αξιολόγηση, δρομολόγηση, ετερογένεια, παρεμβολή.

Abstract

The purpose of this thesis is to investigate and study various techniques by which we can recognize the demands of a computational workload, in order to integrate it optimally into an existing cloud computing infrastructure. We assume that the cloud consists of heterogeneous systems, in which other workloads are already operating and consuming part of the available resources. The techniques under study are intended to ensure that the workload will be assigned to an appropriate section of the cloud computing infrastructure, capable of meeting its requirements, in a manner that will optimize the use of resources and maximize cost savings for the cloud infrastructure.

This delicate balance requires us to examine the demands of a workload against all available resources, while also considering the interference of other workloads already being executed within the same cloud infrastructure. Therefore, before we commit any resources, we need to perform an initial step of classifying the workload. This classification can be performed by the system responsible for managing the cloud's resources, or by a separate system, committed to profiling and scheduling the incoming workloads. For this step we consider various proposed techniques, we evaluate their adequacy and effectiveness, we study their cost and finally we compare them, in order to understand the benefits they offer to the workload itself and to the cloud environment.

Keywords: Could computing, workload, classification, evaluation, profiling scheduling, heterogeneity, interference.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε υπό την επίβλεψη του Λέκτορα Ε.Μ.Π. Γεώργιου Γκούμα, τον οποίο ευχαριστώ θερμά για την βοήθεια και καθοδήγηση κατά τη διάρκεια της συγγραφής. Θα ήθελα επίσης να ευχαριστήσω τον Καθηγητή Ε.Μ.Π. Νεκτάριο Κοζύρη που με εμπιστεύτηκε και μου έδωσε την δυνατότητα να ασχοληθώ με αυτό το πολύ ενδιαφέρον θέμα.

Αφιερώνεται στη μνήμη του πατέρα μου,

Μάνου Γ. Γιαννόπουλου,

τ. Επιμελητή Ε.Μ.Π.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1. Εισαγωγή

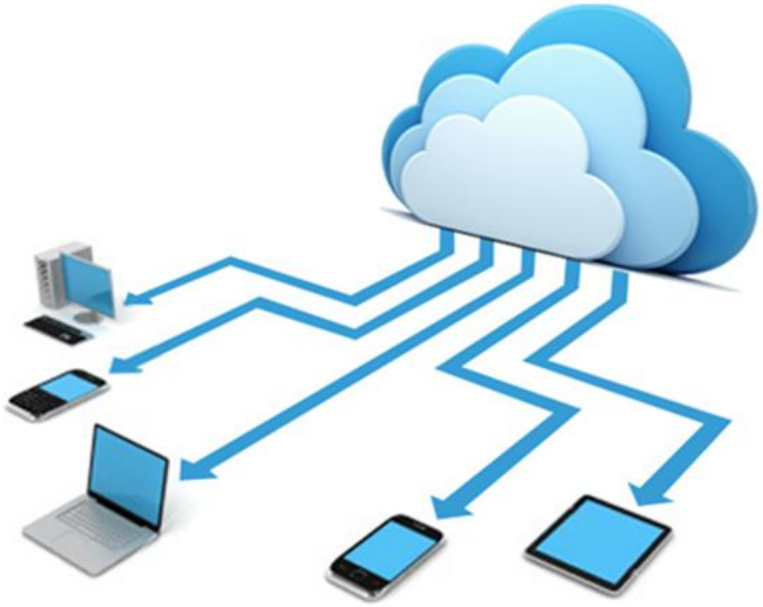
Στο κεφάλαιο αυτό γίνεται μια εισαγωγή στην έννοια των υπηρεσιών νέφους που παρέχονται από σύγχρονα Data Centers, προκειμένου να εξοικειωθεί ο αναγνώστης με τα θέματα που αναλύονται στο κείμενο της εργασίας. Αναφέρονται τα πλεονεκτήματα χρήσης ενός υπολογιστικού νέφους στην εκτέλεση εργασιών, και επισημαίνονται οι αδυναμίες που πρέπει να αντιμετωπιστούν ώστε να προκύπτει αποκλειστικά και μόνον όφελος, τόσο για τους πελάτες των υπηρεσιών, όσο και για τους προμηθευτές (παρόχους υπηρεσιών νέφους).

1.1. Υπηρεσίες Νέφους

Ο όρος "υπολογιστικό νέφος" περιγράφει μια σύγχρονη εξέλιξη που έγινε εφικτή λόγω της σύγκλισης των χώρων της πληροφορικής και των τηλεπικοινωνιών [1]. Ο όρος αναφέρεται σε μεγάλες, συγκεντρωμένες συστοιχίες υπολογιστικών συστημάτων με κοινόχρηστους πόρους (δίκτυα, εξυπηρετητές, μονάδες αποθήκευσης, εφαρμογές και υπηρεσίες). Το περιβάλλον αυτό είναι προσβάσιμο μέσω ταχύτατων τηλεπικοινωνιακών διασυνδέσεων, και είναι ικανό να παραλάβει και να εκτελέσει ταυτόχρονα έναν μεγάλο όγκο υπολογιστικών εργασιών. Η σύγχρονη αυτή εξέλιξη μεταβάλλει δραστικά το μοντέλο χρήσης ηλεκτρονικών υπολογιστών. Όταν προκύπτει μια απαίτηση για υπολογιστικές εργασίες, παύει πλέον να υπάρχει η ανάγκη εγκατάστασης και λειτουργίας μεμονωμένων υπολογιστικών συστημάτων μέσα στον ίδιο φυσικό χώρο. Η απαίτηση μπορεί να μεταφερθεί μέσω τηλεπικοινωνιακών διασυνδέσεων, και ιδίως μέσω του διαδικτύου, σε ένα κεντρικό κόμβο όπου έχει συγκεντρωθεί σημαντική υπολογιστική ισχύς [2]. Εκεί, η εργασία μπορεί να εξυπηρετηθεί με ένα εγγυημένο επίπεδο λειτουργίας, από ένα μοντέλο παροχής υπηρεσιών όπου ο πελάτης θα καταβάλλει ένα τίμημα ανάλογο της χρήσης πόρων που έκανε.

Ο όρος «νέφος» προέκυψε από την γραφική αναπαράσταση του ίδιου του διαδικτύου και των υπολογιστικών συστημάτων που το απαρτίζουν. Συχνά το αναπαριστούμε σε διαγράμματα ως σύννεφο, κάτι που δείχνει ότι δεν μας ενδιαφέρει να δείξουμε λεπτομερώς από ποια συστατικά απαρτίζεται. Απλά «είναι εκεί» και είναι σε θέση να εξυπηρετήσει τις απαιτήσεις μας. Το υπολογιστικό νέφος είναι μια ανάλογη αφηρημένη έννοια, ένα σύνολο υπολογιστικών συστημάτων που είναι ικανό να παραλάβει τις απαιτήσεις μας μέσω τηλεπικοινωνιακών δικτύων, να εκτελέσει τις ζητούμενες εργασίες και να επιστρέψει το αποτέλεσμα μέσω των ίδιων δικτύων. Μερικές φορές μάλιστα οι λεπτομέρειες υλοποίησης του νέφους

παραμένουν σκόπιμα άγνωστες, είτε για λόγους ασφαλείας είτε για λόγους εμπορικού ανταγωνισμού.



Σχήμα 1 - Γραφική αναπαράσταση νέφους που εξυπηρετεί πελάτες με διάφορες συσκευές

Η έννοια της «φιλοξενίας» υπολογιστικών εργασιών είναι διαδεδομένη από τα πρώτα χρόνια αξιοποίησης του διαδικτύου για εφαρμογές. Ως παράδειγμα αναφέρουμε την φιλοξενία web sites σε συστήματα τηλεπικοινωνιακών παρόχων, προκειμένου αυτά να λειτουργούν «πιο κοντά» (δικτυακά) στους καταναλωτές του περιεχομένου τους. Ωστόσο, τα συστήματα αυτά ήταν κατά κύριο λόγο αφιερωμένα σε συγκεκριμένες εργασίες και δεν υπήρχε διαμοιρασμός πόρων ή οικονομία κλίμακας. Η πραγματική αξία του υπολογιστικού νέφους αναδείχθηκε όταν έγινε εφικτή η εικονοποίηση (virtualization) των υποδομών και ο διαμοιρασμός κοινών πόρων σε περισσότερες από μία εργασίες.

Κάθε εργασία έχει συγκεκριμένες απαιτήσεις σε υπολογιστικούς πόρους και εάν εγκατασταθεί σε ένα φυσικό σύστημα που υπερκαλύπτει αυτές τις απαιτήσεις, ένα μέρος των πόρων του συστήματος θα παραμείνει αναξιοποίητο. Είναι σύνηθες για μια εργασία να μην αξιοποιεί το σύνολο των πόρων στο σύνολο των ωρών της ημέρας. Μπορεί να απαιτεί μεγάλο μέρος της μνήμης του και μικρότερο μέρος επεξεργαστικής ισχύος, ενώ μια άλλη εργασία μπορεί να έχει αντίθετες απαιτήσεις. Έτσι οι δύο εργασίες είναι κατάλληλες να συστεγαστούν στο ίδιο σύστημα, χωρίς η μία να παρεμβάλλεται αισθητά στη λειτουργία της άλλης. Οι τεχνικές εικονοποίησης που αναπτύχθηκαν, επέτρεψαν τη δημιουργία εικονικών συστημάτων τα οποία μπορούν να λειτουργούν ταυτόχρονα μέσα σε ένα φυσικό

σύστημα, να μοιράζονται τους πόρους του φυσικού συστήματος και να τους κάνουν διαθέσιμους προς αξιοποίηση σε διάφορες εγκατεστημένες εργασίες. Η χρήση εικονικών συστημάτων δίνει στον τελικό χρήστη μεγάλη ελευθερία στη διαχείριση των πόρων που του ανατίθενται, κρύβοντας παράλληλα τις διεργασίες διαχείρισης των πόρων του φυσικού συστήματος. Όσο πιο μεγάλο είναι το φυσικό σύστημα και όσο πιο πολλούς πόρους διαθέτει, τόσο πιο πιθανό είναι, βάσει των αρχών της στατιστικής, να επιτύχουμε ομαλή κατανομή πολλαπλών εικονικών συστημάτων και εργασιών. Σε περίπτωση δε που ένα φυσικό σύστημα φθάσει στα όριά του, η χρήση των εικονικών συστημάτων μας επιτρέπει την εύκολη μετάπτωση μιας εργασίας σε δεύτερο φυσικό σύστημα, προκειμένου να ελαφρύνουμε την χρήση του πρώτου και να αποφύγουμε την εξάντληση των διαθέσιμων πόρων.

Στον τεχνολογικό λοιπόν κόσμο, οι υπηρεσίες νέφους αναφέρονται στην κατανομή των πόρων ενός μεγάλου υπολογιστικού συστήματος, για την επίτευξη οικονομιών κλίμακας. Μετατρέπουν την πληροφορική σε υπηρεσία με εγγυημένη διαθεσιμότητα και προσφέρουν δυνατότητες άμεσης και απεριόριστης εξυπηρέτησης αιτημάτων. Οι υπηρεσίες παρέχονται κεντρικά, από εταιρίες που χτίζουν σημαντική τεχνογνωσία στον τομέα της λειτουργίας φυσικών και εικονικών συστημάτων και εργασιών. Οι πελάτες δεν χρειάζεται πλέον να επενδύουν σε εξοπλισμό ανάλογο των απαιτήσεών τους, ή σε τεχνικές δεξιότητες που δεν διαθέτουν. Μπορούν να επικεντρωθούν σε αυτό που γνωρίζουν να κάνουν καλύτερα.

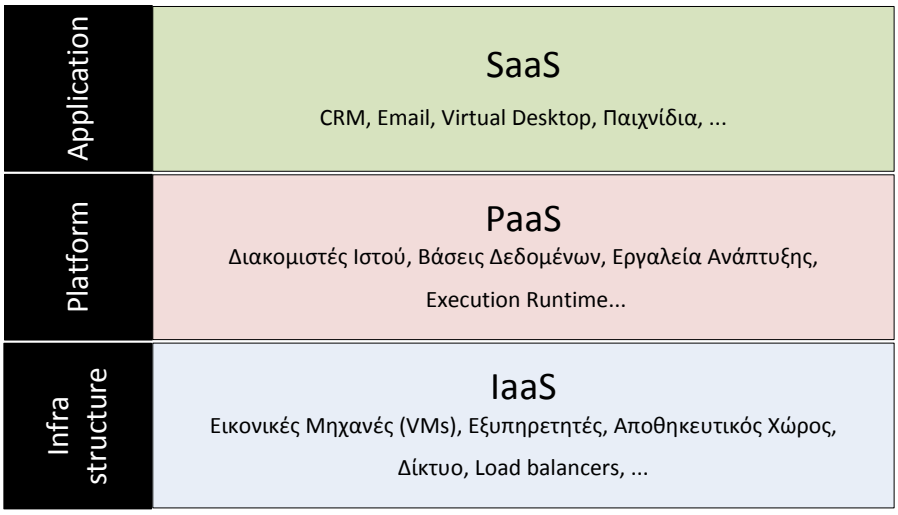
Στον οικονομικό κόσμο, ο όρος «μετακίνηση στο νέφος» αναφέρεται σε έναν οργανισμό που προσπαθεί να απομακρυνθεί από το παραδοσιακό μοντέλο, όπου αγοράζει τον απαραίτητο τεχνολογικό εξοπλισμό με στόχο να τον αποσβέσει σταδιακά μετά από κάποια χρόνια χρήσης (CAPEX). Το νέφος επιτρέπει πλέον στον οργανισμό να επιλέξει ένα νέο οικονομικό μοντέλο, αυτό της αξιοποίησης μιας ήδη διαθέσιμης υποδομής, για την οποία θα πληρώνει μόνο δικαιώματα λειτουργίας και χρήσης (OPEX).

Στον κόσμο των φυσικών προσώπων που χρησιμοποιούν τις υπηρεσίες, το νέφος δεν έχει να κάνει με την κατανομή των πόρων του υλικού, ή με το οικονομικό μοντέλο των προσφερόμενων υπηρεσιών. Αυτό που αντιλαμβάνονται οι χρήστες είναι μια ριζική αλλαγή στο τρόπο που καλούνται να εργαστούν. Τείνουν να εργάζονται όλο και περισσότερο μετακινούμενοι και μέσω διαδικτύου, και λιγότερο σε ένα απομονωμένο σταθερό προσωπικό υπολογιστή. Οι εφαρμογές και τα δεδομένα τους πρέπει να είναι άμεσα διαθέσιμα όπου κι αν βρίσκονται, από όποια συσκευή έχουν διαθέσιμη να χρησιμοποιήσουν τη δεδομένη χρονική στιγμή. Πολλές φορές ο αυστηρός διαχωρισμός του εταιρικού από το κοινωνικό περιβάλλον παύει να υπάρχει. Το ηλεκτρονικό ταχυδρομείο, η κινητή τηλεφωνία, η διείσδυση

του διαδικτύου, η δυνατότητα εικονικών συνδιασκέψεων καθιστούν όλο και πιο μικτό το περιβάλλον στο οποίο λειτουργούν. Ανταλλάσσουν πληροφορίες ταυτόχρονα με μέλη της οικογένειάς τους, με κοινωνικούς εταίρους και με επαγγελματικούς συνεργάτες. Σε αυτό τον κόσμο, οι σύγχρονες υπηρεσίες νέφους έρχονται να παρέχουν πρόσβαση στην αναγκαία πληροφορία και στις εφαρμογές του χρήστη, με διαφανή ασφάλεια, που δεν ενοχλεί ούτε περιορίζει την εμπειρία του χρήστη. Ο χρήστης αντιλαμβάνεται πλέον την παρουσία του νέφους ακόμα και σε απλές καθημερινές του ανάγκες. Για παράδειγμα ο σχετικά περιορισμένος αποθηκευτικός χώρος της κινητής συσκευής του επεκτείνεται στο νέφος, όπου διάφορες λύσεις του επιτρέπουν να αποθηκεύει αρχεία μουσικής ή φωτογραφίας.

1.2. Βασικά μοντέλα υπηρεσιών νέφους

Στον κόσμο των παρόχων υπηρεσιών πληροφορικής και επικοινωνιών, το νέφος αναφέρεται στην δυνατότητά τους να παρέχουν και να πωλούν υπηρεσίες φιλοξενίας εργασιών σε ιδιόκτητες τεχνολογικές υποδομές. Στις υπηρεσίες αυτές δίνονται δημοφιλή ακρωνύμια όπως **SaaS** (Software as a Service) , **PaaS** (Platform as a Service) και **IaaS** (Infrastructure as a Service) [33]. Οι τελικοί πελάτες έχουν πρόσβαση σε υπηρεσίες νέφους από τερματικές συσκευές χαμηλής αξίας, που ήδη διαθέτουν και γνωρίζουν πώς να χρησιμοποιούν. Το επιχειρηματικό λογισμικό και τα δεδομένα τους είναι αποθηκευμένα σε διακομιστές, σε μία ή περισσότερες γεωγραφικά απομακρυσμένες περιοχές, μερικές φορές καλύπτοντας επιπλέον ανάγκες όπως αυτές της υψηλής διαθεσιμότητας και της επιχειρηματικής συνέχειας σε περίπτωση φυσικής καταστροφής. Τα βασικά μοντέλα παροχής των υπηρεσιών νέφους αναπαριστώνται στο Σχήμα 2.



Σχήμα 2 - Τα βασικά μοντέλα παροχής των υπηρεσιών νέφους

Το SaaS (Software as a Service) αναφέρεται στην δυνατότητα του πελάτη να χρησιμοποιεί εφαρμογές του παρόχου, οι οποίες είναι εγκατεστημένες και λειτουργούν σε περιβάλλον νέφους. Ο πελάτης δεν διαχειρίζεται τα συστήματα υποδομής (δίκτυα, εξυπηρετητές, μονάδες αποθήκευσης, εφαρμογές και υπηρεσίες) και συνήθως δεν διαχειρίζεται ούτε τις εφαρμογές που χρησιμοποιεί, εκτός αν του δοθεί περιορισμένη πρόσβαση σε δυνατότητες ή παραμέτρους των εφαρμογών.

Το Platform as a Service (PaaS) αναφέρεται στην δυνατότητα του πελάτη να λειτουργήσει σε περιβάλλον νέφους εφαρμογές που ανέπτυξε ή αγόρασε, και οι οποίες κάνουν χρήση γλωσσών προγραμματισμού, βιβλιοθηκών, υπηρεσιών ή εργαλείων που υποστηρίζει ο πάροχος του νέφους. Ο πελάτης δεν διαχειρίζεται τα συστήματα υποδομής αλλά συνήθως ελέγχει τη διαδικασία εγκατάστασης και λειτουργίας των εφαρμογών του, καθώς και τις ρυθμίσεις που την αφορούν.

Τέλος το Infrastructure as a Service (IaaS) αναφέρεται στη δυνατότητα του πελάτη να αποκτήσει από το νέφος συγκεκριμένους υπολογιστικούς πόρους (επεξεργαστές, μέσα αποθήκευσης δεδομένων, μνήμες κλπ), να εγκαταστήσει λειτουργικά συστήματα και εφαρμογές της επιλογής του, και εν συνεχεία να κάνει χρήση των υπολογιστικών συστημάτων που έχει δημιουργήσει.

1.3. Σύγχρονα Data Centers

Το μοντέλο του νέφους, όπως είναι φυσικό συνέτεινε στη δημιουργία χώρων όπου λειτουργεί μεγάλη υπολογιστική δύναμη. Τα σύγχρονα data centers είναι χώροι όπου φιλοξενούνται υπολογιστικά συστήματα, τηλεπικοινωνιακός εξοπλισμός και συστήματα μεγάλης χωρητικότητας για αποθήκευση δεδομένων. Τα κέντρα αυτά εξασφαλίζουν αδιάλειπτη διαθεσιμότητα ηλεκτρικού ρεύματος με εναλλακτικές μεθόδους (σύνδεση με παρόχους ενέργειας, UPS, μπαταρίες, γεννήτριες ρεύματος). Διαθέτουν επίσης ικανή ψύξη, πυροπροστασία, 24ωρη επίβλεψη από εξειδικευμένο προσωπικό και πολλαπλές δικλείδες ασφαλείας, τόσο στο φυσικό όσο και στο τεχνολογικό επίπεδο. Είναι αυτονόητο ότι οι παροχές αυτές φτάνουν σε επίπεδα που καμία επιχείρηση, πλην ίσως των τραπεζών, δεν είχε την δυνατότητα να δημιουργήσει από μόνη της. Ακόμα λοιπόν κι αν δεν υπήρχε άμεσο όφελος από την κοινή χρήση υπολογιστικών πόρων, το επίπεδο αυτών των υπηρεσιών θα μπορούσε από μόνο του να δικαιολογήσει την μετάβαση στο νέφος, ιδίως για επιχειρηματικά κρίσιμες εφαρμογές [3].

Σήμερα, αρκετές εταιρίες δραστηριοποιούνται στο χώρο της παροχής υπηρεσιών νέφους. Ονόματα με παγκόσμια εμβέλεια όπως το Amazon Elastic Compute Cloud (EC2) [4] , Microsoft Windows Azure [5], Rackspace [6], IBM SoftLayer [7] και Google

Compute Engine [8] εξυπηρετούν δεκάδες χιλιάδες εργασίες σε καθημερινή βάση, σε ιδιόκτητα data centers που είναι γεωγραφικά κατανεμημένα. Χαρακτηριστικό των εταιριών αυτών είναι η τεράστια επένδυση σε υλικό, αλλά και η συνεχής ανάπτυξη και αναβάθμιση της πλατφόρμας διαχείρισης του υλικού, που επιτρέπει στις εταιρίες αυτές να αξιοποιούν την επένδυσή τους στο μέγιστο.

Οι τεχνολογίες νέφους αναπτύχθηκαν σε τέτοιο βαθμό και προσφέρουν τόσα οφέλη, ώστε πλέον ακόμα και εταιρίες που δεν επιθυμούν να μεταβούν άμεσα σε δημόσιο νέφος, οργανώνουν την IT υποδομή τους ως ιδιωτικό νέφος (private cloud) [9], [10], χρησιμοποιώντας συστήματα διαχείρισης της υποδομής όπως το VMWare vShpere [11], Citrix XenServer [12], Microsoft Hyper-V [13], OpenStack [23] και Mesos [43]. Με τον τρόπο αυτό απολαμβάνουν ορισμένα από τα οφέλη της ενοποίησης των διαθέσιμων πόρων και της κοινής τους χρήσης από ετερόκλητες εργασίες.

