Εθνικό Μετσόβιο Πολυτεχνείο

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ &
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαχείριση Κατανεμημένων Συστημάτων με Έμφαση στην Εκτέλεση Επιχειρησιακών Διεργασιών σε Νέφη Πραγματικού Χρόνου

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

ΣΠΥΡΙΔΩΝ Β. ΓΩΓΟΥΒΙΤΗΣ

Αθήνα, Ιούλιος 2013

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων & Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης &
Συστηματων Πληροφορικης

Διαχείριση Κατανεμημένων Συστημάτων με Έμφαση στην Εκτέλεση Επιχειρησιακών Διεργασιών σε Νέφη Πραγματικού Χρόνου

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

ΣΠΥΡΙΔΩΝ Β. ΓΩΓΟΥΒΙΤΗΣ

Συμβουλευτική Επιτροπή: Θεοδώρα Βαρβαρίγου
Ελευθέριος Καγιάφας
Βασίλειος Λούμος

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την

... Θ. Βαρβαρίγου Καθηγήτρια Ε.Μ.Π.	... Ε. Καγιάφας Καθηγητής Ε.Μ.Π.	... Β. Λούμος Καθηγητής Ε.Μ.Π.
... Π. Τσανάκας Καθηγητής Ε.Μ.Π.	... Α. Σταφυλοπάτης Καθηγητής Ε.Μ.Π.	... Σ. Παπαβασιλείου Αν. Καθηγητής Ε.Μ.Π.
	... Ε. Βαρβαρίγος Καθηγητής Παν. Πατρών	

Αθήνα, Ιούλιος 2013

. . .

ΣΠΥΡΙΔΩΝ Β. ΓΩΓΟΥΒΙΤΗΣ
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Πρόλογος

Η διδακτορική διατριβή που παρουσιάζεται στις επόμενες σελίδες εκπονήθηκε από τον Νοέμβριο του 2008 μέχρι τον Ιούλιο του 2013, στο εργαστήριο Τηλεπικοινωνιών του Τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής, στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Κατά την διάρκεια της εκπόνησης αυτής της διατριβής, είχα την ευκαιρία να ασχοληθώ με αρκετά ενδιαφέροντα επιστημονικά θέματα που αφορούν κυρίως στους τομείς της εκτέλεσης επιχειρησιακών διεργασιών και της διαχείρισης υπηρεσιοστρεφών υποδομών και υποδομών Νεφών και να αποκτήσω πολύτιμη εμπειρία και γνώσεις.

Κατ΄ αρχάς θα ήθελα να ευχαριστήσω θερμά την καθηγήτρια μου κ. Θεοδώρα Βαρβαρίγου για τις πολύτιμες συμβουλές της, τη στήριξη και την εμπιστοσύνη που μου έδειξε κατά την διάρκεια εκπόνησης της διατριβής μου. Επίσης θα ήθελα να ευχαριστήσω του καθηγητές της τριμελούς συμβουλευτικής επιτροπής κ.κ. Ελευθέριο Καγιάφα και Βασίλειο Λούμο.

Ευχαριστίες οφείλω επίσης σε όλους τους συναδέλφους με τους οποίους συνεργάστηκα άψογα όλα αυτά τα χρόνια. Ιδιαιτέρως θα ήθελα να

ευχαριστήσω τους Δημοσθένη Κυριαζή, Κλεοπάτρα Κωνσταντέλη, Αθανάσιο Βουλόδημο, Γεώργιο Κουσιουρή, Ανδρέα Μενύχτα, Βασίλη Αλεξάνδρου, Νικολέτα Μαυρογεωργή και Γρηγόρη Κατσαρό για την βοήθειά τους.

Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου Βασίλη και Μελίνα, για την απεριόριστη υποστήριξή τους και τις θυσίες που έκαναν για να μπορέσω να ακολουθήσω τα ενδιαφέροντα μου, τον αδερφό μου Ξενοφώντα, στον οποίο μπορώ να βασίζομαι πάντα και τη σύζυγό μου Ζωή για την υπομονή και την αγάπη της.