Μερικές φορές ένα ιδιωτικό νέφος καλείται να αντιμετωπίσει έκτακτες ανάγκες όπως αύξηση των εργασιών πέρα των δυνατοτήτων του. Σε αυτή τη περίπτωση μπορεί εύκολα να επεκταθεί σε ένα δημόσιο νέφος και να αξιοποιήσει μέρος της υπολογιστική ισχύος του. Η διάταξη αυτή συχνά αναφέρεται ως υβριδικό νέφος (hybrid cloud) [33].

1.4. Οφέλη των υπηρεσιών νέφους

Παρακάτω αναφέρουμε επιγραμματικά τα οφέλη που αποκομίζουν οι χρήστες ή πελάτες από την χρήση των υπηρεσιών νέφους:

- Άμεση διαθεσιμότητα εξοπλισμού, σε αντίθεση με την αναμονή που προκαλεί μια συνήθης διαδικασία προμήθειας και εγκατάστασης.
- Δυνατότητα άμεσης κλιμάκωσης και επέκτασης μόλις παραστεί ανάγκη. Εξάλειψη της ανάγκης για μακροπρόθεσμο σχεδιασμό. Μόλις μια εταιρία αντιμετωπίσει αυξημένες ανάγκες, αποκτά άμεσα μέσω του νέφους πρόσβαση στο επιπλέον υλικό που είναι απαραίτητο για να την εξυπηρετήσει.
- Μείωση ή εξάλειψη του κόστους αγοράς εξοπλισμού καθώς και της ανάγκης για διαρκείς αναβαθμίσεις και του άγχους παρακολούθησης των τεχνολογικών εξελίξεων.
- Μείωση ή εξάλειψη της ανάγκης για προμήθεια, εγκατάσταση και συντήρηση λογισμικού και αδειών χρήσης. Πολλές φορές το νέφος παρέχει το αναγκαίο λογισμικό, ενώ οι άδειες χρήσης συμπεριλαμβάνονται στη τιμή της υπηρεσίας. Η αναβάθμιση και επέκταση των δυνατοτήτων του λογισμικού γίνεται από τον πάροχο των υπηρεσιών νέφους και καθίσταται αυτόματα διαθέσιμη στους χρήστες, χωρίς ανάγκη εγκατάστασης.

- Απόλυτος έλεγχος του κόστους. Συνήθως οι υπηρεσίες νέφους παρέχονται με μονάδες χρήσης (ανά λεπτό χρήσης ενός Core, ανά megabyte χρήσης αποθηκευτικού χώρου κλπ) με αποτέλεσμα οι χρήστες να μην χρεώνονται όταν δεν κάνουν χρήση των υπηρεσιών.
- Πρόσβαση στις εφαρμογές και στα δεδομένα από οποιοδήποτε σημείο και οποιαδήποτε συσκευή. Δυνατότητα απομακρυσμένης συνεργασίας με συναδέλφους ή συνεργάτες, πάνω σε κοινή βάση εγγράφων και δεδομένων.
- Αποφυγή άσκοπου κόστους όταν η ανάγκη χρήσης υπολογιστικών συστημάτων αφορά μικρό χρονικό διάστημα, όπως για παράδειγμα η ανάγκη μεγάλης υπολογιστικής ισχύος σε μια ημέρα εκλογών, ή σε άλλες έκτακτες εργασίες πεπερασμένης διάρκειας (user acceptance testing, περίοδος εγγραφών ή αυξημένων πωλήσεων, περίοδος κατάθεσης φορολογικών δηλώσεων κλπ)
- Εξάλειψη της ανάγκης κατασκευής τεχνολογικών υποδομών όπως data room, συστήματος ψύξης, αδιάλειπτης παροχής ρεύματος καθώς και ύπαρξης εξειδικευμένων μηχανικών για την λειτουργία και συντήρηση όλων αυτών.
- Απελευθέρωση επενδύσεων, καθώς οι τεχνολογικές ανάγκες που θα καλύπτονταν με CAPEX μετατρέπονται σε OPEX.
- Αποδέσμευση από μακροχρόνιες συνεργασίες. Οι πελάτες μπορούν να καλύπτουν τις ανάγκες τους με συνεργασίες που συνάπτουν ανά μήνα, ανά μέρα ή ακόμα και ανά ώρα.
- Υψηλή διαθεσιμότητα δεδομένων και εφαρμογών, γεωγραφική διασπορά που εγγυάται την επιχειρηματική συνέχεια σε περίπτωση φυσικής καταστροφής.
- Εγγυημένη διαθεσιμότητα και επιστροφή χρημάτων σε περίπτωση που η λειτουργία υπολείπεται του συμφωνημένου επιπέδου.
- Τυποποιημένη διαδικασία τήρησης εφεδρικών αντιγράφων και επανάκτησης χαμένων δεδομένων.
- Παράλληλη επεξεργασία μεγάλου όγκου δεδομένων και άμεση αξιοποίηση των αποτελεσμάτων (σε πραγματικό χρόνο) από μια επιχείρηση, όπως επιτάσσει η άλλη μεγάλη τάση της εποχής, τα Big Data.
- Οικονομίες κλίμακας για τα data centers, καθώς προβαίνουν σε μαζικές παραγγελίες που τους επιτρέπουν να αγοράζουν υλικό και υπηρεσίες σε σημαντικά χαμηλότερο κόστος. Υπολογίζεται για παράδειγμα ότι ένα μεγάλο data center μπορεί να προμηθευτεί ενέργεια με κόστος από 1/5 έως και 1/7 της συμβατικής αξίας [14].

Ωστόσο υπάρχουν και μια σειρά από αδυναμίες και ανησυχίες, που πρέπει να αντιμετωπίζονται επιτυχώς από τους παρόχους ώστε οι πελάτες να απολαμβάνουν μόνο οφέλη.

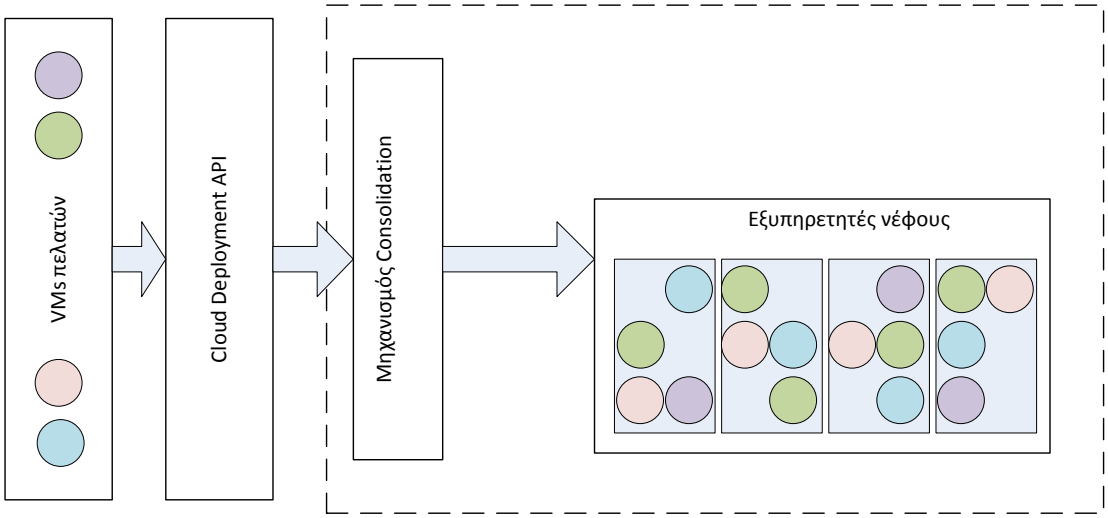
- Το θέμα της ιδιωτικότητας και ασφάλειας, λόγω του γεγονότος ότι τα δεδομένα φιλοξενούνται σε κοινόχρηστα συστήματα. Υπάρχουν κίνδυνοι βιομηχανικής

κατασκοπείας και σε ορισμένες περιπτώσεις η ίδια η ύπαρξη μιας εταιρίας μπορεί να τεθεί σε κίνδυνο, αν υποκλαπούν ή χαθούν τα δεδομένα της. Είναι σημαντικό οι εταιρίες να επιλέγουν παρόχους με αποδεδειγμένη αξιοπιστία, ενώ από την πλευρά των παρόχων πρέπει να αξιοποιούνται οι πλέον σύγχρονες τεχνικές ασφάλειας και κρυπτογράφησης για προστασία των δεδομένων και εφαρμογών των πελατών. Οι συχνές αναφορές για παραβιάσεις δεδομένων, για μη-εξουσιοδοτημένη απόκτηση μυστικών κωδικών και στοιχείων πιστωτικών καρτών, καθώς και προβλήματα σε διαδεδομένα πρωτόκολλα όπως η πρόσφατη παραβίαση του OpenSSL, αποδεικνύουν ότι αυτή είναι μια μάχη που πρέπει να δίνεται διαρκώς. Πολλές φορές λέγεται ότι η μεγαλύτερη αξία ενός παρόχου νέφους δεν είναι το μέγεθος της επένδυσης που έκανε, αλλά το μέγεθος της αξιοπιστίας του.

- Η εξάρτηση από έναν προμηθευτή. Αυτός είναι ο βασικός λόγος που έχει αρχίσει να αναπτύσσεται μια νέα οντότητα, ο *cloud broker*. Ο ρόλος αυτός αναμένεται να καλυφθεί κυρίως από παρόχους τηλεπικοινωνιακών υπηρεσιών, οι οποίοι κατέχουν ταχύτατα τηλεπικοινωνιακά κυκλώματα και μπορούν να καλύψουν τις ανάγκες των πελατών τους κατευθύνοντάς τους σε περισσότερους του ενός παρόχους νέφους. Συνήθως αυτό σημαίνει ότι υλοποιούν ένα επιπλέον επίπεδο απόκτησης και διαχείρισης υπηρεσιών νέφους, το οποίο είναι αγνωστικό ως προς τις τεχνικές λεπτομέρειες του κάθε προμηθευτή υπηρεσιών νέφους.
- Αυξημένη εξάρτηση από τη διαθεσιμότητα τηλεπικοινωνιακών κυκλωμάτων. Προκειμένου να την αντιμετωπίσουν, οι οργανισμοί καταφεύγουν σε λύσεις αυξημένης διαθεσιμότητας, όπως διπλές οδεύσεις οπτικών ινών ή ακόμα και χρήση πολλαπλών τηλεπικοινωνιακών παρόχων.
- Νομικά και ρυθμιστικά προβλήματα που προκύπτουν για την ιδιοκτησία, τα πνευματικά δικαιώματα και τα δικαιώματα χρήσης του περιεχομένου που αποθηκεύεται στο νέφος. Ένα παράδειγμα αμφιλεγόμενης εξέλιξης, είναι η δήλωση της Google ότι αξιοποιεί τα δεδομένα των χρηστών της για στοχευμένες διαφημιστικές υπηρεσίες. Ο χρήστης θα πρέπει να εξετάζει λεπτομερώς τους όρους χρήσης των υπηρεσιών νέφους, και να επιλέγει προμηθευτή λαμβάνοντας υπόψη αυτό το κριτήριο.

1.5. Αρχιτεκτονική υπηρεσιών νέφους

Σε μια τυπική αρχιτεκτονική υπηρεσιών νέφους, όπως στο Σχήμα 3, οι πελάτες υποβάλλουν εικονικές μηχανές προς εγκατάσταση μέσω ενός Cloud Deployment API. Ένας μηχανισμός Consolidation αναγνωρίζει τις απαιτήσεις των εργασιών και αναλαμβάνει να τις εγκαταστήσει σε τμήμα του νέφους που είναι επαρκές να τις εξυπηρετήσει, χωρίς να παραβιαστούν οι απαιτήσεις που του έχουν τεθεί.



Σχήμα 3 - Βασική αρχιτεκτονική υπηρεσιών νέφους

Οι απαιτήσεις αυτές προέρχονται τόσο από τον πελάτη (προδιαγραφές της εργασίας) όσο και από τον πάροχο των υπηρεσιών (αποδοτική λειτουργία πλατφόρμας). Το βασικό πρόβλημα ενός μοντέλου όπου ο πελάτης θέτει τις απαιτήσεις και ζητά τη δέσμευση ανάλογων πόρων, είναι ότι συνήθως υπερεκτιμά τις απαιτήσεις της εργασίας του προκειμένου να μην αντιμετωπίσει πρόβλημα. Επίσης, δεν είναι εύκολο να συνυπολογίσει τις δυναμικές ενός φυσικού συστήματος που διαμοιράζει τους πόρους του σε εικονικές μηχανές.

Το πρόβλημα της εγκατάστασης των εργασιών κατά τρόπο αποδοτικό, γίνεται όλο και πιο σύνθετο καθώς οι εργασίες πολλαπλασιάζονται, η φύση τους ποικίλλει (από μικρές single-process εφαρμογές έως μεγάλες, multi-tier υπηρεσίες) και η κάθε μια θέτει τις δικές της απαιτήσεις που αναφέρονται τόσο σε πόρους που είναι απαραίτητοι για την εκτέλεση της εργασίας, όσο και σε επιδόσεις, όπως το επιθυμητό QoS, ή ο επιθυμητός χρόνος εκτέλεσης [2].

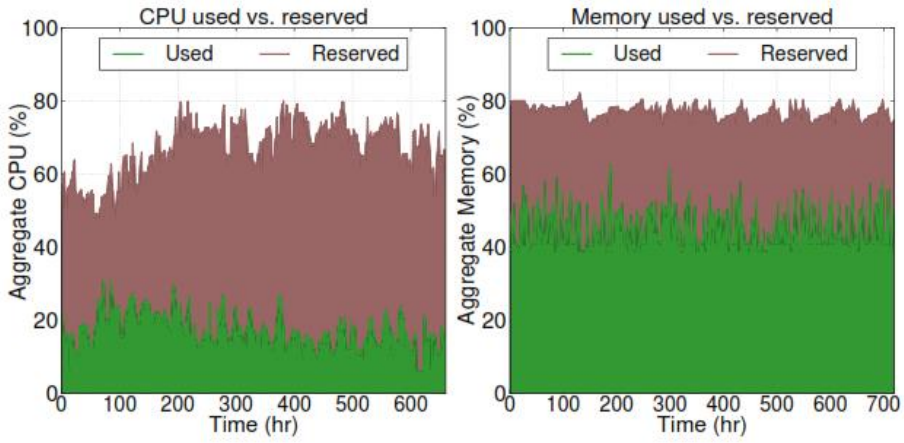
Ο Μηχανισμός Consolidation γνωρίζει ανά πάσα στιγμή την εικόνα των εργασιών που είναι εγκατεστημένες, και των δυνατοτήτων και της κατάστασης λειτουργίας των εξυπηρετητών του νέφους. Επιτηρεί τη λειτουργία του νέφους και παίρνει αποφάσεις για την κατάλληλη δρομολόγηση νέων εργασιών. Ουσιαστικά ένας τέτοιος μηχανισμός καλείται να λάβει δύο αποφάσεις: Αρχικά να υπολογίσει τη σωστή ποσότητα πόρων για μια νέα εργασία (*resource allocation*), και ακολούθως να επιλέξει τη σωστή συστοιχία εξυπηρετητών η οποία μπορεί να διαθέσει τους ζητούμενους πόρους (*resource assignment*). Στη συνέχεια αυτής της εργασίας θα εξετάσουμε διάφορες υλοποιήσεις αυτού του Μηχανισμού Consolidation και θα σχολιάσουμε την αποδοτικότητα κάθε υλοποίησης.

2. Αποτελεσματική διαχείριση σύγχρονων Data Centers

Στο κεφάλαιο αυτό επισημαίνουμε τα βασικά προβλήματα που αντιμετωπίζει ένα σύγχρονο data center, στην προσπάθεια να αξιοποιήσει τις υποδομές του κατά τρόπο βέλτιστο και αποτελεσματικό. Τα προβλήματα αυτά θα πρέπει να λαμβάνει υπόψη του κάθε εξελιγμένος μηχανισμός δρομολόγησης εργασιών.

2.1. Το πρόβλημα της υποχρησιμοποίησης πόρων

Σε πολλά σύγχρονα data centers η χρήση των πόρων κυμαίνεται σε χαμηλά επίπεδα. Σε ανάλογες έρευνες έχουν καταγραφεί μέσοι όροι αξιοποίησης της τάξης του 5% έως 17% [15], [46], [47], [48], [49]. Το πρόβλημα εμφανίζεται ακόμα και σε συμπαγείς εγκαταστάσεις όπου έχει γίνει προσεκτική μελέτη πριν τη δέσμευση πόρων. Στο Σχήμα 4 δείχνουμε το πραγματικό ποσοστό χρήσης δυο πόρων (CPU και μνήμης), που δεσμεύτηκαν σε μια παραγωγική συστοιχία εξυπηρετητών του Twitter, στη διάρκεια 30 ημερών [44].



Σχήμα 4 - Ποσοστό χρήσης δεσμευμένων πόρων σε ένα τυπικό σύστημα παραγωγής

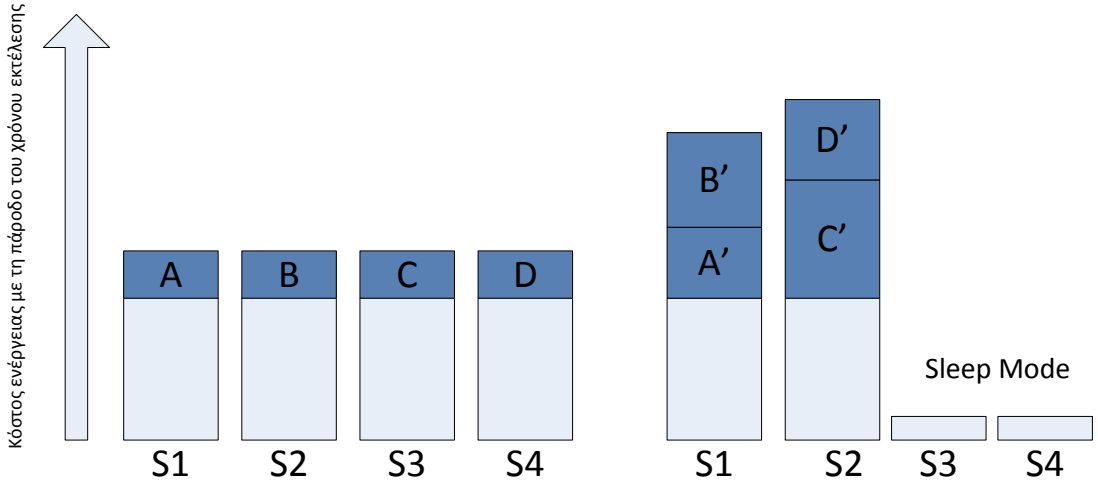
Το μέσο ποσοστό χρήσης της επεξεργαστικής ισχύος είναι κάτω του 20% ενώ η δέσμευση πόρων έφτανε στο 80% των διαθέσιμων. Το μέσο ποσοστό χρήσης της μνήμης είναι υψηλότερο, της τάξης του 40% έως 50%, αλλά και πάλι υπολείπεται σημαντικά των πόρων που δεσμεύτηκαν για τις συγκεκριμένες εργασίες. Παρόμοια μελέτη σε συστοιχία 12.000 Google Servers δείχνει μέσο όρο χρήσης CPU στο 25-35% και μνήμης στο 40%, από εργασίες που είχαν δεσμεύσει το 75% και 60% των αντίστοιχων πόρων [45]. Μάλιστα, σε σχέση με άλλες υλοποιήσεις, οι συστοιχίες των Twitter και Google επιτυγχάνουν σχετικά αυξημένο ποσοστό χρήσης των

διαθέσιμων πόρων. Αυτό οφείλεται στην χρήση εξελιγμένων συστημάτων διαχείρισης νέφους (Mesos και Borg αντίστοιχα).

Πέρα από το γεγονός ότι δεν αξιοποιείται μια πολύ σημαντική επένδυση σε υλικό, ένα επιπλέον πρόβλημα είναι η μη-γραμμική κατανάλωση ισχύος. Ένας σύγχρονος εξυπηρετητής που παραμένει άεργος, καταναλώνει πάνω από το 50% της ισχύος που θα κατανάλωνε σε πλήρη χρήση, ενώ μελλοντικά αναμένεται το ποσοστό να πέσει στο 30% [16]. Πολλοί εξυπηρετητές που λειτουργούν με χαμηλό φόρτο, καταναλώνουν σημαντικά περισσότερη ενέργεια σε σχέση με λιγότερους εξυπηρετητές που λειτουργούν σε κατάσταση υψηλού φόρτου [17], [18], [19]. Παράλληλα, η χρήση περισσότερων εξυπηρετητών συνεπάγεται επιπλέον σπατάλη σε χώρο, ψύξη, καλωδίωση, υπηρεσίες συντήρησης και παρακολούθησης της λειτουργίας.

Στο παραπάνω πρόβλημα, μια προφανής λύση είναι η ενοποίηση των πόρων και η μεταφορά και συστέγαση εργασιών σε λιγότερους εξυπηρετητές, που πλέον θα λειτουργούν με σημαντικά υψηλότερο φόρτο. Οι υπόλοιποι εξυπηρετητές μπορούν να σβήσουν μέχρι να παρουσιαστεί ανάγκη χρήσης τους, κάτι που δεν λύνει το πρόβλημα της αρχικής επένδυσης, αλλά σαφέστατα μειώνει το κόστος λειτουργίας του data center.

Στο Σχήμα 5 δείχνουμε μια ανάλογη κατάσταση. Στα αριστερά, 4 εικονικές μηχανές (A, B, C, D) τοποθετούνται σε 4 ξεχωριστούς εξυπηρετητές. Η έντονα γραμμοσκιασμένη περιοχή δείχνει την ενέργεια που καταναλώνεται σε 1 ώρα χρήσης μιας εργασίας που εκτελείται στην κάθε εικονική μηχανή, ενώ η υπόλοιπη γραμμοσκιασμένη περιοχή δείχνει την ενέργεια που καταναλώνεται μόνο και μόνο επειδή η φυσική μηχανή είναι σε λειτουργία. Η ενέργεια αυτή θα καταναλώνονταν ακόμα κι αν η εικονική μηχανή δεν λειτουργούσε.