Σπυρίδων Β. Γωγουβίτης

Αθήνα, Ιούλιος 2013

Περίληψη

Η αυξανόμενη διαθεσιμότητα ευρυζωνικών συνδέσεων έχει επιτρέψει την εκτέλεση πολλών εργασιών μέσω διαδικτυακών συστημάτων συνεργασίας τα οποία βοηθούν στην επίτευξη υψηλού επιπέδου αλληλεπίδρασης. Έτσι το «Μελλοντικό Διαδίκτυο» αποτελείται από ένα ευρύ φάσμα διαδραστικών εφαρμογών όπως εργαλεία για την ταυτόχρονη σχεδίαση και απεικόνιση στον τομέα της μηχανολογίας, συστήματα για την συνεργασία κατά την δημιουργία βίντεο στον τομέα της ψυχαγωγίας καθώς και εικονικά περιβάλλοντα πολλαπλών χρηστών στον τομέα της εκπαίδευσης.

Πολλές από αυτές τις εφαρμογές τείνουν να χρησιμοποιούν εξειδικευμένο υλισμικό (hardware) προκειμένου να επιτευχθεί η επιθυμητή Ποιότητα Υπηρεσίας (Quality of Service - QoS), αυξάνοντας κατά πολύ το κόστος κτήσης και συντήρησής τους. Αυτό αποτελεί σημαντικό εμπόδιο στις μικρές επιχειρήσεις και νεοσύστατες εταιρείες που θέλουν να έχουν γρήγορη και εύκολη πρόσβαση στην αγορά.

Η ανάπτυξη του Υπολογιστικού Νέφους και των υπηρεσιοστρεφών υποδομών παροχής λογισμικού και πόρων ως υπηρεσίες αποτελούν μία λύση σε

αυτό το πρόβλημα με την δυνατότητα της αμοιβής ανά χρήση (pay-per-use) που παρέχουν, χωρίς την ανάγκη κτήσης ακριβού εξοπλισμού, ενώ ταυτόχρονα οι τελικοί χρήστες μπορούν να επωφεληθούν και από άλλα πλεονεκτήματα των Νεφών, όπως η ελαστικότητα (elasticity) και η αξιοπιστία. Ταυτόχρονα, οι εφαρμογές υιοθετούν μία υπηρεσιοστρεφή αρχιτεκτονική κατά την οποία πολλαπλές υπηρεσίες συνεργάζονται για να παρέχουν την απαιτούμενη λειτουργικότητα. Σε αυτό το πλαίσιο η ύπαρξη μηχανισμών διαχείρισης επιχειρησιακών διεργασιών είναι ιδιαιτέρως σημαντική.

Σκοπός της παρούσης διατριβής είναι η μελέτη των κατανεμημένων περιβαλλόντων Πλέγματος (Grid) και Νέφους (Cloud) και ιδιαιτέρως οι μηχανισμοί ελέγχου και εκτέλεσης ροών εργασίας (workflows) σε εικονικοποιημένα (virtualized) περιβάλλοντα. Ιδιαίτερη σημασία δίνεται στην ανάγκη παροχής εκ μέρους των συστημάτων αυτών εγγυήσεων Ποιότητας Υπηρεσίας ώστε να μπορούν να εκτελεστούν εφαρμογές ελαστικού πραγματικού χρόνου (soft real-time). Για τον λόγο αυτό, αναλύονται οι απαιτήσεις που έχουν οι διαδραστικές εφαρμογές, τα υπάρχοντα συστήματα διαχείρισης ροών εργασίας καθώς και οι δυνατότητες που παρέχουν τα σύγχρονα λειτουργικά συστήματα για εκτέλεση εφαρμογών πραγματικού χρόνου. Στη συνέχεια παρουσιάζεται μία αρχιτεκτονική λύση η οποία μπορεί να παρέχει τις απαιτούμενες υπηρεσίες και μηχανισμούς που επιτρέπουν την μοντελοποίηση και εκτέλεση διαδραστικών εφαρμογών σε κατανεμημένα περιβάλλοντα Νέφους. Η προτεινόμενη λύση επαληθεύεται και αξιολογείται μέσω σειράς πειραμάτων καθώς και μέσω πραγματικών εφαρμογών και παρουσιάζονται τα πλεονεκτήματα. Τέλος, παρουσιάζεται μία λύση για την επίβλεψη κατανεμημένων συστημάτων που εμφανίζει σημαντικές δυνατότητες

κλιμάκωσης και συνάθροισης των παρατηρούμενων μετρικών.

Λέξεις κλειδιά: Διαχείριση Επιχειρησιακών Διεργασιών, Κατανεμημένα Συστήματα, Εικονικοποίηση, Νέφος, Συστήματα Πραγματικού Χρόνου, Υπηρεσιοστρεφείς Υποδομές, Επίβλεψη Κατανεμημένων Συστημάτων.