Σχήμα 5 - Μεταβολή του κόστους της ενέργειας, λόγω της ενοποίησης εργασιών

Στα δεξιά δείχνουμε την διαφορά, όταν οι εικονικές μηχανές C και D έχουν συστεγαστεί με τις A και B αντίστοιχα, ενώ 2 φυσικοί εξυπηρετητές έχουν τεθεί σε κατάσταση sleep.

Ως προς την καταναλισκόμενη ενέργεια, η δεύτερη συστοιχία είναι πιο αποδοτική. Ωστόσο παρατηρούμε αύξηση του χρόνου εκτέλεσης των εργασιών, λόγω της αρνητικής αλληλεπίδρασης της μιας εργασίας στην άλλη, εξαιτίας της παράλληλης λειτουργίας των εργασιών και της χρήσης κοινών πόρων. Κάτω από ακραίες συνθήκες, η αλληλεπίδραση αυτή, που ονομάζουμε *παρεμβολή*, μπορεί να ακυρώσει τα οφέλη της συστέγασης. Μπορεί να καταναλωθεί τελικά περισσότερη ενέργεια, λόγω του επιπλέον χρόνου που απαιτείται για την εκτέλεση των εργασιών, ή μπορεί να παραβιαστούν οι απαιτήσεις ποιότητας (QoS) των εργασιών.

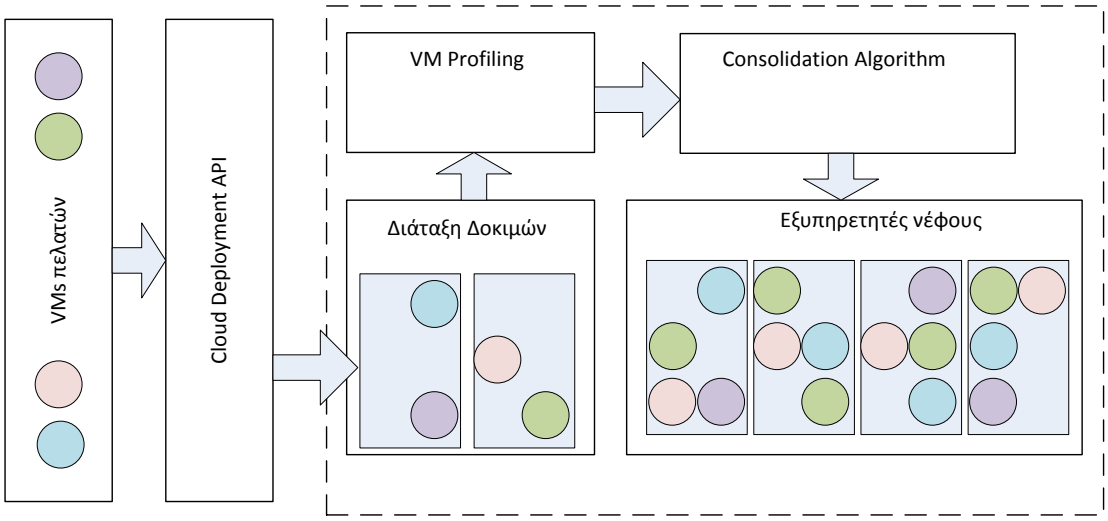
Το γεγονός της πτώσης της ποιότητας ή των επιδόσεων μιας εργασίας λόγω της συστέγασής της με άλλες, δεν μπορεί να αγνοηθεί. Σύμφωνα με μελέτες σε επιχειρηματικά κρίσιμες εργασίες [37], [38], [39], καθυστερήσεις που οφείλονταν σε συστέγαση εργασιών οδήγησαν σε απώλειες της τάξης του 1% έως 20% στα αναμενόμενα έσοδα από τις εργασίες αυτές. Προκειμένου να αποφύγουν αυτό το πρόβλημα, αρκετοί πάροχοι εφαρμόζουν συντηρητικά όρια στη χρήση των κοινόχρηστων πόρων. Η Google για παράδειγμα δρομολογεί εργασίες με κανόνα η χρήση των processor cores να μην ξεπερνά το ποσοστό του 50% [21].

2.2. Το πρόβλημα των παρεμβολών

Νωρίτερα αναφερθήκαμε στις παρεμβολές, την αρνητική δηλαδή επίπτωση που αρχίζει να εμφανίζεται μεταξύ εικονικών μηχανών και εργασιών όταν αυτές συστεγαστούν στην ίδια φυσική μηχανή και μοιραστούν κοινούς πόρους. Ακόμα και εάν αφιερώσουμε ξεχωριστά cores σε κάθε εργασία, η κοινή χρήση καναλιών μνήμης, αποθηκευτικού χώρου και δικτυακών συσκευών, θα συμβάλλουν στο φαινόμενο των παρεμβολών [20], [21], [22].

Το πρόβλημα της υποβάθμισης της επίδοσης μιας εικονικής μηχανής, όταν αυτή συστεγάζεται με άλλες εικονικές μηχανές στο ίδιο σύστημα, έχει αναλυθεί και τεκμηριωθεί επαρκώς και έχουν προταθεί διάφορες μέθοδοι αντιμετώπισης του φαινομένου [34], [35], [36], [50]. Μια από αυτές, είναι η αναγνώριση της αρνητικής επίδρασης των παρεμβολών εκ των υστέρων, και η ανάθεση επιπλέον υπολογιστικών πόρων στο σύστημα προκειμένου να αντισταθμιστεί η παρατηρούμενη πτώση της επίδοσης [22]. Συνήθως όμως προκρίνεται μια προληπτική μέθοδος, που εντοπίζει εργασίες που αλληλεπιδρούν ελάχιστα μεταξύ τους, επομένως είναι κατάλληλες για συστέγαση.

Το πρόβλημα της εξεύρεσης αυτών των εργασιών ανήκει στη κατηγορία “NP-Complete” της θεωρίας πολυπλοκότητας. Είναι εύκολο να ορίσουμε έναν αλγόριθμο επίλυσης, αλλά αυτός είναι εκθετικός. Καθώς μεγαλώνουν τα στιγμιότυπα, καθίσταται δύσκολο και σταδιακά αδύνατο να λύσουμε το πρόβλημα σε αποδεκτό χρόνο. Το πρόβλημα αυτό συνήθως επιλύεται με προσεγγιστικές τεχνικές που δεν απαιτούν μέτρηση της αλληλεπίδρασης όλων των πιθανών συνδυασμών εικονικών μηχανών μεταξύ τους [20], [21], [24]. Πρακτικά για κάθε εικονική μηχανή δημιουργείται ένα προφίλ το οποίο συγκρίνεται με άλλα, ήδη γνωστά προφίλ, προκειμένου να προβλεφθεί η αναμενόμενη υποβάθμιση της επίδοσης της συγκεκριμένης μηχανής όταν την συστεγάσουμε με μηχανές που ανήκουν σε ίδιες ή διαφορετικές κατηγορίες. Η τεχνική αυτή αποτυπώνεται στο Σχήμα 6.



Σχήμα 6 - Διαδικασία Profiling μιας εργασίας, πριν τη συστέγασή της στο νέφος

Το ποσοστό επιτυχίας μιας τέτοιας πρόβλεψης είναι αρκετά υψηλό. Σε πραγματικές συνθήκες λειτουργίας με ένα τυπικό μηχανισμό αναγνώρισης του προφίλ, έχει μετρηθεί απόκλιση μόλις 5% έως 10% μεταξύ εκτιμώμενης και πραγματικής επίδοσης.

Σύμφωνα με την τυπική αρχιτεκτονική ενός νέφους, οι πελάτες αποστέλλουν εικονικές μηχανές προς εγκατάσταση, μέσω ενός κατάλληλου cloud API. Ιδανικά μια διαδικασία εγκατάστασης των εικονικών μηχανών θα έπρεπε να λάβει μια απόφαση στιγμιαία, για την βέλτιστη τοποθέτηση της νέας μηχανής μέσα στο νέφος όπου ήδη λειτουργούν άλλες εικονικές μηχανές.

Καθώς δεν υπάρχει τρόπος να γνωρίζουμε εκ των προτέρων ποια θα είναι η βέλτιστη τοποθέτηση, αρχικά τοποθετούμε τη μηχανή σε μια συστοιχία δοκιμών (συχνά ονομάζεται Profiling, Training ή Testing Configuration). Σκοπός αυτής της συστοιχίας είναι να αναγνωρίσει το προφίλ της εισερχόμενης εργασίας και να την κατατάξει σε κάποια από τις γνωστές κατηγορίες. Εκτελούνται δοκιμές αναγνώρισης του προφίλ [20], [21] που χαρακτηρίζουν τη μηχανή παράγοντας μια σειρά από παραμέτρους, που δείχνουν τι επίπτωση θα δεχθεί αλλά και θα προκαλέσει η εικονική μηχανή όταν συστεγαστεί με άλλες.

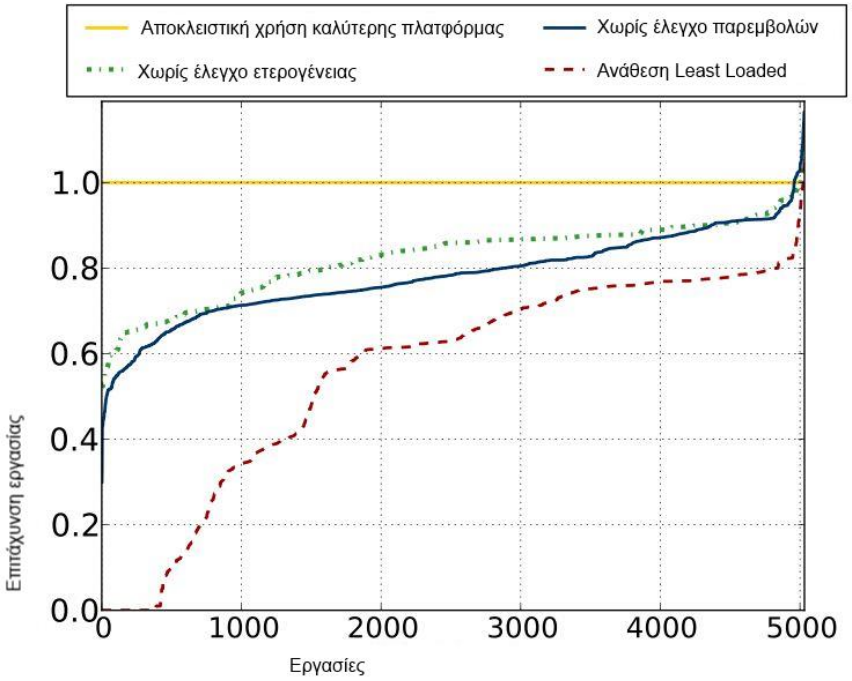
Στο τέλος αυτής της διαδικασίας χρησιμοποιείται ένας αλγόριθμος (Consolidation Algorithm) για την βέλτιστη τοποθέτηση της μηχανής σε μια από τις συστοιχίες των εξυπηρετητών του νέφους. Ο αλγόριθμος τροφοδοτείται από τις παραμέτρους που αναγνωρίσαμε στο προηγούμενο στάδιο. Καθώς λαμβάνει υπόψη τις παρεμβολές μεταξύ εικονικών μηχανών, λειτουργεί αποδοτικότερα από άλλους αλγορίθμους που θα επιχειρούσαν την τοποθέτηση της μηχανής τυχαία, ή με βάση τις ονομαστικές απαιτήσεις της**.**

2.3. Το πρόβλημα της ετερογένειας στις πλατφόρμες

Τα σύγχρονα data centers αναπτύχθηκαν στη διάρκεια των τελευταίων 15 χρόνων, μέσω μιας σταδιακής διαδικασίας εγκατάστασης συστημάτων και αντικατάστασής τους μόλις ξεπεραστεί η διάρκεια ζωής τους [25], [26], [27]. Ως αποτέλεσμα αυτής της διαδικασίας, ένα data center περιλαμβάνει από 3 έως 5 διαφορετικές γενιές συστημάτων. Όμως ακόμη και τα συστήματα της ίδιας γενιάς, έχουν διαφορετικά χαρακτηριστικά ως προς τις ταχύτητες και τις δυνατότητες των επεξεργαστών, μνημών, δίσκων αποθήκευσης και υποσυστημάτων δικτύωσης. Είναι πολύ συνηθισμένο να συναντούμε από 10 έως και 50 διαφορετικά προφίλ συστημάτων στο ίδιο data center. Αν αγνοήσουμε αυτό το γεγονός, ενδέχεται να επηρεάσουμε αρνητικά κάποιες εργασίες που εξαρτώνται πολύ από τη διαμόρφωση του υλικού στο οποίο θα εγκατασταθούν.

Μια εργασία που λειτουργεί ικανοποιητικά σε συστοιχία υψηλών προδιαγραφών, ενδέχεται να λειτουργεί ακόμα καλύτερα εάν δρομολογηθεί σε συστοιχία με χαμηλότερες προδιαγραφές αλλά περισσότερους σε πλήθος εξυπηρετητές. Μπορεί επίσης να λειτουργεί αποδοτικά σε εξυπηρετητές με παλαιότερη τεχνολογία επεξεργαστών, αν η απόδοσή της βασίζεται κυρίως στη ποσότητα μνήμης που δεσμεύεται γι αυτήν, και λιγότερο στην επεξεργαστική ισχύ. Τέτοια παραδείγματα δείχνουν ότι η επιλογή ανάμεσα σε εξυπηρετητές διαφορετικής διαμόρφωσης μπορεί να επηρεάσει την απόδοση μιας εργασίας.

Στο Σχήμα 7 βλέπουμε πραγματικές μετρήσεις από τη χρήση 5000 διαφορετικών εργασιών (benchmark workloads) [28]. Ένας Μηχανισμός Consolidation που δεν λαμβάνει υπόψη του την ετερογένεια, επιβραδύνει κατά μέσο όρο 22% την λειτουργία των εργασιών.



Σχήμα 7 - Επιβράδυνση εργασιών όταν δεν λαμβάνεται υπόψη η ετερογένεια και οι παρεμβολές

Στο ίδιο σχήμα βλέπουμε την επίπτωση ενός μηχανισμού που δεν λαμβάνει υπόψη του τις παρεμβολές και επιβαρύνει τις εργασίες με 34% κατά μέσο όρο. Τέλος, βλέπουμε έναν κοινό αλγόριθμο (Least Loaded) που δεν λαμβάνει υπόψη ούτε την ετερογένεια ούτε τις παρεμβολές και επιβαρύνει τις εργασίες με 48% κατά μέσο όρο, ενώ είναι εμφανές ότι μερικές από αυτές δεν ολοκληρώνονται καν.

Η τοποθέτηση των 5000 εργασιών στο διάγραμμα έχει γίνει από αυτές που επηρεάζονται περισσότερο (αριστερά), προς αυτές που επηρεάζονται λιγότερο ή καθόλου (δεξιά).

Μια επιβράδυνση στην λειτουργία μιας εργασίας επηρεάζει τον πελάτη που αντιλαμβάνεται την καθυστέρηση στην εκτέλεσή της, αλλά ταυτόχρονα επηρεάζει και το data center καθώς καταναλώνει πόρους για μεγαλύτερο χρονικό διάστημα.

2.4. Το πρόβλημα της κλιμάκωσης

Συνήθως η απόφαση για την ποσότητα των πόρων ανά εξυπηρετητή που θα διατεθούν σε μια εργασία, όπως επίσης για τον αριθμό των εξυπηρετητών ανά εργασία, αφήνεται στον πελάτη. Ο ίδιος θέτει τις απαιτήσεις σε πόρους και ο Μηχανισμός Consolidation φροντίζει να του εκχωρηθούν χωρίς να εξετάζει αν η πρόβλεψη είναι επιτυχής ή έχει γίνει υπερεκτίμηση ή υποεκτίμηση. Αυτό οδηγεί σε μη-βέλτιστη χρήση των υποδομών του νέφους. Στην περίπτωση της υπερεκτίμησης, ο πελάτης δεσμεύει και πληρώνει για πόρους που δεν αξιοποιεί. Στην περίπτωση της υποεκτίμησης, η εργασία θα αντιμετωπίσει πρόβλημα λειτουργίας καθώς οι πόροι που έχουν δεσμευτεί γι αυτήν είναι ανεπαρκείς, ενώ ενδεχομένως υπάρχει επάρκεια πόρων στο νέφος.

Μια λύση στο παραπάνω πρόβλημα, είναι η χρήση συστημάτων παρακολούθησης της επίδοσης των εργασιών, με απώτερο στόχο να επανεκτιμώνται οι δρομολογήσεις των εργασιών και να διορθώνονται τυχόν σφάλματα. Η τακτική αυτή έχει δύο μειονεκτήματα: Πρώτον ότι διατίθενται επιπλέον πόροι για τη λειτουργία συστημάτων παρακολούθησης, και δεύτερον ότι το πρόβλημα επιδιορθώνεται αφού συμβεί και αφού έχει ήδη προκαλέσει επίπτωση στην λειτουργία της εργασίας.

Μια εργασία χαρακτηρίζεται από την δυνατότητα κάθετης κλιμάκωσης (scale-up), δηλαδή την δυνατότητα να εκμεταλλευτεί την αύξηση των πόρων ανά εξυπηρετητή, και την δυνατότητα οριζόντιας κλιμάκωσης (scale-out), δηλαδή την δυνατότητα να εκμεταλλευτεί μια αύξηση στον αριθμό εξυπηρετητών. Εάν μπορούμε να μετρήσουμε το όφελος από την αύξηση των πόρων (scale-up impact) και από την αύξηση των εξυπηρετητών (scale-out impact), διαθέτουμε μια σημαντική πληροφορία που μπορεί να αξιοποιηθεί από τον Μηχανισμό Consolidation. Συγκεκριμένα, ο μηχανισμός αυτός θα δρομολογήσει αρχικά την εργασία σε συστοιχία με βέλτιστο αριθμό πόρων και εξυπηρετητών. Εάν οι επιδόσεις της εργασίας μεταβληθούν, ο μηχανισμός μπορεί να αντισταθμίσει την μεταβολή, αξιοποιώντας τις δυνατότητες κατάλληλης κλιμάκωσης της εργασίας, καθώς γνωρίζει εκ των προτέρων την αναμενόμενη επίδρασή της. Εάν το νέφος διαθέτει προσωρινή υπερεπάρκεια πόρων ή εξυπηρετητών, ή αντίθετα αντιμετωπίσει προσωρινή ανεπάρκεια, μπορεί με βάση τις παραπάνω πληροφορίες να οδηγηθεί σε ανακατανομή των εργασιών, επιδιώκοντας αντίστοιχα το μεγαλύτερη όφελος ή την μικρότερη δυνατή επίπτωση.

Ορισμένες σύγχρονες τεχνικές δρομολόγησης εργασιών επιδιώκουν να γνωρίζουν τις δυνατότητες κάθετης και οριζόντιας κλιμάκωσης των εργασιών, και να συνυπολογίζουν αυτή τη πληροφορία κατά τη διαδικασία δρομολόγησης μιας εργασίας.

3. Τεχνικές και αλγόριθμοι δρομολόγησης εργασιών

Στο κεφάλαιο αυτό εξετάζουμε διάφορες τεχνικές δρομολόγησης εργασιών, ξεκινώντας από τις πιο απλές τεχνικές (που είναι οι πιο αναποτελεσματικές) και φτάνοντας έως σύνθετους αλγορίθμους που λαμβάνουν υπόψη τα προβλήματα των παρεμβολών, της ετερογένειας και της κλιμάκωσης, ώστε να οδηγηθούμε σε πιο αποτελεσματική αξιοποίηση των διαθέσιμων πόρων.

3.1. Απλές τεχνικές δρομολόγησης εργασιών

Απλές τεχνικές δρομολόγησης εργασιών συναντάμε στις ίδιες τις πλατφόρμες διαχείρισης υπηρεσιών νέφους (Cluster Management Frameworks). Αυτές οι πλατφόρμες παρέχουν υπηρεσίες ασφάλειας, υψηλής διαθεσιμότητας και παρακολούθησης της λειτουργίας των συστοιχιών που απαρτίζουν το νέφος. Παράλληλα είναι σε θέση να διαχειρίζονται τους κοινόχρηστους πόρους, επομένως να τους κατανέμουν και να τους αναθέτουν σε εργασίες που τους απαιτούν.

Το πρώτο στάδιο είναι να προσδιορίσουν τους πόρους που απαιτούνται για την λειτουργία μιας εργασίας: Αριθμό από εξυπηρετητές, cores, μνήμη, bandwidth κλπ. Το δεύτερο στάδιο είναι να εντοπίσουν τους απαραίτητους πόρους στις διαθέσιμες συστοιχίες εξυπηρετητών και να δρομολογήσουν ανάλογα την κάθε εργασία. Σε αυτό το στάδιο αξιοποιούν απλούς αλγόριθμους κατανομής που θα αναλύσουμε παρακάτω. Ένα τρίτο στάδιο είναι να παρακολουθούν τις επιδόσεις των εργασιών και να δρομολογούν διορθωτικές κινήσεις εφόσον διαπιστώσουν πρόβλημα.

Εφαρμογές διαχείρισης όπως οι Mesos [43], Omega [50] ή Torque [51] απαιτούν από τον ίδιο τον πελάτη να προσδιορίσει τους πόρους που κρίνει απαραίτητους για την εξυπηρέτησή του. Στη συνέχεια, με βάση τον κατάλογο των διαθέσιμων πόρων, προσπαθούν να εφαρμόσουν μια δίκαιη κατανομή τους, ανάμεσα στις εργασίες που τους απαιτούν [52]. Οι εφαρμογές μπορούν να τροφοδοτήσουν και διάφορα ανεξάρτητα πλαίσια προγραμματισμού (όπως το Hadoop) και να προσφέρουν πόρους, τους οποίους θα αξιοποιήσουν εσωτερικά τα ίδια τα πλαίσια.

Πλατφόρμες - μεσάζοντες όπως η Rightscale [57] λειτουργούν απολογιστικά, με την λογική της αυτόματης κλιμάκωσης, προκειμένου να αντισταθμίσουν μια αλλαγή στους εξυπηρετητές. Ανάλογα με τον παρατηρούμενο φόρτο, αυξομειώνουν τον αριθμό των διαθέσιμων φυσικών ή εικονικών μηχανών που χρησιμοποιούνται για την εξυπηρέτηση της εργασίας και προσφέρουν ελαστικότητα στη λύση.