Abstract

The growing availability of broadband Internet connections has enabled people to carry out tasks that were, up until now, parts of offline workflows through online collaborative systems, allowing for higher levels of interactivity. Therefore, emerging future Internet applications involve a broad class of interactive and collaborative tools and environments, including concurrent design and visualization in the engineering sector, media production in the creative industries, and multi-user virtual environments in education and gaming.

Many of these applications tend to use dedicated hardware in order to achieve the desired Quality of Service (QoS), greatly increasing the overall cost for maintaining the needed resources. This can be a major hindrance to small businesses and startup companies that want to make innovative solutions available easily.

Adopting a Cloud solution alleviates this problem by providing the option of pay-per-use without the need to own expensive equipment, while also making use of other advantages of Clouds, such as elasticity and reliability. At the same time, applications adopt a service oriented approach in which multiple services

work together to provide the required functionality. In this context, the existence of mechanisms for managing business processes are particularly important.

The aim of this thesis is the study of distributed Grid and Cloud environments and particularly the mechanisms needed to manage and execute business workflows in virtualized environments. Particular attention is given to the QoS guarantees that these systems need to provide in order for soft real-time applications to be deployed, managed and executed. For this reason, the requirements that interactive applications have are analyzed, existing workflow management systems are studied and the real-time offerings of modern operating systems are researched. Following, we present an architectural solution that can provide the necessary services and tools that enable modeling and execution of interactive applications in distributed Cloud environments. The proposed solution is verified and evaluated through a series of experiments as well as through real-life applications and its advantages are presented. Finally, a scalable monitoring solution able to perform aggregation is presented and evaluated.

Keywords: Business Process Management, Distributed Systems, Virtualization, Cloud, Real-time Systems, Service Oriented Infrastructures, Monitoring.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Συντμήσεων

ΒΔ	Βάση Δεδομένων
ΕΔΥ	Εικονικοποιημένο Δίκτυο Υπηρεσιών
ΕΜΠ	Εθνικό Μετσόβιο Πολυτεχνείο
ΚΜΕ	Κεντρική Μονάδα Επεξεργασίας
ΛΣ	Λειτουργικό Σύστημα
ΛΣ-ΓΧ	Λειτουργικό Σύστημα Γενικής Χρήσης
ΛΣ-ΠΧ	Λειτουργικό Σύστημα Πραγματικού Χρόνου
ΣΔΡΕ	Σύστημα Διαχείρισης Ροών Εργασίας
ΣΥΑ	Συμβόλαιο Επίπεδου Υπηρεσίας
ΤΝΔ	Τεχνητά Νευρωνικά Δίκτυα
API	Application Programming Interface
ASC	Application Service Component
BPEL	Business Process Execution Language
CBS	Constant Bandwidth Server
CDMI	Cloud Data Management Interface
CFS	Completely Fair Scheduler
DNS	Domain Name System
EDF	Earliest Deadline First
FIFO	First In First Out
FPS	Frames Per Second
GPF	Generalized Proportional Fair
GP-OS	General Purpose Operating System
GSI	Grid Security Infrastructure
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IT	Information Technology

NIST	National Institute of Science and Technology
NP-complete	Nondeterministic Polynomial time complete
NP-hard	Nondeterministic Polynomial time hard
NTP	Network Time Protocol
PaaS	Platform as a Service
POSIX	Portable Operating System Interface
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
RT-OS	Real-Time Operating System
SaaS	Software as a Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
POSIX	Portable Operating System Interface
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
VSN	Virtual Service Network
VSND	Virtual Service Network Description
WfMS	Workflow Management System
WSDL	Web Services Description Language
XPDL	XML Process Definition Language

1 Εισαγωγή

1.1 Σκοπός

Σκοπός της παρούσης εργασίας είναι η μελέτη των κατανεμημένων περιβαλλόντων και ιδιαιτέρως οι μηχανισμοί ελέγχου και εκτέλεσης ροών εργασίας (workflows) σε εικονικοποιημένα περιβάλλοντα Πλέγματος (Grid) και Νέφους (Cloud). Ιδιαίτερη σημασία δίνεται στην ανάγκη παροχής, εκ μέρους των συστημάτων αυτών, εγγυήσεων Ποιότητας Υπηρεσίας (Quality of Service - QoS) ώστε να μπορούν να εκτελεστούν εφαρμογές ελαστικού πραγματικού χρόνου.