Η πλατφόρμα DejaVu [53] μπορεί να αναγνωρίσει αν μια εργασία ανήκει σε κάποιο μικρό σύνολο από γνωστές κατηγορίες εργασιών. Γνωρίζοντας ποιοι πόροι είναι

απαραίτητοι για κάθε τέτοια κατηγορία, επιταχύνει την απόδοση των κατάλληλων πόρων. Άλλες πλατφόρμες όπως οι CloudScale [54], Press [55] και Agile [56] επιχειρούν μια πρόβλεψη των απαιτούμενων πόρων, συχνά χωρίς να προϋποθέτουν εκ των προτέρων γνώση των απαιτήσεων της εργασίας. Άλλο χαρακτηριστικό σύστημα διαχείρισης που υλοποιεί τεχνικές δρομολόγησης εργασιών είναι το open-source σύστημα διαχείρισης Eucalyptus [29], [30].

Οι αλγόριθμοι δρομολόγησης που υποστηρίζουν τα συστήματα αυτά είναι συνήθεις αλγόριθμοι κατανομής που προέρχονται από τη βιβλιογραφία του bin-packing. Είναι απλοί και ταχείς στην εκτέλεση και προσπαθούν κυρίως να ικανοποιήσουν τις ονομαστικές απαιτήσεις που θέτουν οι ίδιοι οι χρήστες για τις εργασίες τους. Δεν είναι τόσο αποδοτικοί, καθώς δεν εξετάζουν τις πραγματικές απαιτήσεις και δεν λαμβάνουν υπόψη την ετερογένεια του data center, τις παρεμβολές από άλλες εγκατεστημένες εργασίες και τις ανάγκες μελλοντικής κλιμάκωσης της εργασίας. Για το λόγο αυτό αναφέρονται επιγραμματικά και δεν θα επιμείνουμε στην ανάλυση και αξιολόγησή τους, παρά μόνο ως μέτρο σύγκρισης πιο εξελιγμένων αλγορίθμων.

Κριτήρια για την επιλογή συστοιχίας		Κριτήρια για την επιλογή μηχανής	
Κωδικός Αναφοράς	Όνομα Κριτηρίου	Κωδικός Αναφοράς	Όνομα Κριτηρίου
LFF	Least Full First	FF	FIRST FIT
		NF	NEXT FIT
PAL	Percent Allocated	MF	MOST-FULL FIRST
		LF	LEAST-FULL FIRST
RAN	Random	RA	RANDOM
		TP	TAG & PACK

Σχήμα 8 - Κριτήρια επιλογής συστοιχίας & μηχανής

Οι αλγόριθμοι αυτοί υλοποιούν συνήθως δύο στάδια επιλογής, όπως δείχνουμε στο Σχήμα 8. Το πρώτο στάδιο αφορά την επιλογή συστοιχίας υπολογιστών στην οποία θα εγκατασταθεί η εργασία, και το δεύτερο στάδιο αφορά την επιλογή συγκεκριμένου εξυπηρετητή από την συστοιχία. Εάν μια εργασία θέτει ως κριτήριο την επικοινωνία της με άλλη εργασία που ήδη έχει εγκατασταθεί, τότε συνήθως προτιμάται η εγκατάστασή της στην ίδια συστοιχία προκειμένου να επιτύχουμε ταχύτερη (τοπική) επικοινωνία.

Ο αλγόριθμος **Least-Full First** (LFF) κατατάσσει τις συστοιχίες ξεκινώντας από αυτήν με το χαμηλότερο ποσοστό χρήστης και καταλήγει σε αυτήν με το υψηλότερο. Το σύστημα επιχειρεί να εγκαταστήσει την εργασία στη πρώτη επιλογή της λίστας. Σε

περίπτωση επιτυχίας ο αλγόριθμος τερματίζει ενώ σε περίπτωση αποτυχίας επιχειρεί την εγκατάσταση στη δεύτερη επιλογή της λίστας κ.ο.κ.

Ο αλγόριθμος **Percent Allocated** (PAL) τοποθετεί σε λίστα τις συστοιχίες του νέφους ξεκινώντας από αυτή που μπορεί να ικανοποιήσει το μεγαλύτερο ποσοστό των εργασιών που εκκρεμούν προς εγκατάσταση. Ενδείκνυται σε περιπτώσεις όπου εκκρεμεί η εξυπηρέτηση πάρα πολλών αιτημάτων, καθώς επιχειρεί να εξυπηρετήσει γρήγορα τον μεγαλύτερο δυνατό αριθμό εργασιών.

Ο αλγόριθμος **Random** (RAN) τοποθετεί τυχαία σε λίστα τις συστοιχίες του νέφους και επιχειρεί να εγκαταστήσει την εργασία στη πρώτη συστοιχία της λίστας. Εάν επιτύχει σταματά ενώ αν αποτύχει εξετάζει την επόμενη επιλογή στη λίστα. Πρόκειται για έναν από τους ταχύτερους σε εκτέλεση αλγόριθμους και είναι αποδοτικός σε περιπτώσεις που το data center υποχρησιμοποιείται στο σύνολό του.

Ω προς την επιλογή εξυπηρετητή, ο αλγόριθμος **First Fit (FF)** διαθέτει μια απλή λίστα των διαθέσιμων μηχανών, για παράδειγμα με τη σειρά που έχουν ενταχθεί στη συστοιχία (από παλιότερη προς νεώτερη). Δοκιμάζει σειριακά να τοποθετήσει μια εισερχόμενη εργασία σε κάθε διαθέσιμη μηχανή, και εξετάζει αν το σύστημα είναι επαρκές να εξυπηρετήσει τις ονομαστικές απαιτήσεις. Εάν η προς εξέταση μηχανή είναι επαρκής ο αλγόριθμος σταματά, ενώ αν είναι ανεπαρκής προχωρά στην εξέταση της επόμενης μηχανής. Ουσιαστικά ο αλγόριθμος στοχεύει να ικανοποιήσει το αίτημα και όχι να βελτιστοποιήσει τη χρήση πόρων, γι αυτό μειώνει πολύ σημαντικά την πιθανότητα βέλτιστης τοποθέτησης σε σχέση με άλλους αλγόριθμους.

Ο αλγόριθμος **Next Fit** (NF) είναι παραλλαγή του FF. Θυμάται σε ποια μηχανή εγκαταστάθηκε η πιο πρόσφατη εργασία και ξεκινά δοκιμές από την αμέσως επόμενη στη λίστα. Ουσιαστικά με την παραλλαγή αυτή προσπαθεί να αποφύγει την άσκοπη διέλευση από μηχανές που έχουν ικανοποιήσει πρόσφατες απαιτήσεις, και επομένως εξασφαλίζει μια πιο ομοιόμορφη κατανομή των εργασιών στις διαθέσιμες μηχανές.

Ο αλγόριθμος **Least-Full First** (LF) εξετάζει την δεδομένη κατάσταση λειτουργίας της συστοιχίας και αποφασίζει να τοποθετήσει μια εισερχόμενη εργασία στη μηχανή με τον λιγότερο φόρτο. Προφανώς απαιτεί την ύπαρξη συστήματος που παρακολουθεί την χρήση των πόρων και τροφοδοτεί με πληροφορίες το σύστημα διαχείρισης. Πρόκειται για έναν αλγόριθμο που επιχειρεί να ισοκατανείμει τις εργασίες στο σύνολο των διαθέσιμων πόρων και εμμέσως πετυχαίνει να ελαχιστοποιήσει τις παρεμβολές.

Παραλλαγή του LF είναι ο αλγόριθμος **Most-Full First** (MF) που ξεκινά από τις μηχανές με τον υψηλότερο φόρτο. Ενδείκνυται στο σενάριο του Σχήματος 4 που

εξετάσαμε σε προηγούμενο κεφάλαιο, όπου μεγάλος αριθμός μηχανών βρίσκεται σε sleep mode. Καταβάλλει προσπάθεια να εκμεταλλευτεί πλήρως τους πόρους των μηχανών που ήδη λειτουργούν και εξυπηρετούν αιτήματα, πριν προχωρήσει σε χρήση νέων. Ο αλγόριθμος ενδείκνυται για την εξοικονόμηση κόστους λειτουργίας (κόστους ενέργειας, ψύξης κλπ) αλλά οδηγεί σε μηχανές που λειτουργούν σε μέγιστο φόρτο, όπου το φαινόμενο των παρεμβολών είναι ιδιαίτερα έντονο και το QoS των πελατών συχνά δεν ικανοποιείται.

Ο αλγόριθμος **Random** (RA) τοποθετεί τυχαία σε λίστα τις μηχανές της συστοιχίας και επιχειρεί να εγκαταστήσει την εργασία στη πρώτη μηχανή της λίστας. Εάν επιτύχει σταματά ενώ αν αποτύχει εξετάζει την επόμενη επιλογή στη λίστα.

Τέλος ο αλγόριθμος **Tag & Pack** (TP) θεωρεί ότι κάθε μηχανή τη συστοιχίας εξειδικεύεται στην εξυπηρέτηση συγκεκριμένου είδους εργασιών. Μόλις αναγνωριστεί ένα αίτημα για την εγκατάσταση μιας τέτοιας εργασίας, ο αλγόριθμος εξετάζει κατ' αρχάς τη δυνατότητα εγκατάστασης στις κατάλληλες μηχανές και μόνο αν αποτύχει εξετάζει τις υπόλοιπες, που έχει αποκλείσει αρχικά ως ακατάλληλες.

Μια τυπική τεχνική δρομολόγησης που συναντάται σε περιβάλλον υπολογιστικού νέφους, είναι η **Least Loaded** (LL). Ουσιαστικά υλοποιείται με αλγόριθμους Least-Full First τόσο για την επιλογή της λιγότερο επιβαρυμένης συστοιχίας του νέφους, όσο και στην επιλογή του λιγότερου επιβαρυμένου εξυπηρετητή, ανάμεσα σε αυτούς που απαρτίζουν τη συστοιχία που επελέγη.

3.2. Τεχνική PACMan

Η τεχνική Performance Aware Virtual Machine Consolidation Manager (PACMan) δρομολογεί εισερχόμενες εργασίες σε περιβάλλον νέφους, λαμβάνοντας υπόψη τις παρεμβολές, δηλαδή την πιθανή επίπτωση από άλλες εργασίες με τις οποίες θα συστεγαστεί [40]. Βασικός στόχος της μεθόδου είναι, μετά την συστέγαση της εργασίας με άλλες, η επίδοσή της να παραμείνει η μέγιστη δυνατή, μέσα σε ανεκτά επίπεδα που έχουν προκαθοριστεί. Η τεχνική περιλαμβάνει δύο μεθόδους λειτουργίας, την P-Mode και την Eco-Mode που αναλύονται παρακάτω.

3.2.1. Performance Mode ή P-Mode

Στόχος της μεθόδου είναι να αναγνωρίσει τη μείωση της επίδοσης που θα υποστεί κάθε εργασία λόγω παρεμβολών από τη συστέγασή της με άλλες, και να επιλέξει συνδυασμούς εργασιών που ελαχιστοποιούν αυτές τις επιπτώσεις.

Για μια δεδομένη εργασία *i*, συμβολίζουμε με d_i την μείωση της επίδοσης λόγω παρεμβολών (d - από degradation). Προφανώς για εργασία που λειτουργεί μόνη της, έχουμε d_i = 1. Για δύο συστεγασμένες εργασίες Α και Β, εάν παρατηρήσουμε ότι ο χρόνος εκτέλεσης της Α αυξάνει κατά 50% εξαιτίας της συστέγασης με την Β, το αναπαριστούμε με d_A = 1.5. Σαν παράδειγμα, δείχνουμε στο Πίνακα 1 την μείωση της επίδοσης τριών εργασιών Α, Β, C όταν λειτουργούν μόνες τους, και σε συστέγαση ανά δύο, σε εξυπηρετητή που μπορεί να δεχθεί το πολύ 2 εικονικές μηχανές (αποκλείουμε για τις ανάγκες του παραδείγματος το σενάριο ABC).

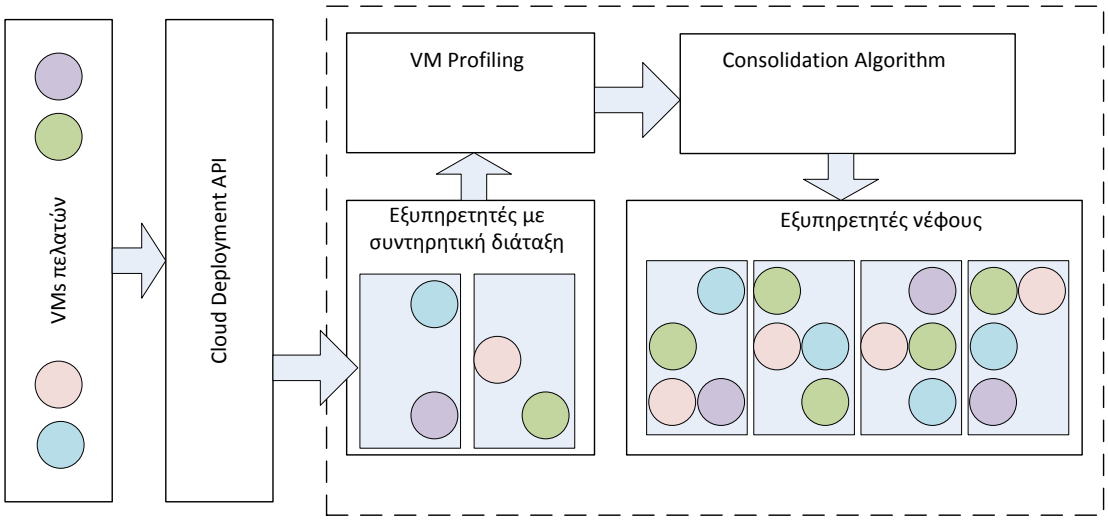
AB	AC	BC	A	B	C
$d_A = 1.1$ $d_B = 1.1$	$d_A = 1.0$ $d_C = 1.5$	$d_B = 1.0$ $d_C = 1.1$	1	1	1

Πίνακας 1 - Μείωση απόδοσης εργασιών λόγω παρεμβολών

Η μέθοδος απαιτεί να ορίσουμε την μέγιστη αποδεκτή μείωση στην επίδοση, για παράδειγμα d_i = 1.1 (10% μείωση επίδοσης) και βάσει αυτής αποκλείει συνδυασμούς που παραβιάζουν το επιθυμητό επίπεδο της υπηρεσίας, όπως ο συνδυασμός AC του παραδείγματος. Είναι επίσης κατανοητό ότι η εργασία Β είναι καταλληλότερη να συστεγαστεί με την C παρά με την Α, λόγω της μικρότερης επίπτωσης που υφίσταται. Ωστόσο και η συστέγαση της Β με την Α είναι αποδεκτή, εντός των ορίων που έχουν τεθεί.

Σε μια μεγάλη εγκατάσταση η μέθοδος αυτή θα έπρεπε να επεκταθεί σε δοκιμές όλων των πιθανών συνδυασμών των εργασιών μεταξύ τους. Όπως έχει αναφερθεί, πρόκειται για πρόβλημα μεγάλης πολυπλοκότητας (NP-Complete). Για να ξεπεράσουμε τη δυσκολία του οδηγούμαστε σε χρήση μιας μεθόδου profiling, η οποία προσπαθεί να κατατάξει την εργασία σε γνωστές κατηγορίες, συγκρίνοντας το προφίλ της με άλλα χαρακτηριστικά προφίλ. Ο αριθμός των χαρακτηριστικών προφίλ διατηρείται μικρός, για παράδειγμα της τάξης των 128, προκειμένου να μην αυξηθεί πολύ η πολυπλοκότητα του αλγορίθμου.

Στο Σχήμα 9 αναπαριστούμε τα βασικά στάδια της τεχνικής PACMan.



Σχήμα 9 - Σχηματική αναπαράσταση τεχνικής PACMan

Αρχικά τοποθετούμε την εργασία σε μια συστοιχία όπου λειτουργούν ήδη άλλες εργασίες σε πολύ συντηρητική διάταξη (Conservatively Packed Servers). Ως συντηρητική εννοούμε μια διάταξη που κρατά αναξιοποίητο ένα μεγάλο ποσοστό των διαθέσιμων πόρων της (της τάξης του 50% και άνω) [21]. Η διάταξη αυτή αποτελείται από μικρό αριθμό εξυπηρετητών (τυπικά 1% έως 5% των διαθέσιμων εξυπηρετητών του νέφους) και μια εργασία παραμένει σε αυτήν για μικρό χρονικό διάστημα (30 έως 60 λεπτά) προκειμένου να αναγνωριστεί το προφίλ της. Στο διάστημα αυτό η εργασία λειτουργεί κανονικά και όχι δοκιμαστικά, επομένως ο πελάτης επωφελείται ήδη των υπηρεσιών του νέφους. Καθώς η συστοιχία εκτελεί τις εργασίες, ακολουθούμε μεθόδους αναγνώρισης του προφίλ [20], [21] που χαρακτηρίζουν την εργασία και συμπληρώνουν μια σειρά από παραμέτρους, που δείχνουν την επίπτωση που υφίσταται και προκαλεί λόγω της συστέγασής της με άλλες.

Στο τέλος αυτής της διαδικασίας, ένας αλγόριθμος Consolidation λαμβάνει ως είσοδο τις μετρικές που ορίζουν την επίδραση από παρεμβολές, όπως υπολογίστηκαν κατά τη διαδικασία Profiling. Με βάση αυτές υποδεικνύει το βέλτιστο εξυπηρετητή στον οποίο θα πρέπει να εγκατασταθεί η εργασία, έχοντας λάβει υπόψη το πρόβλημα των παρεμβολών. Οι λεπτομέρειες υλοποίησης του αλγορίθμου έχουν τεκμηριωθεί [40] και επιγραμματικά αναφέρονται παρακάτω:

- Ο αλγόριθμος ορίζει όλα τα πιθανά σετ S εργασιών που είναι πιθανόν να συστεγαστούν.

- Για κάθε σετ S εργασιών, ορίζεται μια τιμή $w(S)$ που προσδιορίζει την χρήση ενός πόρου που μας ενδιαφέρει να εξοικονομήσουμε (μπορεί να είναι το κόστος της ενέργειας που καταναλώνουν, ο αριθμός των cores που χρησιμοποιούν,

κλπ). Φυσικά ο αλγόριθμος μπορεί να τρέξει πολλές φορές, ώστε να συνεκτιμηθούν πολλαπλοί πόροι.

- Ορίζεται ένα κατώφλι επίδοσης που δεν επιτρέπεται να παραβιαστεί, και ο αλγόριθμος απαλείφει εκείνα τα σετ που το παραβιάζουν.

- Ο αλγόριθμος υπολογίζει μια τιμή $V(S) = w(S) / |S|$ που χαρακτηρίζει το κόστος χρήσης του πόρου από το σετ S. Σετ με περισσότερες εργασίες (υψηλότερη τιμή $|S|$) και μικρή χρήση πόρων (χαμηλή τιμή $w(S)$) χαρακτηρίζονται από χαμηλό $V(S)$.

- Ο αλγόριθμος κατατάσσει τα σετ κατά αύξον $V(S)$. Σετ που εμφανίζονται πρώτα στη λίστα έχουν χαμηλότερο κόστος από άλλα που εμφανίζονται μετέπειτα.

- Σαν τελευταίο βήμα, ο αλγόριθμος κάνει ένα μοναδικό πέρασμα της λίστας και εντοπίζει κάθε σετ S το οποίο δεν περιλαμβάνει εργασίες που έχουν ήδη συμπεριληφθεί σε προηγούμενο σετ.

- Ο αλγόριθμος τερματίζει είτε μόλις κάνει ένα πλήρες πέρασμα, είτε νωρίτερα αν έχει δρομολογήσει όλες τις διαθέσιμες εργασίες.

Προφανώς το πρώτο σετ της λίστας επιλέγεται πάντα, καθώς όλες οι εργασίες που περιλαμβάνει δεν μπορεί να εμπεριέχονται σε προηγούμενο σετ. Αν το δεύτερο σετ της λίστας δεν περιέχει κοινές εργασίες με το πρώτο, τότε επιλέγεται. Αλλιώς απορρίπτεται, ο αλγόριθμος εξετάζει το τρίτο σετ της λίστας και ούτω καθεξής.

Στο παράδειγμα του Πίνακα 1 που δώσαμε προηγουμένως με τα σετ AB, AC, BC, A, B, C, υποθέτουμε για απλούστευση ότι $w(S)$ είναι ο αριθμός των εξυπηρετητών, δηλαδή είναι 1 ανεξαρτήτως σετ, και ότι δεν είναι επιθυμητή τιμή $d_i > 1.1$. Σετ με 2 εργασίες θα έχουν $V(S) = {}^{1}/_{2}$ ενώ σετ με 1 εργασία θα έχουν $V(S) = 1$. Ο αλγόριθμος θα απέρριπτε το σετ AC που παραβιάζει το επιθυμητό κόστος. Ακολούθως θα τοποθετούσε τα υπόλοιπα σετ στην αύξουσα σειρά κόστους: BC, AB, A, B, C. Επιλέγοντας το πρώτο στοιχείο της λίστας (BC) ο αλγόριθμος θα δρομολογούσε τις εργασίες B και C σε έναν εξυπηρετητή. Ακολούθως θα αγνοούσε το δεύτερο στοιχείο της λίστας (AB) επειδή εμπεριέχει την εργασία B που έχει ήδη εξυπηρετηθεί. Στη συνέχεια θα επέλεγε το τρίτο στοιχείο της λίστας (A) και με βάση αυτή την επιλογή θα δρομολογούσε την εργασία A σε δεύτερο εξυπηρετητή. Εκεί ο αλγόριθμος θα τερματιζόταν, καθώς έχουν εξυπηρετηθεί όλες οι εργασίες.

Ήδη από αυτή την απλή εφαρμογή του αλγορίθμου, διαπιστώνουμε ότι οι τρεις εργασίες A, B, C αντί να δρομολογηθούν σε 3 φυσικές μηχανές, έχουν επωφεληθεί

από τη δυνατότητα ενοποίησης και έχουν εγκατασταθεί σε 2 φυσικές μηχανές κατά τον πλέον βέλτιστο τρόπο, χωρίς να παραβιάζονται οι προδιαγραφές που θέσαμε.