Η αυξανόμενη διαθεσιμότητα ευρυζωνικών συνδέσεων έχει επιτρέψει την εκτέλεση πολλών εργασιών μέσω διαδικτυακών συστημάτων συνεργασίας τα οποία βοηθούν στην επίτευξη υψηλού επιπέδου αλληλεπίδρασης. Έτσι το «Μελλοντικό Διαδίκτυο» αποτελείται από ένα ευρύ φάσμα διαδραστικών εφαρμογών, όπως εργαλεία για την ταυτόχρονη σχεδίαση και απεικόνιση στον τομέα της εφαρμοσμένης μηχανικής, συστήματα για την συνεργασία κατά την δημιουργία βίντεο στον τομέα της ψυχαγωγίας καθώς και εικονικά

περιβάλλοντα πολλαπλών χρηστών στην εκπαίδευση.

Πολλές από αυτές τις εφαρμογές τείνουν να χρησιμοποιούν εξειδικευμένο υλισμικό (hardware) προκειμένου να επιτευχθεί η επιθυμητή Ποιότητα Υπηρεσίας, αυξάνοντας κατά πολύ το κόστος κτήσης και συντήρησής τους. Αυτό αποτελεί σημαντικό εμπόδιο στις μικρές επιχειρήσεις και νεοσύστατες εταιρείες που θέλουν να έχουν γρήγορη και εύκολη πρόσβαση στην αγορά. Η υιοθέτηση ενός Υπολογιστικού Νέφους αποτελεί μία λύση σε αυτό το πρόβλημα με την δυνατότητα της αμοιβής ανά χρήση (pay-per-use) που παρέχει, χωρίς την ανάγκη κτήσης ακριβού εξοπλισμού. Τα Υπολογιστικά Νέφη προσφέρουν πολλά άλλα πλεονεκτήματα όπως η ελαστικότητα και η αξιοπιστία.

1.2 Οργάνωση της Διατριβής

Η παρούσα διατριβή αποτελείται από 9 κεφάλαια. Στις ενότητες των κεφαλαίων αυτών παρουσιάζεται ουσιαστικά και με αναλυτικό τρόπο το αντικείμενο της διδακτορικής διατριβής. Το κεφάλαιο 2 παρέχει γενικές πληροφορίες για τα Υπολογιστικά Νέφη και τα Συστήματα Πραγματικού Χρόνου. Στο κεφάλαιο 3 αναλύονται οι απαιτήσεις που έχουν οι εφαρμογές ελαστικού πραγματικού χρόνου και το πώς αυτές επηρεάζουν τα συστήματα διαχείρισης ροών εργασίας. Στο κεφάλαιο 4 γίνεται μία επισκόπηση της τρέχουσας κατάστασης στον τομέα των συστημάτων διαχείρισης ροών εργασίας (ΣΔΡΕ - Workflow Management System - WfMS), ενώ στο κεφάλαιο 5 αναλύονται οι δυνατότητες πραγματικού χρόνου που παρέχουν τα Λειτουργικά Συστήματα Γενικής Χρήσης (ΛΣ-ΓΧ). Στο κεφάλαιο 6 παρουσιάζονται η

προτεινόμενη αρχιτεκτονική και οι σχεδιαστικές αποφάσεις που ελήφθησαν, ενώ στο κεφάλαιο 7 παρουσιάζεται η ποσοτική και ποιοτική αξιολόγηση του προτεινόμενου συστήματος διαχείρισης ροών εργασίας. Στο κεφάλαιο 8 παρουσιάζεται μία επέκταση του μηχανισμού παρακολούθησης με αυξημένες δυνατότητες. Τέλος στο κεφάλαιο 9 παρουσιάζονται τα συμπεράσματα και η μελλοντική ερευνητική εργασία που μπορεί να αποτελέσει συνέχεια της διατριβής. Ακολουθεί η σχετική βιβλιογραφία και οι δημοσιεύσεις που αποτελούν προϊόν της διδακτορικής διατριβής.

2

Ορισμός του Μοντέλου Υπηρεσιών Νέφους και των Συστημάτων Πραγματικού χρόνου

2.1 Το Μοντέλο Υπηρεσιών Νέφους

Με την εμφάνιση των Υπηρεσιοστρεφών Αρχιτεκτονικών (Service Oriented Architectures-SOA) (Erl, 2005) καθώς και τεχνολογιών όπως το Πλέγμα (Grid) (Foster & Kesselman, 1999) και το Υπολογιστικό Νέφος (Computational Cloud) (Hayes, 2008), αυξάνονται διαρκώς οι εφαρμογές που δεν αναπτύσσονται πλέον κατά έναν τρόπο που θα μπορούσε να χαρακτηριστεί ως μονολιθικός, αλλά ακολουθούν ένα μοντέλο κατά το οποίο μία εφαρμογή αποτελείται από πολλαπλές μεμονωμένες υπηρεσίες, οι οποίες συνεργάζονται υπό τον έλεγχο κάποιου Συστήματος Διαχείρισης Ροών Εργασίας (ΣΔΡΕ - Workflow Management System - WfMS).

Το μοντέλο αυτό ονομάζεται Μοντέλο Υπηρεσιών Νέφους (Cloud Service Model) (Mell & Grance, 2009), (Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009) και καλύπτει όλα τα επίπεδα της Τεχνολογίας Πληροφοριών (ΤΠ - Information Technology - IT). Για το υπολογιστικό Νέφος έχουν προταθεί

διάφοροι ορισμοί. Το Αμερικανικό Εθνικό Ίδρυμα Προτύπων και Τεχνολογίας (National Institute of Standards and Technology - NIST) δίνει τον ακόλουθο ορισμό για το Νέφος (Mell & Grance, 2011):

«Το Υπολογιστικό Νέφος είναι ένα μοντέλο για την πραγματοποίηση της συνεχούς, εύκολης και κατά παραγγελία πρόσβασης μέσω δικτύου σε μια κοινή ομάδα διαμορφώσιμων πόρων υπολογισμού (π.χ. δίκτυα, κεντρικοί υπολογιστές, αποθηκευτικοί χώροι, εφαρμογές και υπηρεσίες) που μπορούν να είναι διαθέσιμοι γρήγορα και με την ελάχιστη διαχειριστική προσπάθεια ή αλληλεπίδραση φορέων παροχής υπηρεσιών».

Ένας άλλος ορισμός, που προέρχεται από την έκθεση εμπειρογνωμόνων για το μέλλον του υπολογιστικού Νέφους, έχει ως εξής (Schubert, 2010):

« Ένα Νέφος είναι ένα ελαστικό περιβάλλον εκτέλεσης εφαρμογών που αποτελείται από πολλαπλούς πόρους, περιλαμβάνει διάφορους ενδιαφερόμενους χρήστες (stakeholders) και παρέχει μια μετρήσιμη υπηρεσία σε πολλαπλούς βαθμούς λεπτομέρειας (granularities) για ένα καθορισμένο επίπεδο ποιότητας (της υπηρεσίας).»

Το Μοντέλο Υπηρεσιών Νέφους αποτελείται κυρίως από 3 στρώματα, δηλαδή Υποδομή-ως-Υπηρεσία (Infrastructure-as-a-Service, IaaS), Πλατφόρμα-ως-Υπηρεσία (Platform-as-a-Service, PaaS) και Λογισμικό-ως-Υπηρεσία (Software-as-a-Service, SaaS) (Weinhardt, Anandasivam, Blau, & Stosser, 2009).

Υποδομή-ως-Υπηρεσία. Αυτός ο τύπος νεφών παρέχει τους εικονικοποιημένους πόρους στο χρήστη. Οι πόροι μπορούν να είναι επεξεργασίας, αποθήκευσης και δικτύωσης και τους οποίους ο καταναλωτής

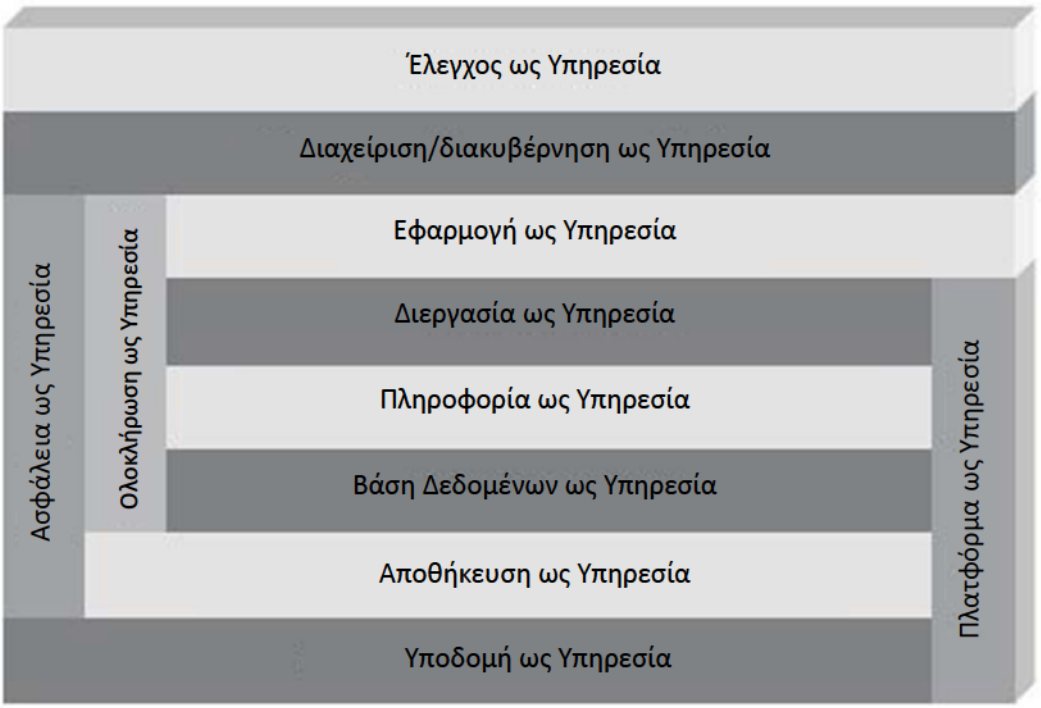
μπορεί να χρησιμοποιήσει για να εκτελέσει το λογισμικό. Ο καταναλωτής δεν έχει τον έλεγχο των φυσικών πόρων, αλλά έχει τον έλεγχο σε θέματα των λειτουργικών συστημάτων, της αποθήκευσης, των εφαρμογών και ενδεχομένως περιορισμένο έλεγχο τμημάτων του δικτύου (π.χ. τείχος προστασίας).