3.2.2. Eco – mode

Σε κάποιες περιπτώσεις προέχει η βελτιστοποίηση των πόρων και όχι η επίτευξη των μεγαλύτερων δυνατών επιδόσεων. Προκειμένου να ανταποκριθεί σε αυτή την απαίτηση, η τεχνική PACMan διαθέτει μια διαφορετική μέθοδο λειτουργίας που ονομάζεται eco-mode. Εδώ ο αριθμός εξυπηρετητών είναι δεδομένος και οι εργασίες έχουν ήδη δρομολογηθεί και εγκατασταθεί. Ο στόχος είναι να ελαχιστοποιηθούν οι παρεμβολές. Εφόσον διαπιστωθεί ότι κάποιες συστεγασμένες εργασίες προκαλούν έντονες παρεμβολές, τεχνικές swapping (αμοιβαίας μετάθεσης εργασιών) μπορούν να οδηγήσουν σε αποτελεσματικότερη κατανομή [42]. Ο αλγόριθμος consolidation που εφαρμόζεται σε αυτή τη περίπτωση, λειτουργεί με την παρακάτω λογική:

- Ξεκινάμε εξετάζοντας την δεδομένη κατανομή εργασιών και εντοπίζουμε όλες τις πιθανές καταστάσεις που θα μπορούσαν να προκύψουν, αν κάναμε μια και μοναδική αμοιβαία μετάθεση.

- Για κάθε τέτοια πιθανή κατάσταση, υπολογίζουμε ως σκορ ενός εξυπηρετητή, την μείωση της επίδοσης της εργασίας εκείνης που έχει πληγεί περισσότερο από την αλλαγή που έγινε. Πρόκειται για την εργασία δηλαδή που βάσει του profiling engine θα δεχθεί την μεγαλύτερη παρεμβολή, από την νέα εργασία που σκοπεύουμε να δρομολογήσουμε στον εξυπηρετητή, μέσω αμοιβαίας μετάθεσής της με κάποια υφιστάμενη.

- Το σύνολο αυτών των σκορ (για όλους τους εξυπηρετητές) είναι το κόστος του συγκεκριμένου σεναρίου αμοιβαίας μετάθεσης.

- Από όλα τα σενάρια αμοιβαίας μετάθεσης που εξετάστηκαν ως πιθανά, ένας απλός άπληστος αλγόριθμος επιλέγει εκείνο το σενάριο που εμφανίζει το χαμηλότερο σκορ και εκτελεί στη πράξη την αμοιβαία μετάθεση.

- Η παραπάνω διαδικασία επαναλαμβάνεται εκ νέου, όσο επιτρέπονται αμοιβαίες μεταθέσεις οι οποίες οδηγούν σε βελτιωμένη κατανομή εργασιών.

Η παραπάνω τεχνική δεν εγγυάται ότι το σύστημα θα φτάσει στην βέλτιστη δυνατή κατανομή εργασιών. Ωστόσο σε κάθε βήμα επιτυγχάνει μια καλύτερη κατανομή σε σχέση με την προηγούμενη κατάσταση. Αυτό από μόνο του αποτελεί κέρδος αν και το πραγματικό όφελος θα πρέπει να συγκριθεί με το κόστος που προκαλεί η ίδια η διαδικασία αμοιβαίας μετάθεσης.

Μια καλή ιδέα είναι να χρησιμοποιείται η λειτουργία performance-mode κατά την αρχική δρομολόγηση εργασιών, και περιοδικά η λειτουργία eco-mode για βελτιστοποίηση της λειτουργίας του περιβάλλοντος, όταν ένας μηχανισμός παρακολούθησης του QoS των εγκατεστημένων εργασιών διαπιστώνει πως μια μετρήσιμη ποσότητα πλησιάζει ή ξεπερνά ένα προαποφασισμένο κατώφλι.

3.3. Τεχνική Paragon (P)

Το Paragon [28] υλοποιεί μια τεχνική δρομολόγησης εργασιών που λαμβάνει υπόψη τόσο την ετερογένεια όσο και τις παρεμβολές. Το βασικό χαρακτηριστικό της τεχνικής αυτής είναι η ικανότητά της να κατηγοριοποιεί και να ταξινομεί μια άγνωστη εργασία γρήγορα και με ακρίβεια. Η τεχνική αυτή αξιοποιεί υπάρχοντα δεδομένα από προηγούμενες εγκατεστημένες εργασίες και διαθέτει δυνατότητες εκμάθησης τη συμπεριφοράς των εργασιών. Απαιτεί ένα ελάχιστο σήμα από μια νέα εργασία προκειμένου να την κατατάξει, αξιοποιώντας έναν αλγόριθμο που προβαίνει σε συστάσεις, παρόμοιες με τον αλγόριθμο του Netflix Challenge [31]. Ο αλγόριθμος του Netflix ανακαλύπτει ομοιότητες μεταξύ ταινιών και με βάση αυτές προτείνει νέες ταινίες που θα ενδιέφεραν τους χρήστες του συστήματος. Με την ίδια λογική το Paragon βρίσκει ομοιότητες στις προτιμήσεις νέων εργασιών σε σχέση με την ετερογένεια και τις παρεμβολές και έτσι τις κατηγοριοποιεί ως παρόμοιες με άλλες που έχει ήδη κατηγοριοποιήσει στο παρελθόν και γνωρίζει τη συμπεριφορά τους.

Μόλις μια εισερχόμενη αίτηση έχει ταξινομηθεί, δρομολογείται σε εκείνον το εξυπηρετητή που ταιριάζει καλύτερα ως προς την διαμόρφωση υλικού, και εμφανίζει την ελάχιστη παρεμβολή μεταξύ όλων των εργασιών που μοιράζονται τους πόρους του. Το Paragon λόγω του σχεδιασμού του προσφέρει ταχύτατη κατηγοριοποίηση σε περιβάλλον μεγάλου data center, με δεκάδες διαφορετικές πλατφόρμες και δεκάδες χιλιάδες εξυπηρετητές στους οποίους αποστέλλεται μεγάλος αριθμός από εργασίες που ήταν προηγουμένως άγνωστες.

Ειδικότερα, το Paragon έχει δοκιμαστεί σε περιβάλλον ιδιωτικού νέφους με συστοιχία 40 servers με 10 διαφορετικές διαμορφώσεις υλικού, καθώς και σε περιβάλλον Amazon EC2 με 1000 servers και 14 διαφορετικές διαμορφώσεις υλικού. Στο περιβάλλον αυτό, κατανέμοντας 2500 εργασίες, πέτυχε τήρηση του QoS σε ποσοστό 91%. Σε παρόμοιο περιβάλλον, οι αλγόριθμοι που δεν λαμβάνουν υπόψη την ετερογένεια και τις παρεμβολές πέτυχαν αντίστοιχα 14% και 11%, ενώ ένας απλός αλγόριθμος (Least Loaded) πέτυχε μόλις 3% [28].

3.3.1. Ταξινόμηση ως προς την ετερογένεια

Το Paragon χρησιμοποιεί τεχνικές Collaborative Filtering προκειμένου να προβλέψει τις επιδόσεις μιας εργασίας σε διαφορετικές διαμορφώσεις υλικού. Οι τεχνικές αυτές αξιοποιούνται ευρύτατα σε πλατφόρμες που προβαίνουν σε συστάσεις με βάση προηγούμενη συμπεριφορά και αξιοποιούν δύο αναλυτικές μεθόδους, την Singular Value Decomposition (SVD) και την PQ-Reconstruction (PQ) [32].

Αρχικά δημιουργείται ένας πίνακας $A_{m,n}$ όπου οι m γραμμές απεικονίζουν εργασίες και οι n στήλες απεικονίζουν διαφορετικές διαμορφώσεις υλικού.

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

Το κάθε στοιχείο $a_{m,n}$ του πίνακα A είναι ένα «σκορ» που χαρακτηρίζει πόσο καλά λειτουργεί η m εργασία στον n-εξυπηρετητή. Το σκορ μπορεί να προκύψει από μια μετρήσιμη τιμή που επηρεάζεται από διαφορετικές διαμορφώσεις υλικού, όπως πχ τα instructions committed per second (IPS).

Υπό κανονικές συνθήκες θα έπρεπε να δοκιμαστούν όλες οι εργασίες σε όλες τις διαμορφώσεις υλικού, ώστε να έχουμε πλήρως συμπληρωμένο τον πίνακα. Στην πράξη όμως οι παραπάνω μέθοδοι μας επιτρέπουν να επιλέξουμε λίγες εργασίες (μερικές δεκάδες) και να τις δοκιμάσουμε σε όλες τις διαμορφώσεις υλικού. Αυτό μπορεί να γίνει κατά την αρχικοποίηση της τεχνικής και δεν χρειάζεται να επαναληφθεί. Καταλήγουμε σε έναν πίνακα $A_{m,n}$ όπου οι μερικές δεκάδες γραμμές που αντιστοιχούν στις εργασίες που επιλέξαμε είναι πλήρως συμπληρωμένες. Οι υπόλοιπες άγνωστες γραμμές συμπληρώνονται με PQ-reconstruction και Stochastic Gradient Descent (SGD). Στη συνέχεια εφαρμόζεται SVD ώστε να διαπιστωθεί ποιες από τις τιμές που συμπληρώθηκαν έχουν έντονη ομοιότητα και μπορούν να θεωρηθούν ως εργασίες με παρόμοια συμπεριφορά.

Εάν στο data center προσθέσουμε έναν νέο εξυπηρετητή (n+1) με διαφορετική διαμόρφωση υλικού, αρκεί να δοκιμάσουμε τον πεπερασμένο αριθμό εργασιών σε αυτό το νέο εξυπηρετητή και να συμπληρώσουμε μια νέα στήλη στον πίνακα $A_{m,n+1}$.

Σε πραγματικό χρόνο τώρα, δοκιμάζουμε μια εισερχόμενη εργασία για περίοδο 1 λεπτού. Αρκεί να την δοκιμάσουμε σε 2 διαφορετικές διαμορφώσεις εξυπηρετητών (*Server Configurations – SC*) και να συμπληρώσουμε μια νέα γραμμή στον πίνακα $A_{m+1,n}$. Σε αυτή τη νέα γραμμή μόνο 2 τιμές είναι γνωστές, ωστόσο αξιοποιώντας το Collaborative Filtering αναγνωρίζουμε ομοιότητες μεταξύ παλιών (γνωστών)

εργασιών και της νέας που προστέθηκε και έτσι προβλέπουμε τις υπολειπόμενες τιμές. Καθώς νέες εργασίες προστίθενται στο σύστημα, η πυκνότητα του πίνακα αυξάνει, κάτι που βελτιώνει το μηχανισμό πρόβλεψης.

Ο Πίνακας 2 συνοψίζει τις δοκιμές αυτής της τεχνικής σε περιβάλλον 40 εξυπηρετητών με 10 διαφορετικά server configurations (SC) [28]. Εξετάστηκε μεγάλος αριθμός από δοκιμαστικές εργασίες single-threaded (ST), multi-threaded (MT), multi-programmed (MP) και I/O-bound (IO). Οι εργασίες ανατέθηκαν στο βέλτιστο εξυπηρετητή σε ποσοστό 83% έως 89% ενώ ανατέθηκαν σε καλό εξυπηρετητή (με απόκλιση 5% από το βέλτιστο) σε ποσοστό 89% - 92%.

Πιο κάτω στον ίδιο πίνακα φαίνεται η επιτυχία του μηχανισμού πρόβλεψης η οποία κρίνεται ως αρκετά υψηλή, με δεδομένο ότι οι εργασίες δοκιμάστηκαν για 1 μόλις λεπτό σε 2 εξυπηρετητές.

Metric	Applications (%)			
	ST	MT	MP	IO
Selected best SC	86%	86%	83%	89%
Selected SC within 5% of best	91%	90%	89%	92%
Correct SC ranking (best to worst)	67%	62%	59%	43%
90% correct SC ranking	78%	71%	63%	58%
50% correct SC ranking	93%	91%	89%	90%
Training & best SC match	28%	24%	18%	22%

Πίνακας 2 - Μετρικές για την κατηγοριοποίηση ως προς την ετερογένεια

Στην τελευταία γραμμή μάλιστα φαίνεται ότι το Training Configuration (οι εξυπηρετητές στους οποίους δοκιμάστηκαν οι εργασίες), δεν συνέπιπτε με το Best Match Configuration (τους εξυπηρετητές που αναδείχθηκαν ως βέλτιστοι). Έτυχε και συνέπεσαν σε χαμηλό ποσοστό, 18 – 28%, κάτι που σημαίνει ότι ακόμα και με τη χρήση μη-βέλτιστων εξυπηρετητών στις δοκιμές, η μέθοδος πέτυχε βέλτιστη κατανομή.

3.3.2. Ταξινόμηση ως προς την παρεμβολή

Υπάρχουν δύο ειδών παρεμβολές που μας ενδιαφέρει να λάβουμε υπόψη. Η παρεμβολή που μπορεί *να δεχθεί* μια εργασία, μόλις λειτουργήσει σε εξυπηρετητή όπου ήδη εκτελούνται άλλες εργασίες, και η παρεμβολή που θα *προκαλέσει* η εργασία στις υπόλοιπες. Ο τρόπος που ανιχνεύουμε τις παρεμβολές, είναι να την εκτελούμε παράλληλα με benchmark workloads, ειδικές δηλαδή εργασίες που έχουμε προαποφασίσει να χρησιμοποιήσουμε ως αναφορές. Αυξάνουμε σταδιακά

την ένταση του benchmark μέχρι η εργασία μας να παραβιάσει το QoS της, δηλαδή το μέγεθος της παρεμβολής να φτάσει σε μη-αποδεκτό επίπεδο. Εδώ να σημειώσουμε ότι δεν χρειάζεται να δεχθούμε το QoS που ορίζει ο πελάτης, γιατί αυτό μπορεί να είναι επιτεύξιμο ακόμα και υπό συνθήκες ακραίας παρεμβολής. Στο Paragon, ως QoS ορίζεται το 95% της απόδοσης που έχει μια εργασία σε περιβάλλον απομόνωσης, όταν δηλαδή λειτουργεί μόνη της σε έναν εξυπηρετητή.

Συνήθως επιλέγονται συγκεκριμένα benchmark workloads, καθένα από τα οποία αποτελεί χαρακτηριστική πηγή παρεμβολής κυρίως σε έναν από τους διαθέσιμους κοινόχρηστους πόρους (CPU, χωρητικότητα και εύρος ζώνης μνήμης, εύρος ζώνης δικτύου, χωρητικότητα και εύρος ζώνης συστήματος αποθήκευσης, ιεραρχία cache κλπ). Στο Paragon έχουν προσδιοριστεί 10 τέτοιες πηγές παρεμβολής (*Source of Interference – SoI*). Η μεθοδολογία όμως μπορεί να επεκταθεί και σε επιπλέον κοινόχρηστους πόρους.

Για κάθε SoI έχει οριστεί μια χαρακτηριστική εργασία-benchmark, βάσει των οποίων γίνονται οι δοκιμές και προκύπτει μια βαθμολογία ως προς την ευαισθησία (*Sensitivity Score - SSC*). Σε αυτή τη περίπτωση το σκορ αποτυπώνει την ανεκτικότητα στη παρεμβολή. Εργασίες με υψηλό SSC της τάξης 60% και άνω προσφέρονται για συστέγαση, ενώ εργασίες με χαμηλή ανεκτικότητα πρέπει να αποφεύγονται.

Αντίστοιχα, μετράμε την επίπτωση της εργασίας στο benchmark, ανεβάζοντας σταδιακά την έντασή της μέχρι το benchmark να υποστεί μείωση των επιδόσεών του κατά 5% σε σχέση με τις επιδόσεις του σε περιβάλλον απομόνωσης. Σε αυτή τη περίπτωση, υψηλό SSC της τάξης 60% και άνω αντιστοιχεί σε εργασίες που προκαλούν σοβαρές παρεμβολές στο συγκεκριμένο κοινόχρηστο πόρο.

Για να ταξινομήσουμε μια εργασία ως προς την παρεμβολή, χρησιμοποιούμε δύο φορές την τεχνική collaborative filtering που αναφέρθηκε στην παράγραφο 3.4.1. Σχηματίζουμε δύο πίνακες $A_{m,n}$ όπου οι m γραμμές απεικονίζουν εργασίες και οι n στήλες απεικονίζουν πηγές παρεμβολής (SoI). Το στοιχείο $a_{m,n}$ του κάθε πίνακα είναι ένα «σκορ» που χαρακτηρίζει την ευαισθησία (SSC) στην δεχόμενη και προκαλούμενη παρεμβολή αντίστοιχα.

Όπως και στην ταξινόμηση ως προς την ετερογένεια, αρχικοποιούμε την εφαρμογή Paragon επιλέγοντας λίγες εργασίες (μερικές δεκάδες) που δοκιμάζονται έναντι όλων των benchmarks. Σε πραγματικό χρόνο, δοκιμάζουμε μια εισερχόμενη εργασία για περίοδο 1 λεπτού έναντι δύο (2) τυχαίων benchmarks, και το σκορ τους προστίθεται σε νέα γραμμή του πίνακα, η οποία συμπληρώνεται με τις αναλυτικές μεθόδους SVD και PQ. Η μέθοδος αυτή, πέραν του ότι αποδεικνύεται ακριβής, είναι

και ταχύτατη. Εφαρμόζεται σε τεράστιες εγκαταστάσεις με δεκάδες χιλιάδες εγκατεστημένες εργασίες και υψηλό ρυθμό εισερχόμενων νέων εργασιών.

Ο Πίνακας 3 συνοψίζει τις δοκιμές αυτής της τεχνικής σε περιβάλλον 40 εξυπηρετητών με single-threaded (ST) και multi-threaded (MT) εργασίες [28].

Metric	Percentage (%)
Average sensitivity error across all SoIs	5.3%
Average error for sensitivities $< 30\%$	7.1%
Average error for sensitivities $< 60\%$	5.6%
Average error for sensitivities $> 60\%$	3.4%
Apps with $< 5\%$ error	ST: 65% MT: 58%
Apps with $< 10\%$ error	ST: 81% MT: 63%
Apps with $< 20\%$ error	ST: 90% MT: 89%
SoI with highest error	
for ST: L1 i-cache	15.8%
for MT: LLC capacity	7.8%
Frequency L1 i-cache used as offline SoI	14.6%
Frequency LLC cap used as offline SoI	11.5%
SoI with lowest error	
for ST: network bandwidth	1.8%
for MT: storage bandwidth	0.9%

Πίνακας 3 - Μετρικές για την κατηγοριοποίηση ως προς την παρεμβολή

Οι λανθασμένες αναθέσεις περιορίστηκαν στο ποσοστό του 5.3% συνολικά. Οι πηγές παρεμβολής (SoI) με το μεγαλύτερο ποσοστό σφάλματος αφορούν L1 instruction cache για single-threaded εργασίες, και LLC Capacity (L2/L3) για multi-threaded εργασίες. Πάντως ο λόγος της υψηλότερης αστοχίας δεν οφείλεται στη συγκεκριμένη μέθοδο, καθώς έχει τεκμηριωθεί και με την μέθοδο Bubble-Up [21] η δυσκολία να προσδιοριστεί επακριβώς ένα σκορ για τις παρεμβολές που προκαλούνται από κοινή χρήση των συγκεκριμένων πόρων.

3.3.3. Μηχανισμός επιλογής εξυπηρετητών

Μόλις μια εισερχόμενη εργασία ταξινομηθεί ως προς την ετερογένεια και τις παρεμβολές, η τεχνική Paragon αναλαμβάνει να επιλέξει εξυπηρετητή με έναν απλό *άπληστο αλγόριθμο*. Ο άπληστος αλγόριθμος έχει τα χαρακτηριστικά ότι είναι απλός, γρήγορος και «φυσιολογικός». Στην περίπτωση της τεχνικής Paragon επιτυγχάνει την βέλτιστη λύση σε υψηλά ποσοστά, λόγω του ότι τον τροφοδοτούμε με σκορ μεγάλης αξιοπιστίας τα οποία από μόνα τους υποδεικνύουν τις βέλτιστες λύσεις. Το σκορ ταξινόμησης ως προς την ετερογένεια, εγγυάται ότι σε κάθε server

configuration (SC) θα εγκαταστήσουμε μόνο εργασίες που πραγματικά θα ωφεληθούν από τα ιδιαίτερα χαρακτηριστικά του συγκεκριμένου SC. Το σκορ ταξινόμησης ως προς τις παρεμβολές εγγυάται ότι θα εγκαταστήσουμε την εργασία σε εκείνον τον εξυπηρετητή όπου θα δεχθεί και θα προκαλέσει τις λιγότερες παρεμβολές. Και οι δύο επιλογές οδηγούν σε ελαχιστοποίηση του χρόνου εκτέλεσης της εργασίας, κάτι που συνεπάγεται συντομότερη απελευθέρωση του εξυπηρετητή για νέες εργασίες ή για αδρανοποίηση, σε περίπτωση που δεν αξιοποιείται από καμία εργασία.

Ο αλγόριθμος αρχικά επιλέγει τις δρομολογήσεις που θα προκαλέσουν και θα δεχθούν τη μικρότερη δυνατή παρεμβολή από άλλες εγκατεστημένες εργασίες, με βάση τα σκορ που προέκυψαν από τη διαδικασία ταξινόμησης. Ουσιαστικά υπολογίζει δύο μετρικές, την $D_1 = t_{server} - c_{newapp}$ (παρεμβολή που δέχεται ο εξυπηρετητής και προκαλείται από την εργασία) και την $D_2 = t_{newapp} - c_{server}$ (παρεμβολή που δέχεται η εργασία και προκαλείται από τον εξυπηρετητή), όπου συμβατικά t αντιστοιχεί στην tolerated (ανεχόμενη) και c στην caused (προκαλούμενη) παρεμβολή για κάθε *SoI*. Η ευαισθησία ενός εξυπηρετητή σε προκαλούμενες παρεμβολές είναι το άθροισμα των σκορ των εγκατεστημένων εργασιών, ενώ η ανεχόμενη ευαισθησία είναι η ελάχιστη τιμή αυτών. Ιδανικός υποψήφιος είναι ο εξυπηρετητής του οποίου τα D_1 και D_2 είναι ακριβώς μηδέν για όλα τα SoI, κάτι που υποδεικνύει ότι δεν υπάρχει καθόλου παρεμβολή. Πρακτικά αναζητούμε μικρές θετικές τιμές για τα D_1 και D_2. Μεγάλες θετικές τιμές υποδεικνύουν κακή αξιοποίηση πόρων, ενώ αρνητικές τιμές οδηγούν σε παραβίαση του QoS και πρέπει να αποφεύγονται.

Έπειτα, μεταξύ αυτών ο αλγόριθμος επιλέγει το καταλληλότερο server configuration (SC). Η σειρά με την οποία γίνεται η επιλογή (πρώτα ως προς τις παρεμβολές και έπειτα ως προς την ετερογένεια), βασίζεται στην παρατήρηση ότι τα προβλήματα λόγω παρεμβολών τυπικά έχουν μεγαλύτερες επιπτώσεις στη λειτουργία της εργασίας σε σχέση με τα προβλήματα λόγω ετερογένειας, όπως δείχνουμε και στο Σχήμα 6.