Πλατφόρμα-ως-Υπηρεσία. Οι πόροι παρέχονται μέσω μιας πλατφόρμας επάνω στην οποία οι υπηρεσίες και οι εφαρμογές που αναπτύσσονται ή κατέχονται από τον χρήστη είναι τοποθετημένες. Ο πάροχος εκθέτει μία συγκεκριμένη προγραμματιστική διεπαφή API την οποία ο χρήστης μπορεί να χρησιμοποιήσει για να αναπτύξει ή να επεκτείνει μια εφαρμογή. Ο καταναλωτής δεν έχει κανέναν έλεγχο της υποδομής επάνω στην οποία τρέχει η εφαρμογή. Παραδείγματα υλοποίησης PaaS περιλαμβάνουν τα Google App Engine και Windows Azure.

Λογισμικό-ως-Υπηρεσία. Ο χρήστης είναι σε θέση να χρησιμοποιήσει τις εφαρμογές του παρόχου που τρέχουν σε μια υποδομή ή μια πλατφόρμα Νέφους. Ο χρήστης δεν έχει κανέναν έλεγχο της υποδομής στην οποία τρέχει η εφαρμογή, αλλά χρησιμοποιεί τις λειτουργίες της, συνήθως μέσω ενός λεπτού πελάτη (thin client). Αυτός ο τύπος Νέφους περιγράφεται επίσης ως Νέφος υπηρεσιών ή εφαρμογής. Παραδείγματα υλοποίησης SaaS περιλαμβάνουν τα Salesforce Customer Relationships Management (CRM), SAP Business by Design και Google Docs.

Εκτός από αυτά τα βασικά στρώματα άλλες υποστηριζόμενες υπηρεσίες όπως φαίνονται στο Σχήμα 2.1 (Linthicum, 2009) είναι:

1. Αποθήκευση ως υπηρεσία
2. Βάση δεδομένων ως υπηρεσία



Σχήμα 2.1: *Στρώματα που αντιπροσωπεύουν τις υποστηριζόμενες υπηρεσίες του Μοντέλου Υπηρεσιών Νέφους*

3. Πληροφορία ως υπηρεσία
4. Διεργασία ως υπηρεσία
5. Ολοκλήρωση ως υπηρεσία
6. Ασφάλεια ως υπηρεσία
7. Διαχείριση/διακυβέρνηση ως υπηρεσία
8. Έλεγχος ως υπηρεσία

Οι τύποι ανάπτυξης (deployment) σύμφωνα με το NIST είναι:

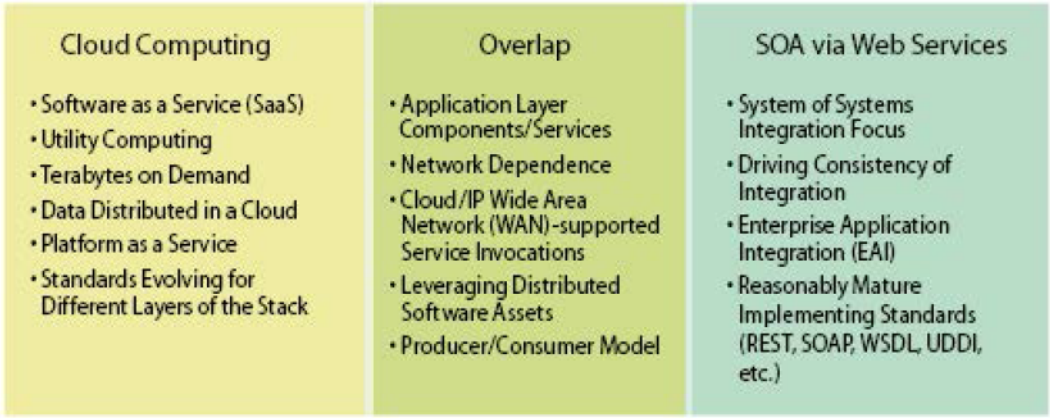
Ιδιωτικό Νέφος. Η υποδομή Νέφους χρησιμοποιείται από μια μόνο οργάνωση. Μπορεί να το διαχειρίζεται η οργάνωση ή ένας τρίτος και μπορεί είτε να συστεγάζεται με την οργάνωση (on premise) είτε όχι (off premise).

Κοινοτικό Νέφος. Η υποδομή Νέφους μοιράζεται από διάφορες οργανώσεις και υποστηρίζει μια συγκεκριμένη κοινότητα που έχει παρόμοιες απαιτήσεις (π.χ. απαιτήσεις ασφάλειας). Μπορεί τη διαχείρισή της να την έχει είτε κάποιο από τα μέλη της κοινότητας είτε κάποιος τρίτος και μπορεί να είναι είτε *on premise* είτε *off premise*.

Δημόσιο Νέφος. Η υποδομή Νέφους κατέχεται από μία οργάνωση η οποία παρέχει τις υπηρεσίες της στο ευρύ κοινό.

Υβριδικό Νέφος. Η υποδομή Νέφους είναι μια σύνθεση δύο ή περισσότερων νεφών (ιδιωτικό, κοινοτικό ή δημόσιο) που παραμένουν μοναδικές οντότητες αλλά δεσμεύονται μαζί από την τυποποιημένη ή ιδιόκτητη τεχνολογία που επιτρέπει τη φορητότητα δεδομένων και εφαρμογών (π.χ., Νέφος που εκρήγνυται για την εξισορρόπηση του φόρτου μεταξύ των νεφών).

Όσον αφορά τη συγγένεια του Νεφών και των Υπηρεσιοστραφών Αρχιτεκτονικών (SOA) υπάρχουν σημαντικές επικαλύψεις και εκτιμήσεις, όπως φαίνεται στο Σχήμα 2.2 (Raines, 2009). Και οι δύο έννοιες περιέχουν στοιχεία προσανατολισμού προς τις υπηρεσίες. Οι υπηρεσίες πολλών τύπων είναι διαθέσιμες σε ένα κοινό δίκτυο προς χρήση από τους καταναλωτές. Το μοντέλο του Νέφους εστιάζει στη μετατροπή των επιπέδων της Τεχνολογίας Πληροφοριών σε προϊόντα που μπορούν να αγοραστούν επαυξητικά από τους βασισμένους στο Νέφος παρόχους και μπορούν να θεωρηθούν σε πολλές περιπτώσεις ως ένας τύπος εξωτερικής ανάθεσης (outsourcing).