Ο αλγόριθμος επεξηγείται αναλυτικά για την μετρική D_1 αλλά ισχύει παρομοίως και για την D_2:

- Ο μηχανισμός επιλογής εξυπηρετητών κρατάει τα states D_1, D_2 με το άθροισμα των σκορ όλων των εγκατεστημένων εργασιών, ανά Server Configuration (SC) και ανά Source of Interference (SoI).

- Μετά την κατηγοριοποίηση μιας νέας εισερχόμενης εργασίας, ξεκινάμε από τον πόρο που έχει το υψηλότερο σκορ ευαισθησίας, δηλαδή είναι ο πιο δύσκολος πόρος να εξυπηρετηθεί.

- Ανακτούμε τα states των εξυπηρετητών και επιλέγουμε το σύνολο των εξυπηρετητών με μη-αρνητικό D_1 για το συγκεκριμένο SoI.

- Ακολούθως εκτελούμε την ίδια διαδικασία για το δεύτερο κατά σειρά ταξινόμησης SoI, απαλείφοντας από τη λίστα όσους εξυπηρετητές έχουν αρνητικό D_1 για το συγκεκριμένο SoI.

- Η διαδικασία επαναλαμβάνεται έως ότου έχουν ληφθεί υπόψη όλα τα SoI.

- Ακολούθως λαμβάνουμε υπόψη την ετερογένεια. Από τη λίστα των υποψήφιων εξυπηρετητών που έχει απομείνει, επιλέγουμε εκείνους που διαθέτουν το καλύτερο SC για τη συγκεκριμένη εργασία.

- Επαναλαμβάνουμε παρόμοια διαδικασία για την μετρική D_2.

- Τέλος, από τη λίστα εξυπηρετητών που έχει απομείνει, επιλέγουμε εκείνον με το $min(||D_1 + D_2||_{L1})$.

Καθώς ο αλγόριθμος αφαιρεί εξυπηρετητές σε κάθε βήμα του, είναι πιθανό σε κάποιο στάδιο η λίστα επιλογής να καταλήξει άδεια. Σε αυτή τη περίπτωση ο αλγόριθμος εφαρμόζει backtracking, δηλαδή επιστρέφει στο προηγούμενο SoI και χαλαρώνει τις προδιαγραφές έως η λίστα να περιέχει υποψήφιους εξυπηρετητές. Σε συνθήκες πραγματικής λειτουργίας, έχει διαπιστωθεί βέλτιστη δρομολόγηση χωρίς backtracking σε ποσοστό 89%, backtracking για ένα και μοναδικό SoI σε επιπλέον ποσοστό 8%, και ανάγκη για πολλαπλό backtracking σε ποσοστό 3%.

Μια επιπλέον μέριμνα της μεθόδου, προκειμένου να μην εισαχθεί καθυστέρηση λόγω πολυπλοκότητας, είναι η καθιέρωση ενός timeout. Εάν παρέλθει ένας προκαθορισμένος χρόνος και ο αλγόριθμος εξακολουθεί να εκτελείται, τότε επιλέγεται η βέλτιστη έως εκείνη τη στιγμή λύση, χωρίς να αναμένουμε το τέλος της διαδικασίας. Σε συνθήκες πραγματικής λειτουργίας της μεθόδου, τα timeouts παρατηρούνται σε λιγότερο του 0.1% των περιπτώσεων.

3.3.4. Επισκόπηση Μεθόδου

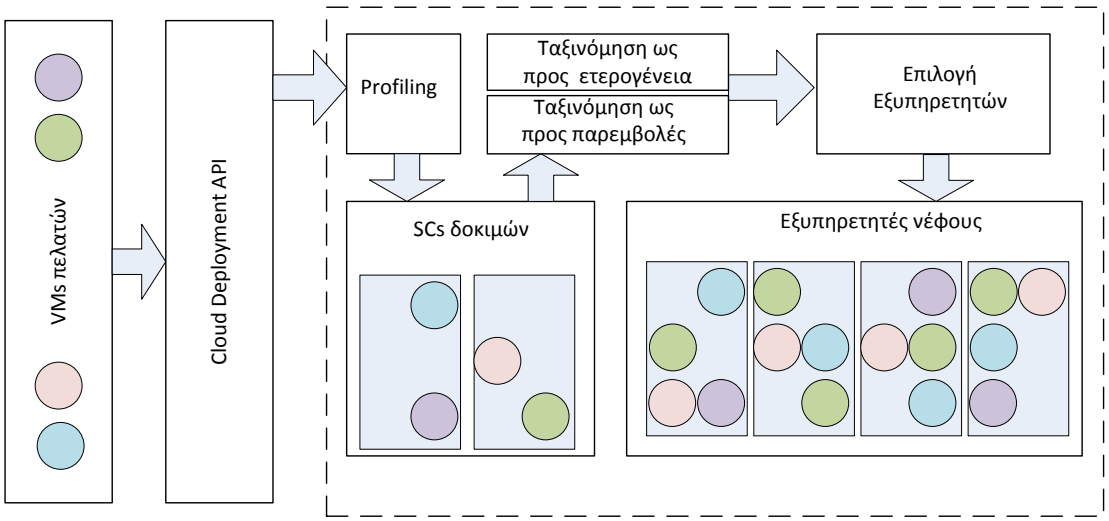
Η μέθοδος Paragon απαιτεί ελάχιστο χρόνο δοκιμαστικής λειτουργίας μιας εισερχόμενης εργασίας.

Το πρώτο στάδιο (Profiling) ουσιαστικά απαιτεί 2 δοκιμές του 1 λεπτού για να ταξινομηθεί μια εργασία ως προς την ετερογένεια, λειτουργώντας την σε 2 διαφορετικά Server Configurations (SCs). Επίσης απαιτεί άλλες 2 δοκιμές του 1 λεπτού για να ταξινομηθεί μια εργασία ως προς την παρεμβολή, λειτουργώντας την παράλληλα με 2 benchmark εργασίες. Εδώ χρησιμοποιείται Server Configuration (SC) με όσο το δυνατόν πιο υψηλές προδιαγραφές, προκειμένου να αποσυσχετίσουμε τα αποτελέσματα από τυχόν επιδράσεις ανεπαρκούς υλικού.

Στη συνέχεια, ακολουθεί το δεύτερο στάδιο (Classification) όπου εφαρμόζονται οι αναλυτικές μέθοδοι που αναφέραμε, οι οποίες διαρκούν ελάχιστα msec, προκειμένου να προκύψουν όλα τα απαιτούμενα σκορ για την σωστή ταξινόμηση της εργασίας ως προς την ετερογένεια και τις παρεμβολές.

Στο τρίτο στάδιο (Scheduling) γίνεται η επιλογή του εξυπηρετητή στον οποίο θα δρομολογηθεί τελικά η εργασία, ο οποίος είναι ο πλέον κατάλληλος, έχοντας λάβει υπόψη την ετερογένεια του περιβάλλοντος και τις πιθανές παρεμβολές από και προς τις ήδη εγκατεστημένες εργασίες.

Στο Σχήμα 10 αναπαριστούμε τα τρία αυτά στάδια της τεχνικής Paragon.



Σχήμα 10 - Αναπαράσταση Τεχνικής Paragon

Για τις ανάγκες της αρχικοποίησης, απαιτούνται περίπου 20 εργασίες σε μια φάρμα 40 εξυπηρετητών, και 50 εργασίες για μια φάρμα 1000 εξυπηρετητών. Καθώς μια νέα εργασία δοκιμάζεται μόνο σε 2 server configurations (SC) και μόνο με 2 benchmarks, μειώνεται ο χρόνος δοκιμής αλλά και η ποσότητα των εξυπηρετητών που πρέπει να διαθέσουμε για αυτή τη διαδικασία. Αυτό δείχνει πως η τεχνική είναι

οικονομική και κλιμακώνεται άνετα σε περιβάλλον μεγάλου data center όπου καταφθάνουν νέες εργασίες με πολύ μεγάλο ρυθμό.

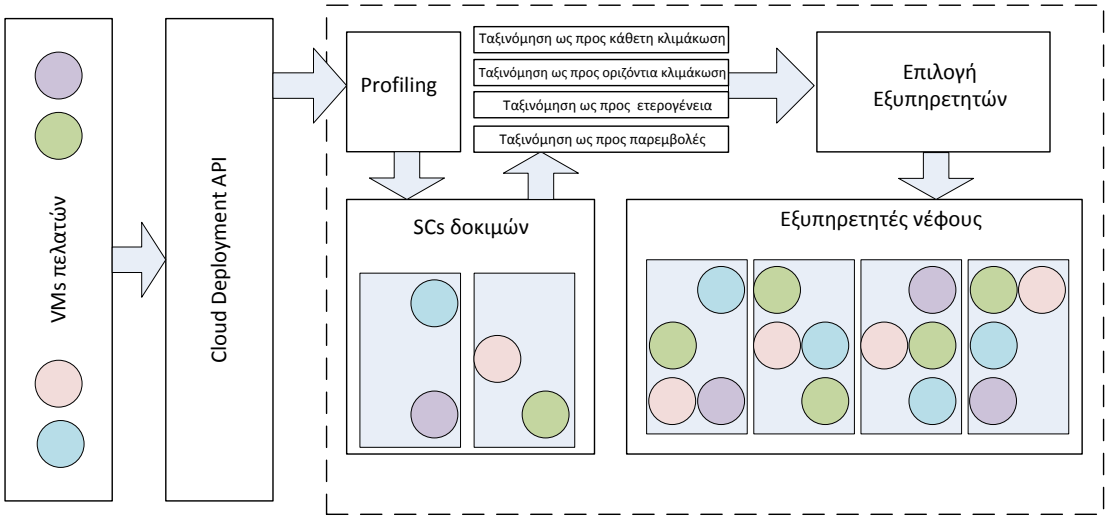
Σε αντίθεση με παραδοσιακές τεχνικές profiling, οι οποίες γίνονται σε συνθήκες πραγματικής λειτουργίας, εδώ το πρώτο στάδιο είναι δοκιμαστικό, δηλαδή ο πελάτης δεν εξυπηρετείται. Ο λόγος που συμβαίνει κάτι τέτοιο, είναι ότι η τεχνική επιδιώκει την παραβίαση του QoS της εργασίας, σε πολύ σύντομο χρονικό διάστημα, προκειμένου να την κατατάξει. Ωστόσο το overhead αυτό μπορεί να θεωρηθεί αμελητέο, ειδικά αν συνυπολογίσουμε το όφελος στο συνολικό χρόνο εκτέλεσης της εργασίας, από το γεγονός ότι επιλέγουμε με μεγάλο ποσοστό επιτυχίας τον βέλτιστο εξυπηρετητή, και συνεπώς η εργασία εκτελείται στο μικρότερο δυνατό χρόνο. Σε πραγματικές συνθήκες λειτουργίας, το overhead του σταδίου Profiling έχει μετρηθεί μικρότερο του 1.2% και του σταδίου Classification μικρότερο του 0.09%.

3.4. Τεχνική Quasar (Q)

Το Quasar [44] υλοποιεί μια τεχνική δρομολόγησης εργασιών που επιχειρεί να αξιοποιήσει στο μέγιστο τους κοινόχρηστους πόρους, ενώ παράλληλα να ικανοποιεί τις προδιαγραφές επίδοσης και QoS που θέτει μια εργασία. Στις περισσότερες τεχνικές, οι πελάτες αιτούνται τη χρήση συγκεκριμένων πόρων για την εργασία τους και με τον τρόπο αυτό δεσμεύουν πόρους του νέφους, ανεξάρτητα αν τελικά θα τους χρησιμοποιήσουν ή όχι. Αυτή η τεχνική απαιτεί από τους πελάτες κάτι διαφορετικό: να ορίσουν απαιτήσεις και περιορισμούς επίδοσης, όπως throughput ή latency, ανάλογα με τον τύπο της εργασίας. Για παράδειγμα, σε latency-critical εργασίες ο περιορισμός ορίζεται σε queries per second (QPS). Σε πλαίσια όπως το Hadoop ή το Spark ο περιορισμός είναι ο χρόνος εκτέλεσης (execution time). Η λειτουργία αυτή επιτρέπει στο Quasar να υπολογίζει και να εντοπίζει τους πόρους που απαιτούνται για να ικανοποιηθούν οι προδιαγραφές της εργασίας. Καθώς ο καθορισμός των απαραίτητων πόρων δεν αφήνεται στον πελάτη, αποφεύγονται φαινόμενα υπερεκτίμησής τους.

Δεύτερο χαρακτηριστικό της τεχνικής είναι ότι χρησιμοποιεί ταχύτατους αλγόριθμους ταξινόμησης, προκειμένου να αντιληφθεί τον αντίκτυπο που θα έχουν διάφορες πιθανές κατανομές της εργασίας σε συστήματα του νέφους. Η τεχνική συνδυάζει μικρό αριθμό από πληροφορίες profiling της ίδιας της εργασίας, και μεγαλύτερο αριθμό από πληροφορίες που προέρχονται από ήδη δρομολογημένες εργασίες. Η ταξινόμηση γίνεται σε τέσσερα επίπεδα: ως προς την ετερογένεια (ποιες διαμορφώσεις εξυπηρετητών είναι κατάλληλες για την εργασία), ως προς την

παρεμβολή (ποιες εργασίες είναι κατάλληλες προς συστέγαση), ως προς την κάθετη κλιμάκωση (ποια η απαιτούμενη ποσότητα πόρων ανά εξυπηρετητή) και ως προς την οριζόντια κλιμάκωση (ποιος ο απαιτούμενος αριθμός εξυπηρετητών ανά εργασία). Καθώς το πρόβλημα αποσυντίθεται σε τέσσερα επί μέρους, απλοποιείται σημαντικά.



Σχήμα 11 - Αναπαράσταση Τεχνικής Quasar

Τρίτο χαρακτηριστικό της τεχνικής είναι ότι κατανέμει και εκχωρεί τους απαιτούμενους πόρους αξιοποιώντας τα αποτελέσματα της ταξινόμησης, ταυτόχρονα όμως παρακολουθεί την επίδοση της εργασίας. Ένας άπληστος αλγόριθμος συνδυάζει τα αποτελέσματα των τεσσάρων ανεξάρτητων ταξινομήσεων και επιλέγει τους κατάλληλους πόρους που θα ικανοποιήσουν ή θα προσεγγίσουν όσο το δυνατόν καλύτερα τις απαιτήσεις. Το Quasar παρακολουθεί τις επιδόσεις της εργασίας. Εάν παρατηρηθεί απόκλιση από τα όρια που έχουν τεθεί ή εάν οι πόροι υποχρησιμοποιούνται, η τεχνική ταξινομεί εκ νέου την εργασία και την δρομολογεί σε πιο κατάλληλους εξυπηρετητές προκειμένου να επαναφέρει τις επιδόσεις της εντός προδιαγραφών ή να εξοικονομήσει πόρους. Αυτό μπορεί να συμβεί είτε γιατί άλλαξε η συμπεριφορά της εργασίας, είτε γιατί έγινε ανεπιτυχής ταξινόμηση, είτε γιατί ο άπληστος αλγόριθμος κατέληξε σε μη-βέλτιστο αποτέλεσμα. Σε κάθε περίπτωση η τεχνική είναι σε θέση να επαναλάβει τη δρομολόγηση και να βελτιώσει τις επιδόσεις της εργασίας.

Σε σύγκριση με την τεχνική Paragon, επισημαίνουμε ότι η τεχνική Quasar καλύπτει μεγαλύτερο εύρος της διαδικασίας δρομολόγησης εργασιών. Η τεχνική Paragon προϋποθέτει ότι οι προδιαγραφές μιας εργασίας (όπως οι απαιτήσεις της σε πόρους) έχουν ήδη τεθεί με κάποιο τρόπο. Χειρίζεται μόνο την δρομολόγηση και την ανάθεση στους πλέον κατάλληλους πόρους. Η τεχνική Quasar έχει την

ικανότητα να υπολογίσει τους απαιτούμενους πόρους και έτσι να οδηγήσει σε βέλτιστη αξιοποίηση του συστήματος στο οποίο θα δρομολογηθεί η εργασία. Επιπλέον, η τεχνική Quasar περιέχει την έννοια της ανάδρασης και μπορεί να βελτιώσει μια δρομολόγηση που κρίνεται ως μη-βέλτιστη. Ο τρόπος που επιτυγχάνεται η ανάδραση είναι σχετικά απλός. Όταν οι πραγματικές επιδόσεις μιας εργασίας σε ένα συγκεκριμένο περιβάλλον λειτουργίας διαφέρουν αισθητά από τις εκτιμώμενες, αντικαθίσταται στους πίνακες ταξινόμησης η εκτιμώμενη τιμή από την πραγματική, και με βάση αυτή την αλλαγή εκτελείται εκ νέου η διαδικασία επιλογής εξυπηρετητή.

3.4.1. Ταξινόμηση ως προς την ετερογένεια και παρεμβολή

Σε αυτές τις ταξινομήσεις η μέθοδος είναι πανομοιότυπη με όσα περιγράψαμε στην τεχνική Paragon [28]. Κατά την αρχικοποίηση της μεθόδου, εκτελείται εκτεταμένο Profiling σε μερικές εργασίες, προκειμένου να αποτελέσουν τη βάση για την κατηγοριοποίηση και κατάταξη των υπολοίπων εργασιών. Για μια νέα εισερχόμενη εργασία εκτελείται Profiling για ένα λεπτό, σε 2 από τους ετερογενείς εξυπηρετητές με και χωρίς παρεμβολές για 2 κοινόχρηστους πόρους. Στα αποτελέσματα εφαρμόζεται Collaborative Filtering προκειμένου να εκτιμηθούν οι επιδόσεις της εργασίας στους υπόλοιπους εξυπηρετητές, καθώς και η επίπτωση των παρεμβολών σε ότι αφορά τους υπόλοιπους κοινόχρηστους πόρους. Για περισσότερες λεπτομέρειες παραπέμπουμε στις παραγράφους 3.3.1. και 3.3.2.

3.4.2. Ταξινόμηση ως προς την κάθετη κλιμάκωση (scale-up)

Αυτή η ταξινόμηση προσδιορίζει πως επηρεάζεται η επίδοση μιας εργασίας, από την ποσότητα των πόρων που διατίθενται για την εργασία αυτή σε έναν εξυπηρετητή. Η μέθοδος στη παρούσα φάση εξετάζει μόνο cores, μνήμη και αποθηκευτικό χώρο, ενώ στο μέλλον θα επεκταθεί σε άλλους πόρους. Η διαδικασία ταξινόμησης εκτελείται στον εξυπηρετητή με τις υψηλότερες προδιαγραφές και δυνατότητες κλιμάκωσης. Μια εργασία δοκιμάζεται για μικρό χρόνο σε δύο τυχαίες διαφορετικές διαμορφώσεις πόρων (scale-up configurations). Το profiling συλλέγει μετρήσεις της επίδοσης, σε μονάδες παρόμοιες με τους στόχους που έχουν τεθεί, για παράδειγμα Expected Completion Time ή Queries Per Second (QPS) και εισάγει αυτές τις μετρήσεις σε πίνακα $A_{m,n}$ με m γραμμές, μια για κάθε εργασία, και n στήλες, μια για κάθε διαμόρφωση πόρων.

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

Τα άγνωστα στοιχεία της γραμμής που αφορά την εργασία προς εξέταση παράγονται και συμπληρώνονται με τεχνικές Collaborative Filtering, ώστε τελικά να έχουμε γνωστές μετρήσεις για την επίδοση της εργασίας σε όλες τις πιθανές διαμορφώσεις πόρων.

Οι παράμετροι και ο χρόνος που απαιτείται για το Profiling εξαρτάται από τον τύπο της εργασίας. Για παράδειγμα, σε εργασίες Hadoop το profiling εκτελείται σε 2 διαφορετικές διαμορφώσεις υλικού, με διαφορετικές ρυθμίσεις στις βασικές παραμέτρους του πλαισίου (για παράδειγμα mappers per node, JVM heapsize, block size, memory per task, replication factor, compression). Το profiling διαρκεί έως να υπάρχει μια καλή εκτίμηση του χρόνου ολοκλήρωσης, με βάση τον ρυθμό εκτέλεσης. Συνήθως αυτό επιτυγχάνεται όταν τα map tasks φτάσουν στο 20% του χρόνου ολοκλήρωσης.

3.4.3. Ταξινόμηση ως προς την οριζόντια κλιμάκωση (scale-out)

Αυτή η ταξινόμηση αφορά μόνο εργασίες που μπορούν να χρησιμοποιήσουν πολλαπλούς εξυπηρετητές, όπως συμβαίνει με κατανεμημένα πλαίσια (Spark, Hadoop), με Web services, Cassandra (distributed database system) κλπ. Η ταξινόμηση προσδιορίζει πως επηρεάζεται η επίδοση μιας εργασίας, από την προσθήκη επιπλέον εξυπηρετητών. Η μέθοδος δεν διαφέρει από την ταξινόμηση που περιγράψαμε νωρίτερα για single-node διαμορφώσεις. Εδώ χρησιμοποιούνται πολλοί εξυπηρετητές (scale-out) με πανομοιότυπη διαμόρφωση, ώστε να αποσυσχετίσουμε το αποτέλεσμα από την επίπτωση διαφορετικών διαμορφώσεων (scale-up). Σε αυτή τη μέθοδο, εξακολουθούμε να έχουμε μια γραμμή για κάθε εργασία στον πίνακα $A_{m,n}$, και μια στήλη για κάθε διαφορετικό αριθμό εξυπηρετητών.

Το Profiling γίνεται κατά τα γνωστά για 2 διαφορετικές διαμορφώσεις (δύο δοκιμές σε διαφορετικό αριθμό εξυπηρετητών). Τεχνικές Collaborative Filtering ανακτούν τις καταχωρήσεις του πίνακα που λείπουν, προκειμένου να εκτιμηθούν οι επιδόσεις της εργασίας σε όλες τις διαμορφώσεις (διαφορετικούς αριθμούς εξυπηρετητών). Η διαδικασία profiling εκτελείται σε μικρό αριθμό εξυπηρετητών (έναν έως τέσσερις) ώστε να εξοικονομηθούν πόροι αλλά και για να είναι σύντομος ο χρόνος εκτέλεσης της διαδικασίας. Κατά την αρχικοποίηση όμως της μεθόδου, ένας αριθμός εργασιών

(20-30) δοκιμάζεται σε μεγάλο αριθμό εξυπηρετητών (για παράδειγμα σε 1 – 100 nodes) ώστε να διαθέτουμε στοιχεία για μεγάλες συστοιχίες. Το βήμα αυτό δεν χρειάζεται να επαναληφθεί κατά τη διάρκεια κανονικής λειτουργίας, εκτός εάν προκύψουν σημαντικές αλλαγές στη διαμόρφωση των συστοιχιών.

3.4.4. Μηχανισμός επιλογής εξυπηρετητών

Το αποτέλεσμα των παραπάνω ταξινομήσεων τροφοδοτεί έναν απλό άπληστο αλγόριθμο, ο οποίος αποφασίζει για την ποσότητα, τον τύπο και την ακριβή διαμόρφωση των πόρων που θα διατεθούν για την εργασία. Στόχος είναι να διαθέσει τον μικρότερο δυνατό αριθμό πόρων, που μπορούν να καλύψουν τις δοθείσες προδιαγραφές επίδοσης.

Ο μηχανισμός αρχικά χρησιμοποιεί το αποτέλεσμα της ταξινόμησης ώστε να τοποθετήσει τους εξυπηρετητές σε φθίνουσα σειρά ως προς την ποιότητα των πόρων που διαθέτουν, δηλαδή προηγούνται οι εξυπηρετητές υψηλών επιδόσεων με τον χαμηλότερο δείκτη παρεμβολών. Στη συνέχεια προσδιορίζει τους πόρους που πρέπει να διατεθούν, ώστε να ικανοποιηθούν οι προδιαγραφές επιδόσεων. Για παράδειγμα, αν ένας web server έχει προδιαγραφεί να αποδίδει 100K QPS και ο πρώτος στη λίστα εξυπηρετητής μπορεί να επιτύχει 20K QPS, η εργασία θα απαιτούσε 5 τέτοιους εξυπηρετητές. Εάν ο αριθμός είναι διαθέσιμος, ο αλγόριθμος θα υποδείξει αυτή τη λύση. Εάν ο αριθμός δεν είναι διαθέσιμος, ο αλγόριθμος θα κατέβει να επιλέξει εξυπηρετητές χαμηλότερων προδιαγραφών και θα αυξήσει κατάλληλα τον αριθμό τους. Σε αυτή τη διαδικασία ο αλγόριθμος προτιμά πρώτα να αυξήσει τους πόρους ανά εξυπηρετητή (scale-up) και έπειτα να αυξήσει τον αριθμό εξυπηρετητών (scale-out), σε μια προσπάθεια να συγκεντρώσει όσο το δυνατόν τους πόρους που διατίθενται σε κάθε εργασία. Παρόλα αυτά, είναι εφικτό να επιλεγεί μια διαφορετική στρατηγική, από παρόχους που επιθυμούν να κατανείμουν με άλλη λογική τις εργασίες, στοχεύοντας για παράδειγμα να παρέχουν υπηρεσίες αυξημένης διαθεσιμότητας ή γεωγραφικής διασποράς.

3.4.5. Επισκόπηση μεθόδου

Για κάθε εισερχόμενη εργασία η τεχνική Quasar συλλέγει δεδομένα profiling ως προς την κάθετη και οριζόντια κλιμάκωση, την ετερογένεια και την παρεμβολή. Αυτό απαιτεί έως τέσσερις δοκιμές που εκτελούνται παράλληλα. Το overhead αυτής της διαδικασίας εξαρτάται από τον τύπο της εργασίας αλλά σε κάθε περίπτωση δεν υπερβαίνει τα 5 λεπτά. Μόλις οι μετρήσεις profiling είναι διαθέσιμες, οι αναλυτικές

μέθοδοι που περιγράψαμε παρέχουν πλήρη εικόνα για την ταξινόμηση της εργασίας. Στη συνέχεια ο αλγόριθμος δρομολογεί την εργασία στους κατάλληλους πόρους. Η τεχνική διατηρεί μια κατάσταση (state) ανά εργασία και ανά εξυπηρετητή. Η κατάσταση ανά εργασία περιέχει τα στοιχεία ταξινόμησης. Η κατάσταση ανά εξυπηρετητή περιέχει πληροφορίες για τις εγκατεστημένες εργασίες και την αθροιστική τους παρεμβολή. Αυτή η κατάσταση ανανεώνεται κάθε φορά που δρομολογείται μια νέα εργασία στον συγκεκριμένο εξυπηρετητή.

Ο Πινάκας 4 περιέχει μια πειραματική αξιολόγηση της τεχνικής, χρησιμοποιώντας συστοιχία 40 εξυπηρετητών και εργασίες από Hadoop (10 data mining jobs), latency-critical υπηρεσίες (10 memcached jobs και 10 Apache webserver loads) και 413 single-node benchmark workloads (SPEC, PARSEC, SPLASH-2, BioParallel, Minebench, SpecJbb).

Default density constraint: 2 entries per row, per classification												
	scale-up			**scale-out**			**heterogeneity**			**interference**		
Classification err.	*avg*	*90 %ile*	*max*	*avg*	*90 %ile*	*max*	*avg*	*90 %ile*	*max*	*avg*	*90 %ile*	*max*
Hadoop (10 Jobs)	5.2%	9.8%	11%	5.0%	14.5%	17%	4.1%	4.6%	5.0%	1.8%	5.1%	6%
Memcached (10)	6.3%	9.2%	11%	6.6%	10.5%	12%	5.2%	5.7%	6.5%	7.2%	9.1%	10%
Webserver (10)	8.0%	10.1%	13%	7.5%	11.6%	14%	4.1%	5.1%	5.2%	3.2%	8.1%	9%
Single-node (413)	4.0%	8.1%	9%	-	-	-	3.5%	6.9%	8.0%	4.4%	9.2%	10%

Πίνακας 4 - Μετρικές για την αξιολόγηση της τεχνικής Quasar

Ο πίνακας δείχνει τα σφάλματα, συγκρίνοντας την εκτιμώμενη επίδοση των εργασιών με βάση τις τεχνικές ταξινόμησης, και την πραγματική επίδοση των εργασιών όπως μετρήθηκε για τις ανάγκες της αξιολόγησης. Κατά μέσο όρο τα σφάλματα είναι λιγότερα από 8% σε όλες τις κατηγορίες εργασιών, και τα μέγιστα λάθη είναι κάτω του 17%.

Σε αντίθεση με παραδοσιακές τεχνικές που απαιτούν από τον πελάτη να ορίσει και να δεσμεύσει τους επιθυμητούς πόρους οδηγώντας πολλές φορές σε σπατάλη τους, η τεχνική Quasar απαιτεί να οριστεί η επιθυμητή επίδοση της εργασίας, εφόσον ανήκει σε συγκεκριμένες κατηγορίες εργασιών που υποστηρίζονται (distributed analytical frameworks, εφαρμογές web serving, NoSQL datastores, single-node batch workloads). Ο άπληστος αλγόριθμος που χρησιμοποιείται, φροντίζει με μεγάλο ποσοστό επιτυχίας να διαθέσει τους ελάχιστους δυνατούς πόρους για την εξυπηρέτηση της εργασίας. Λαμβάνει υπόψη την επίπτωση της ποσότητας των πόρων που διατίθενται (scale-up και scale-out), την επίπτωση της ετερογένεια και την επίπτωση των παρεμβολών. Ταυτόχρονα η τεχνική διαθέτει μηχανισμό ανάδρασης, προκειμένου να διορθώσει τυχόν λανθασμένες δρομολογήσεις.

4. Ποιοτική σύγκριση τεχνικών δρομολόγησης και τελικά συμπεράσματα

Στην εργασία αυτή δείξαμε ότι οι παραδοσιακές τεχνικές δρομολόγησης εργασιών, που βασίζονται αποκλειστικά στην απαίτηση του πελάτη για δέσμευση συγκεκριμένων πόρων, οδηγούν σε μη-βέλτιστες κατανομές των εργασιών στο νέφος. Προκειμένου να ανταποκριθούν στις απαιτήσεις ποιότητας που θέτουν οι πελάτες, οι πάροχοι υπηρεσιών νέφους είναι υποχρεωμένοι να διατηρούν αναξιοποίητο ένα υψηλό ποσοστό πόρων, της τάξης του 50% ή παραπάνω, σαν μια ασφαλιστική δικλείδα που επιτρέπει την διάθεση επιπλέον πόρων ή επιτρέπει μια εναλλακτική δρομολόγηση των εργασιών, σε περίπτωση που η επιλεχθείσα κατανομή αποδειχθεί ανεπαρκής. Το γεγονός αυτό έχει αρκετές συνέπειες: Ένα αχρείαστο, αρνητικό οικολογικό αποτύπωμα, καθώς και αυξημένο κόστος λειτουργίας που μοιραία μετακυλύεται στον πελάτη.

Καθώς η αρχική δρομολόγηση των εργασιών γίνεται με τεχνικές που εκτιμούν τις ονομαστικές και όχι τις πραγματικές ανάγκες σε υπολογιστικούς πόρους, δημιουργείται μια ανάγκη διαρκούς παρακολούθησης της επίδοσης των εργασιών, ώστε να αναγνωρίζονται τυχόν αποκλίσεις από τις προβλεπόμενες ή αποδεκτές επιδόσεις, να επανεκτιμούνται οι αρχικές επιλογές δρομολόγησης και περιοδικά οι εργασίες να δρομολογούνται σε νέους εξυπηρετητές, προκειμένου να επιτευχθεί βελτίωση της λειτουργίας τους αλλά και της συνολικής λειτουργίας του νέφους. Η διαδικασία επανεκτίμησης της δρομολόγησης περιγράφεται συχνά με τους όρους migration, resizing ή consolidation, ανάλογα με την ανάγκη που εξυπηρετεί, και έχει ένα επιπλέον κόστος λειτουργίας. Εύλογα δημιουργείται η απορία, αν είναι δυνατόν να προβλεφθεί εκ των προτέρων και όσο το δυνατόν με μεγαλύτερη ακρίβεια η συμπεριφορά των εργασιών, να ληφθεί υπόψη κατά την πρώτη δρομολόγηση, να επιτευχθεί μια καλύτερη αρχική κατανομή και να ελαχιστοποιηθούν οι αλλαγές εκ των υστέρων.

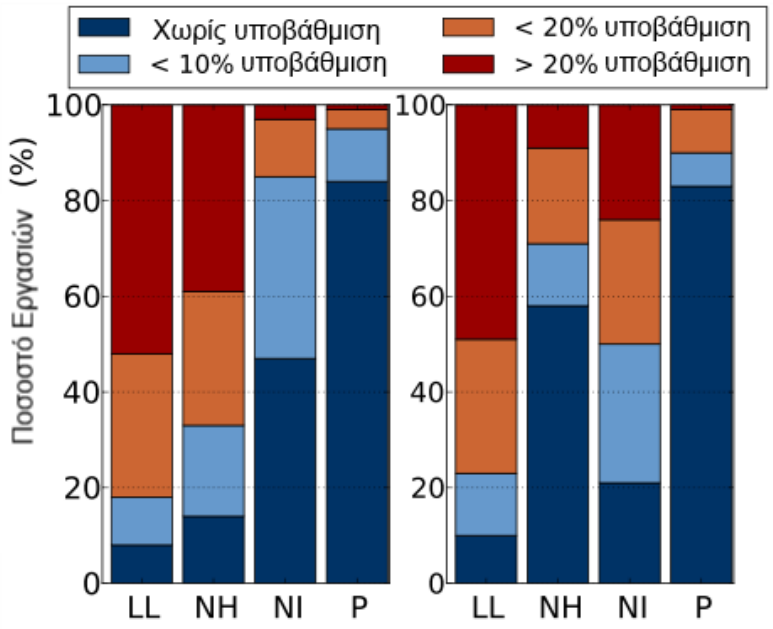
Αναφερθήκαμε σε νέες τεχνικές που βρίσκονται στο στάδιο της έρευνας, με στόχο να βελτιώσουν την αρχική δρομολόγηση των εργασιών, να τοποθετήσουν την κάθε εργασία στη πλέον κατάλληλη συστοιχία και εξυπηρετητή, και να οδηγήσουν σε μια πιο συμπαγή κατανομή χρησιμοποιώντας λιγότερο υλικό. Οι τεχνικές αυτές λαμβάνουν υπόψη την ετερογένεια των σύγχρονων data centers, τις παρεμβολές που θα εμφανιστούν μεταξύ εργασιών που πρόκειται να συστεγαστούν, και τέλος τη βελτίωση της επίδοσης των εργασιών, που μπορεί να προκύψει από την οριζόντια ή κάθετη κλιμάκωση, εφόσον υπάρχει διαθέσιμο επιπλέον υλικό. Το όφελος που αποτυπώνεται στις πειραματικές εφαρμογές αυτών των μεθόδων είναι σημαντικό: 22% όφελος κατά μέσο όρο αν ληφθεί υπόψη η ετερογένεια, 34%

όφελος αν ληφθούν υπόψη οι παρεμβολές. Όσο για το όφελος από την οριζόντια ή κάθετη κλιμάκωση, είναι προφανές ότι οδηγεί σε ένα νέο μοντέλο διάθεσης των υπηρεσιών νέφους, που απαιτεί ιδιαίτερη συζήτηση.

Το συνδυασμένο όφελος από την χρήση αυτών των τεχνικών φαίνεται πειραματικά να είναι υψηλό. Η αύξηση της χρήσης των εξυπηρετητών (utilization) κυμαίνεται μεταξύ 50% έως 90% χωρίς ουσιαστική επίπτωση στο QoS των εργασιών, ενώ οι γρήγοροι αλγόριθμοί τους φαίνεται να έχουν μεγάλη ακρίβεια πρόβλεψης. Έχουν ελάχιστη απόκλιση (1% έως 2%) από το αποτέλεσμα που θα πετύχαινε μια εξαντλητική δοκιμή όλων των πιθανών κατανομών εργασιών στους διαθέσιμους εξυπηρετητές.

Το ερώτημα που γεννάται, είναι αν οι νέες τεχνικές είναι αρκετά στιβαρές για να αξιοποιηθούν στη παραγωγή, ιδίως σε μεγάλα υπολογιστικά νέφη με χιλιάδες εξυπηρετητές και δεκάδες χιλιάδες εργασίες.

Στο Σχήμα 12 φαίνεται η επίπτωση τεσσάρων τεχνικών δρομολόγησης, για την ετερογένεια (αριστερά) και την παρεμβολή (δεξιά). Η τεχνική Paragon (P) συγκρίνεται με μια τεχνική παρόμοια με την τεχνική PACMan, που δεν λαμβάνει υπόψη την ετερογένεια (NH), μια τεχνική που δεν λαμβάνει υπόψη την παρεμβολή (NI) και μια απλή τεχνική Least Loaded (LL) που δεν λαμβάνει υπόψη ούτε την ετερογένεια ούτε την παρεμβολή.



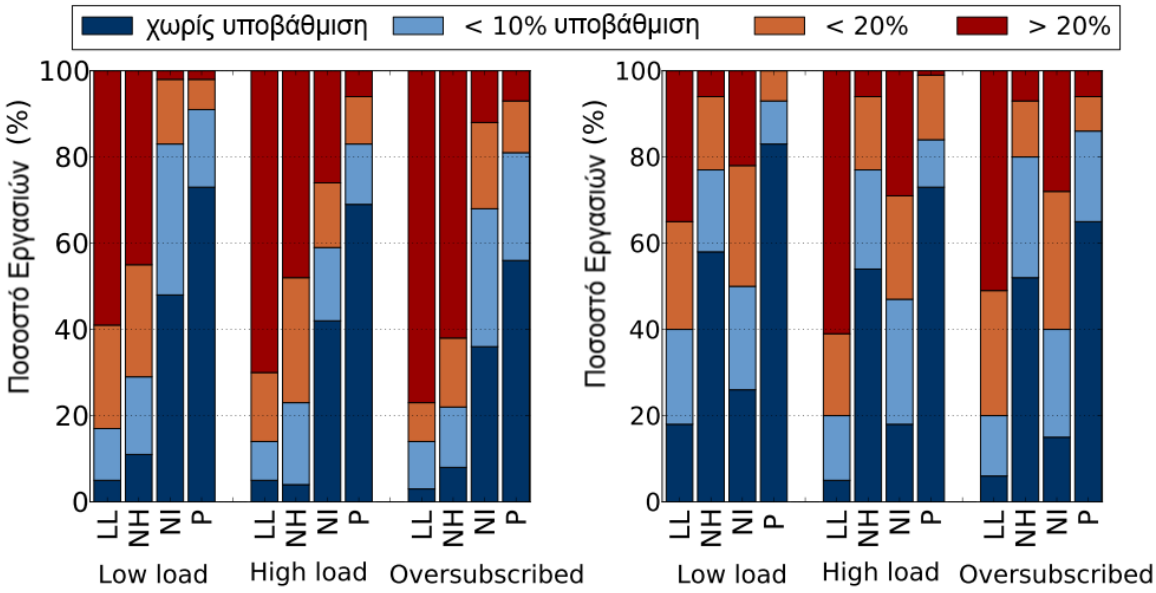
Σχήμα 12 - Σύγκριση της τεχνικής Paragon (P) με άλλες τεχνικές δρομολόγησης

Ως προς την επιλογή πλατφόρμας σε ένα ετερογενές περιβάλλον, η τεχνική LL (που δεν λαμβάνει υπόψη την ετερογένεια) δρομολογεί τις εργασίες σε μη-βέλτιστα Server Configurations σε μεγάλο ποσοστό. Η τεχνική NH (που επίσης δεν λαμβάνει υπόψη την ετερογένεια) δρομολογεί τις εργασίες σε κάπως καλύτερο Server Configuration προσπαθώντας να ικανοποιήσει τις προδιαγραφές παρεμβολής. Οι τεχνικές που λαμβάνουν υπόψη την ετερογένεια (NI και P) πετυχαίνουν σαφώς καλύτερα αποτελέσματα. Ο λόγος που η τεχνική NI αστοχεί μετά το 50% των εργασιών, είναι ότι επιλέγει διαρκώς τα καλύτερα Server Configurations χωρίς να λαμβάνει υπόψη τις παρεμβολές, οπότε σταδιακά οδηγεί τους εξυπηρετητές σε κορεσμό, οι παρεμβολές επιδρούν καταστροφικά και οι εργασίες υφίστανται σημαντικές επιπτώσεις στην επίδοσή τους.

Αντίστοιχη εικόνα έχουμε στις αποφάσεις που παίρνουν οι τεχνικές σε ότι αφορά τις παρεμβολές. Οι τεχνικές που τις λαμβάνουν υπόψη (NH και P) οδηγούν σε καλύτερες κατανομές από τις υπόλοιπες δύο (LL, NI). Είναι επίσης προφανές ότι η τεχνική Paragon (P) οδηγεί σε συνολικά βέλτιστη κατανομή καθώς λαμβάνει υπόψη και τις δύο παραμέτρους.

Η τεχνική Paragon είναι μια ώριμη και στιβαρή τεχνική, καθώς είναι σχεδιασμένη με στόχο να αξιοποιηθεί σε μεγάλες εγκαταστάσεις, με μικρό overhead, μικρό χρόνο απόφασης και χαμηλό κόστος λειτουργίας. Για να δοκιμαστεί η αποτελεσματικότητά της, δημιουργήθηκε περιβάλλον 1000 εξυπηρετητών στο νέφος της Amazon. Για τις δοκιμές ως προς την ετερογένεια χρησιμοποιήθηκαν 14 διαφορετικά Server Configurations (SC), με διάφορα μεγέθη και ταχύτητες μνήμης, αλλά και επεξεργαστές διαφόρων γενιών, αρχιτεκτονικών (Atom, Xeon, Opteron) και ταχυτήτων. Ανάλογες δοκιμές έχουν γίνει σε περιβάλλον Windows Azure (500 Servers / 8 SC) και Google Compute Engine (100 Servers / 4 SC).

Η αποτελεσματικότητά της ως προς την παρεμβολή έχει δοκιμαστεί επιλέγοντας μεγάλο αριθμό δοκιμαστικών εργασιών (single threaded, multi-threaded, multi-programmed, i/o bound benchmark workloads από τις σουίτες SPEC CPU2006, PARSEC, SPLASH-2, BioParallel, Minebench, SPECjbb, Hadoop, Matlab κλπ) σε τρία σενάρια: low-load με 2500 δοκιμαστικές εργασίες που υποβάλλονται ανά 1 δευτερόλεπτο, high-load με 5000 εργασίες που υποβάλλονται ανά 1 δευτερόλεπτο, και oversubscribed με 7500 εργασίες που υποβάλλονται ανά 1 δευτερόλεπτο και επιπλέον 1000 εργασίες που υποβάλλονται μαζικά ανά 0,1 δευτερόλεπτο στη μέση της δοκιμής. Στο Σχήμα 13 δείχνουμε τη σύγκριση της τεχνικής Paragon με άλλες τεχνικές (LL, NH, NI) για αυτά τα τρία σενάρια. Ήδη από το low-load σενάριο εμφανίζονται οφέλη από τη χρήση της τεχνικής, ενώ προχωρώντας προς το oversubscribed σενάριο οι υπόλοιπες τεχνικές επιδεινώνουν δραματικά τις επιδόσεις των εργασιών.



Σχήμα 13 - Δοκιμή τεχνικής Paragon (P) σε 3 διαφορετικά σενάρια

Σε ότι αφορά την τεχνική Quasar, είναι σημαντικό να πούμε ότι βασίζεται ουσιαστικά στην τεχνική Paragon (άρα φέρει όλα τα οφέλη της) και επιπλέον λαμβάνει υπόψη τα οφέλη που μπορεί να προκύψουν από την οριζόντια ή κάθετη κλιμάκωση. Εδώ όμως διαφοροποιείται σημαντικά, καθώς δεν έχει μελετηθεί επαρκώς για όλα τα διαφορετικά είδη εργασιών που μπορούν να επωφεληθούν μιας κλιμάκωσης. Μπορεί να αξιοποιηθεί μόνο σε περιορισμένο προφίλ εργασιών (πχ για εργασίες που λειτουργούν στα πλαίσια Hadoop, Spark ή Storm, σε κατανεμημένες εφαρμογές webserving και σε NoSQL datastores), επομένως απαιτείται να λειτουργήσει σε μια ξεχωριστή νησίδα του νέφους που θα εξυπηρετεί μόνο συμβατές εργασίες.

Απαιτεί επίσης ένα νέο μοντέλο καταγραφής των αρχικών απαιτήσεων της κάθε εργασίας. Όλες οι υπόλοιπες τεχνικές που περιγράψαμε, εξακολουθούν να βασίζονται σε ένα μοντέλο διάθεσης υπηρεσιών νέφους, όπου ο πελάτης αρχικά καλείται να καθορίσει τον αριθμό των απαιτούμενων πόρων και να καταβάλλει κάποιο τίμημα γι αυτούς. Ο Μηχανισμός Consolidation καλείται να τους δεσμεύσει και να τους αναθέσει στην εργασία, με λιγότερο ή περισσότερο αποδοτικό τρόπο. Πιο αποδοτικός τρόπος, σημαίνει ότι το datacenter αυξάνει τις οικονομίες κλίμακας και μπορεί να διαθέσει τις υπηρεσίες του φθηνότερα. Τελικά όμως οι πόροι που διατίθενται είναι αυτοί που ζήτησε και δέσμευσε ο πελάτης. Αντίθετα, μια τεχνική σαν την Quasar που λαμβάνει υπόψη τις δυνατότητες οριζόντιας ή κάθετης κλιμάκωσης μιας εργασίας, οδηγεί αναγκαστικά σε ένα νέο μοντέλο. Ο πελάτης

πρέπει να θέσει απαιτήσεις και περιορισμούς που έχουν να κάνουν με την επίδοση της εργασίας, και ο Μηχανισμός Consolidation είναι ελεύθερος να επιλέξει το σύνολο των πόρων που μπορούν να εξυπηρετήσουν την απαίτηση.

Εκ πρώτης όψης η διάθεση υπηρεσιών με αυτό το μοντέλο ακούγεται πιο λογική. Ο πελάτης, ανάλογα με το είδος της εργασίας του, καλείται να θέσει απαιτήσεις που έχουν να κάνουν με επιδόσεις (πχ latency ή throughput). Ενδεικτικά, ορίζει τα queries per second που πρέπει να εξυπηρετεί η εργασία, ή τον χρόνο εκτέλεσης και ολοκλήρωσής της, χωρίς να χρειάζεται να μεταφράσει αυτήν την απαίτηση σε cores, μνήμη, δίσκο και λοιπούς υπολογιστικούς πόρους. Στη πράξη όμως, η αγορά υπηρεσιών νέφους δεν λειτουργεί με αυτή τη λογική. Η αγορά είναι εκπαιδευμένη στην απλοϊκή λογική της διάθεσης υπηρεσιών με τιμή ανά core, ανά MB μνήμης, ανά GB αποθηκευτικού χώρου κλπ. Μάλιστα, υπάρχει περαιτέρω διαφοροποίηση των υπηρεσιών ανάλογα με τα τεχνικά χαρακτηριστικά των πόρων, για παράδειγμα οι ταχύτεροι δίσκοι SSD μπορεί να αποτιμώνται ακριβότερα.

Με λίγα λόγια, το κόστος ενός εξυπηρετητή επιμερίζεται στους πόρους που τον απαρτίζουν και στο χρόνο απόσβεσης της επένδυσης. Με τη σειρά τους, οι πόροι αυτοί διατίθενται με κάποιο περιθώριο κέρδους. Στην οικονομική άσκηση έχει συνυπολογιστεί τόσο το overselling, όσο και η αναγκαιότητα για ελεύθερους πόρους που θα είναι άεργοι αλλά διαθέσιμοι ώστε να αντιμετωπίσουν πιθανά προβλήματα στη διαθεσιμότητα ή στις επιδόσεις των εγκατεστημένων εργασιών. Σε κάθε περίπτωση, η οικονομική άσκηση είναι θετική: Η χρήση υπηρεσιών νέφους είναι πιο συμφέρουσα, για όλους τους λόγους που εξηγήσαμε στην παράγραφο 1.4. Ωστόσο αποδεικνύεται ότι αν πάμε σε ένα νέο μοντέλο που διευκολύνει την αξιοποίηση των δυνατοτήτων οριζόντιας ή κάθετης κλιμάκωσης, θα υπάρξει περαιτέρω εξοικονόμηση κόστους. Το νέο αυτό μοντέλο προμήθειας υπηρεσιών νέφους αποτελεί σημαντική πρόκληση: Μπορεί να γίνει κατανοητό και αποδεκτό από την αγορά;

Όπως συμβαίνει συνήθως σε κάθε νέο μοντέλο, απαιτείται σταδιακή προσαρμογή και ενδεχομένως μια μεταβατική περίοδος στην οποία ο πελάτης θα διαπιστώσει τα οφέλη, ιδίως αν οδηγούν σε οικονομικότερες λύσεις. Μια ιδέα θα ήταν ο πελάτης να επιλέγει όπως σήμερα ένα αρχικό configuration, και εν συνεχεία το περιβάλλον νέφους να του προτείνει επιπλέον λύσεις υψηλότερης κλιμάκωσης, προσφέροντάς του δωρεάν ένα μέρος της ελεύθερης υποδομή που προβλέπεται να είναι ούτως ή άλλως άεργη κατά το χρόνο εκτέλεσης της εργασίας του. Η μέθοδος αυτή προκρίνεται ιδίως σε εργασίες που εκτελούνται σε περιορισμένο χρόνο. Η διάθεση επιπλέον πόρων επισπεύδει την εκτέλεση, και απελευθερώνει νωρίτερα το νέφος ώστε να είναι σε θέση να εξυπηρετήσει νέους πελάτες.

Μια δεύτερη ιδέα είναι ο πελάτης, ανάμεσα στις άλλες απαιτήσεις και περιορισμούς που θέτει και έχουν να κάνουν με την επίδοση της εργασίας του, να θέτει και περιορισμούς στο κόστος της προσφερόμενης λύσης. Ο περιορισμός κόστους θα λειτουργήσει ως όριο για την ποσότητα των πόρων που μπορούν να διατεθούν. Ο Μηχανισμός Consolidation θα πρέπει να λαμβάνει υπόψη και αυτόν τον περιορισμό. Μέχρι στιγμής ένας τέτοιος αλγόριθμος δεν έχει υλοποιηθεί στην τεχνική Quasar.

Τρίτη ιδέα είναι το περιβάλλον νέφους να κάνει μια αντιπροσφορά, προτείνοντας στον πελάτη ένα διαφορετικό σύνολο πόρων, που μπορούν να ανταποκριθούν εξίσου καλά στις απαιτήσεις της εργασίας του, αλλά εξυπηρετούν καλύτερα τον πάροχο υπηρεσιών νέφους καθώς κατανέμουν καλύτερα την εργασία στις διαθέσιμες συστοιχίες.

Σταδιακά, αξιοποιώντας κάποιες ή όλες από τις παραπάνω ιδέες, οι πελάτες θα συνηθίσουν να θέτουν τις απαιτήσεις τους περιγράφοντας επιθυμητές επιδόσεις και όχι επιθυμητούς πόρους.

Κλείνοντας αυτή την ανάλυση, θα πρέπει να επισημάνουμε δύο ελλείψεις των μηχανισμών ταξινόμησης που αναφέραμε. Πρώτον, οι μηχανισμοί αυτοί δεν εξασφαλίζουν την υψηλή διαθεσιμότητα των εργασιών. Σε περίπτωση αστοχίας ενός μέρους του νέφους, η ανακατανομή των εργασιών πρέπει να βασιστεί στο υπάρχον πλαίσιο διαχείρισης του νέφους και όχι στις τεχνικές δρομολόγησης. Δεύτερον, οι μηχανισμοί αυτοί έχουν σχεδιαστεί να λειτουργούν μέσα σε ένα συγκεκριμένο περιβάλλον νέφους, στο οποίο γνωρίζουν ανά πάσα στιγμή την κατάσταση λειτουργίας του. Δεν έχουν επεκταθεί ώστε να ελέγχουν πολλαπλά υπολογιστικά νέφη, ώστε να εξυπηρετήσουν σενάρια cloud brokerage. Δεν μπορούν δηλαδή να δρομολογήσουν μια εργασία αν τα διαθέσιμα υπολογιστικά νέφη είναι περισσότερα του ενός.

5. Μελλοντικές κατευθύνσεις

Οι τεχνικές δρομολόγησης εργασιών που εξετάζονται στα πλαίσια της διπλωματικής αυτής εργασίας έχουν προταθεί μέσα στον τελευταίο ένα χρόνο (2013-2014) και βρίσκονται ακόμα σε ερευνητικό στάδιο. Θα είχε ιδιαίτερο ενδιαφέρον να μελετηθούν σε μεταπτυχιακό επίπεδο στο Ε.Μ.Π., και ενδεχομένως να αναπτυχθούν και να εφαρμοστούν στο περιβάλλον νέφους Oceanos του ΕΔΕΤ [58]. Ιδιαίτερα μια τεχνική στο μοντέλο του Paragon θα μπορούσε να συμπληρώσει το cloud stack Synnefo [59] και να οδηγήσει σε αποδοτικότερες και οικονομικότερες κατανομές των VMs στο περιβάλλον νέφους.

Μια έλλειψη που διαπιστώσαμε, είναι ότι οι τεχνικές δεν έχουν σχεδιαστεί (μέχρι στιγμής) να αντιμετωπίζουν περιστατικά μη-διαθεσιμότητας και να αναλαμβάνουν την αναδρομολόγηση μιας εργασίας, με επανάκτηση των δεδομένων και της κατάστασης λειτουργίας της. Αυτός ο ρόλος εξακολουθεί να ανήκει στο πλαίσιο διαχείρισης του νέφους, αν και θα λειτουργούσε αποδοτικότερα στο επίπεδο των τεχνικών δρομολόγησης. Θα είχε ενδιαφέρον η ανάπτυξη ενός τέτοιου μηχανισμού, που θα παρακολουθεί την κατάσταση λειτουργίας του νέφους μέσα από το API του πλαισίου λειτουργίας του, και θα αναλαμβάνει (αντί για το ίδιο το πλαίσιο) να αναδρομολογήσει τις εργασίες σε περίπτωση τεχνικής αστοχίας.

Μια ακόμη περιοχή για μελλοντική έρευνα, είναι η δυνατότητα χρήσης παρόμοιων τεχνικών για την επιλογή του πλέον κατάλληλου νέφους, από μια πληθώρα νεφών που είναι διαθέσιμα. Ένας τέτοιος μηχανισμός θα εξυπηρετούσε στη πράξη το μοντέλο του cloud broker.

Θα είχε τέλος ενδιαφέρον να αξιοποιηθούν πολλαπλοί schedulers και μηχανισμοί consolidation, είτε όμοιοι (για λόγους απόδοσης και υψηλής διαθεσιμότητας) είτε ετερογενείς. Η χρήση πολλαπλών ετερογενών schedulers ενδείκνυται σε περιπτώσεις όπως της τεχνικής Quasar που αποδίδει μόνο για συγκεκριμένο τύπο εργασιών. Θα μπορούσε δηλαδή για ένα σετ εργασιών να αξιοποιείται μηχανισμός βασισμένος στην τεχνική Quasar, και για άλλα σετ εργασιών να αξιοποιούνται διαφορετικοί, πιο εξειδικευμένοι μηχανισμοί δρομολόγησης.

Η χρήση πολλαπλών schedulers δημιουργεί μεγάλη ευελιξία, αλλά απαιτεί ένα συντονισμό. Ένας τέτοιος μηχανισμός συντονισμού θα μπορούσε να αναπτυχθεί σε περιβάλλον ZooKeeper [60].

Βιβλιογραφία και παραπομπές

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia. “Above the Clouds: A Berkley View on Cloud Computing”, Electrical Engineering and Computer Sciences University of California at Berkeley, February 10 2009

[2] L. A. Barroso, U. Holzle.“The Datacenter as a Computer". Synthesis Series on Computer Architecture, May 2009.

[3] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal. “Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities”, 2008

[4] Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/

[5] Windows Azure. http://www.windowsazure.com/

[6] Rackspace. http://www.rackspace.com/

[7] IBM SoftLayer. http://www.softlayer.com/

[8] Google Compute Engine. http://cloud.google.com/compute

[9] Chris Preimesberger. “Get off my Cloud: Private Cloud Computing Takes Shape”, 4 November 2008

[10] Michael Vizard. “The need to build private clouds”, 30 April 2009

[11] VMWare vSphere. http://www.vmware.com/products/vsphere/

[12] Citrix XenServer. http://www.citrix.com/products/

[13] Microsoft Hyper-V. http://www.microsoft.com/virtualization/

[14] HAMILTON, J. “Cost of Power in Large-Scale Data Centers”. November2008.

[15] ENVIRONMENTAL PROTECTION AGENCY. “U. S. Report to congress on server and data center energy efficiency public law 109-431”. Tech. rep., EPA ENERGY STAR Program, 2007.

[16] GANDHI, A., HARCHOL-BALTER, M., DAS, R., AND LEFURGY, C. “Optimal power allocation in server farms”. In Proceedings of ACM SIGMETRICS (2009).

[17] L. Barroso. "Warehouse-Scale Computing: Entering the Teenage Decade". ISCA Keynote, SJ, June 2011.

[18] J. Leverich, C. Kozyrakis. “On the Energy (In)Efficiency of Hadoop Clusters”. In Proc. of HotPower, October 2009.

[19] D. Meisner, C. Sadler, L. A. Barroso, et al. “Power Management of On-line Data-Intensive Services”. In Proc. of ISCA, SJ, June 2011.

[20] S. Govindan, et al. “Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines”. In Proc. of SOCC, 2011.

[21] J. Mars, L. Tang, et al. “Bubble-Up: Increasing Utilization in Modern Warehouse Scale Computers via Sensible Co-locations”. In Proc. of MICRO-44, Brazil, December 2011

[22] R. Nathuji, et al. “Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds”. In Proc. of EuroSys, 2010.

[23] Openstack cloud software. http://www.openstack.org/.

[24] KOH, Y., KNAUERHASE, R., BRETT, P., BOWMAN, M., WEN, Z., AND PU, C. “An analysis of performance interference effects in virtual environments”. In Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (2007).

[25] C. Kozyrakis, A. Kansal, et al. “Server Engineering Insights for Large-Scale Online Services”. In IEEE Micro, vol.30, no.4, July 2010.

[26] J. Mars, L. Tang, R. Hundt. “Heterogeneity in ”Homogeneous” Warehouse-Scale Computers: A Performance Opportunity”. In IEEE CAL, July-December 2011.

[27] R. Nathuji, C. Isci, E. Gorbatov. “Exploiting platform heterogeneity for power efficient data centers”. In Proc. of ICAC, 2007.

[28] Christina Delimitrou, Christos Kozyrakis "Paragon: QoS-Aware Scheduling for Heterogeneous Datacenters", Electrical Engineering Department, Stanford University, 2013

[29] K. Mills, J. Filliben , C. Dabrowski “Comparing VM-Placement Algorithms for On-Demand Clouds”. Information Technology Laboratory, 2011

[30] D. Nurmi, et al., “The Eucalyptus Open-Source Cloud-Computing System”, Proceedings of the 9 th IEEE/ACM International Symposium on Cluster Computing and the Grid, May 18-21, 2009, pp. 124-131.

[31] R. M. Bell. Y. Koren, C. Volinsky “The BellKor 2008 Solution to the Netflix Prize". Technical report, AT&T Labs, Oct 2007.

[32] A. Rajaraman and J. Ullman. “Textbook on Mining of Massive Datasets”, 2011.

[33] P.Mell, T. Grance “The NIST Definition of Cloud Computing”, National Institute of Standards and Technology, September 2011

[34] JIANG, Y., SHEN, X., CHEN, J., AND TRIPATHI, R. “Analysis and approximation of optimal co-scheduling on chip multiprocessors”. In Proceedings of the seventeenth International Conference on Parallel Architectures and Compilation Techniques (2008).

[35] JIANG, Y., TIAN, K., AND SHEN, X. “Combining locality analysis with online proactive job co-scheduling in chip multiprocessors”. In Proceedings of the International Conference on High-Performance Embedded Architectures and Compilers (2010), pp. 6–8.

[36] TIAN, K., JIANG, Y., AND SHEN, X. “A study on optimally co-scheduling jobs of different lengths on chip multiprocessors”. In Proceedings of the ACM International Conference on Computing Frontiers (2009).

[37] GREENBERG, A., HAMILTON, J., MALTZ, D. A., AND PATEL, P. “The cost of a cloud: research problems in data center networks”. In ACM SIGCOMM Comput. Commun. Rev. 39, 1 (January 2009), 68–73.

[38] KOHAVI, R., HENNE, R. M., AND SOMMERFIELD, D. “Practical guide to controlled experiments on the web: listen to your customers not to the hippo”. In Proceedings of the thirteenth annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007), pp. 959–967.

[39] SCHURMAN, E., AND BRUTLAG, J. “Performance related changes and their user impact”. In O’REILLY: Web Performance and Operations Conference (Velocity) (2009).

[40] Alan Roytman, Aman Kansal, Sriram Govindan, Jie Liu, Suman Nath. “PACMan: Performance Aware Virtual Machine Consolidation “,10th International Conference on Autonomic Computing , 2013.

[41] DEAN, J., AND GHEMAWAT, S. “MapReduce: A flexible data processing tool”. Communications of the ACM 53, 1 (2010), 72–77.

[42] VERMA, A., AHUJA, P., AND NEOGI, A. “pmapper: power and migration cost aware application placement in virtualized systems”. In Middleware (2008).

[43] Ben Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A.D. Joseph, R. Katz, S. Shenker, and I. Stoica. “Mesos: A platform for fine-grained resource sharing in the data center”. In Proc. of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI). Boston, MA, 2011.

[44] Ch. Delimitrou, Ch. Kozyrakis: “Quasar: Resource-Efficient and QoS-Aware Cluster Management”, Electrical Engineering Department, Stanford University, 2014

[45] Charles Reiss, Alexey Tumanov, Gregory Ganger, Randy Katz, and Michael Kozych. “Heterogeneity and dynamicity of clouds at scale: Google trace analysis”. In Proc. of the Third ACM Symposium on Cloud Computing (SOCC). San Jose, CA, 2012.

[46] McKinsey & Company. “Revolutionizing data center efficiency”. In Uptime Institute Symposium, 2008.

[47] Gartner. “Efficient data center design can lead to 300 percent capacity growth in 60 percent less space”. http://www.gartner.com/newsroom/id/1472714.

[48] Arunchandar Vasan, Anand Sivasubramaniam, Vikrant Shimpi, T. Sivabalan, and Rajesh Subbiah. “Worth their watts? An empirical study of datacenter servers”. In Proc. of the 16th International Symposium on High Performance Computer Architecture (HPCA). Bangalore, India, 2010.

[49] Host server cpu utilization in amazon ec2 cloud. http://huanliu.wordpress.com/2012/02/17/host-server-cpu-utilization-in-amazon-ec2-cloud/.

[50] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and JohnWilkes. “Omega: flexible, scalable schedulers for large compute clusters”. In Proc. of the 8th ACM European Conference on Computer Systems (EuroSys). Prague, Czech Republic, April 2013.

[51] Torque resource manager. http://www.adaptivecomputing.com/products/open-source/torque/.

[52] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. “Dominant resource fairness: fair allocation of multiple resource types”. In Proc. of the 8th USENIX conference on Networked systems design and implementation (NSDI). Boston, MA, 2011.

[53] Nedeljko Vasi´c, Dejan Novakovi´c, Svetozar Miuˇcin, Dejan Kosti´c, and Ricardo Bianchini. “Dejavu: accelerating resource allocation in virtualized environments”. In Proc. of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). London, UK, 2012.

[54] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. “Cloudscale: elastic resource scaling for multi-tenant cloud systems”. In Proc. of the 2nd ACM Symposium on Cloud Computing (SOCC). Cascais, Portugal, 2011.

[55] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. “Press: Predictive elastic resource scaling for cloud systems”. In Proc. Of the International Conference on Network and Service Management (CNSM). Niagara Falls, ON, 2010.

[56] Hiep Nguyen, Zhiming Shen, Xiaohui Gu, Sethuraman Subbiah, and John Wilkes. “Agile: Elastic distributed resource scaling for infrastructure-as-a-service”. In Proc. of the USENIX International Conference on Automated Computing (ICAC’13). San Jose, CA, 2013.

[57] Rightscale. https://aws.amazon.com/solution-providers/isv/rightscale.

[58] Oceanos IaaS Service. https://okeanos.grnet.gr/home/

[59] Synnefo open source Cloud Stack. https://www.synnefo.org/

[60] Apache ZooKeeper. http://zookeeper.apache.org/

Ορολογία και συντμήσεις

Στον παρακάτω πίνακα αναφέρονται συνήθεις όροι της Αγγλικής γλώσσας και συντμήσεις που έχουν χρησιμοποιηθεί σε αυτήν την εργασία και οι αντίστοιχοι όροι στην Ελληνική γλώσσα.

benchmark workloads	δοκιμαστικές εργασίες
classification	κατηγοριοποίηση, ταξινόμηση, χαρακτηρισμός (αναφέρεται σε εργασίες)
cloud computing	υπολογιστικό νέφος (ή «νέφος» για συντομία)
cluster	συστοιχία υπολογιστών
co-location	συστέγαση
consolidation	ενοποίηση πόρων
encryption	κρυπτογράφηση
heterogeneity	ετερογένεια (αναφέρεται στα ετερογενή υπολογιστικά συστήματα που συστήνουν ένα υπολογιστικό νέφος)
hybrid cloud	υβριδικό νέφος
interference	παρεμβολή (αρνητική επίπτωση σε μια εργασία από την παράλληλη λειτουργία άλλων εργασιών)
latency	χρόνος απόκρισης
private cloud	ιδιωτικό νέφος
QoS (Quality Of Service)	ποιότητα παρεχομένων υπηρεσιών (συνήθως ορίζει το αποδεκτό επίπεδο μιας παρεχόμενης υπηρεσίας)
replication	διαμοιρασμός
scaling	κλιμάκωση, επέκταση
scheduling	χρονοδρομολόγηση (αναφέρεται σε εργασίες)
SLA(Service Level Agreement)	συμφωνημένο ή εγγυημένο επίπεδο υπηρεσίας
sensitivity score (SSC)	σκορ ευαισθησίας (αναφέρεται σε παρεμβολές)
server configuration (SC)	διαμόρφωση εξυπηρετητή

source of interference (SoI)	πηγή της παρεμβολής
virtual machine	εικονική μηχανή
virtualization	εικονοποίηση (αναφέρεται σε συστημικές υποδομές)
workload	υπολογιστική εργασία (ή «εργασία» για συντομία)