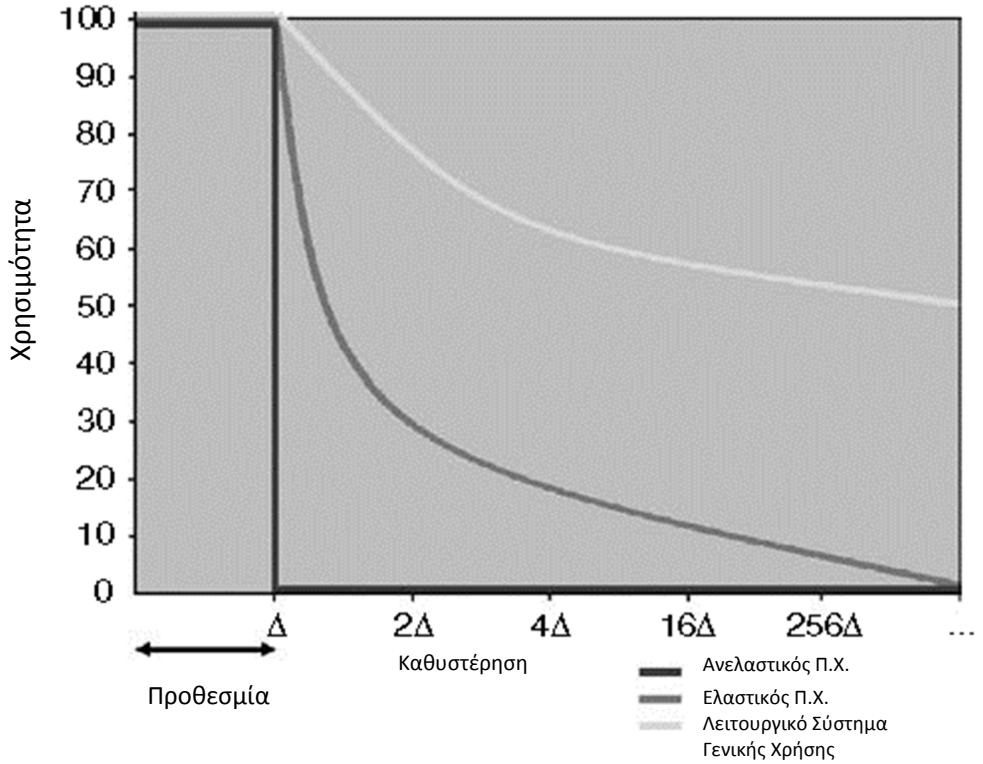


Σχήμα 2.2: *Νέφος και Υπηρεσιοστρεφείς Αρχιτεκτονικές*

2.2 Συστήματα Πραγματικού Χρόνου

Οι απαιτήσεις που έχουν πολλές εφαρμογές ως προς την Ποιότητα Υπηρεσία τις κατατάσσουν ως συστήματα ελαστικού πραγματικού χρόνου. Τα Συστήματα Πραγματικού Χρόνου (Real-Time Systems) είναι συστήματα λογισμικού/υλισμικού, η ορθότητα της λειτουργίας των οποίων εξαρτάται από το αν πληρούνται ή όχι συγκεκριμένοι χρονικοί περιορισμοί. Σε «πραγματικό χρόνο» δεν σημαίνει «πολύ γρήγορα», αλλά ότι οι εκτελέσεις πρέπει να γίνονται πριν την λήξη των προθεσμιών τους, διαφορετικά έχουν αποτύχει (Buttazzo, Lipari, Abeni, & Caccamo, 2005), (Sha κ. συν., 2004).

Παραδοσιακά, τα συστήματα πραγματικού χρόνου διαιρούνται σε δύο κατηγορίες, ανελαστικού και ελαστικού πραγματικού χρόνου. Τα συστήματα ανελαστικού πραγματικού χρόνου (hard real-time systems) είναι συστήματα στα οποία η μη τήρηση μιας χρονικής προθεσμίας μπορεί να προκαλέσει γενική αποτυχία και δυσλειτουργία του συστήματος. Παραδείγματα αποτελούν στρατιωτικά συστήματα ελέγχου, συστήματα πλοήγησης αεροσκαφών,



Σχήμα 2.3: *Χρησιμότητα μίας εργασίας μετά τον περιορισμό*

συστήματα ελέγχου της παραγωγικής διαδικασίας εργοστασίων παράγωγης πυρηνικής ενέργειας και γενικά συστήματα εντοπισμού κρίσιμων συνθηκών τα οποία συνήθως αναπτύσσονται ως ενσωματωμένα συστήματα.

Τα συστήματα ελαστικού πραγματικού χρόνου οφείλουν να λειτουργούν εντός συγκεκριμένων χρονικών περιορισμών. Μικρές αποκλίσεις δεν είναι καταστροφικές, αλλά το σύστημα λειτουργεί με χαμηλότερης ποιότητας προσφερόμενη υπηρεσία. Τα συστήματα αυτά εμφανίζονται πολύ συχνά σε ηλεκτρονικούς υπολογιστές γενικής χρήσης και περιλαμβάνουν εφαρμογές όπως η αναπαραγωγή εικόνας και ήχου, η οπτικοποίηση επιστημονικών δεδομένων, τα παιχνίδια και γενικά διαδραστικές εφαρμογές. Στο Σχήμα 2.3 φαίνονται οι διαφορές ανάμεσα στα συστήματα ελαστικού και ανελαστικού

πραγματικού χρόνου (Bishop, 2007).

Πρέπει εδώ να τονιστεί ότι θα μπορούσε κανείς να θεωρήσει όλα τα συστήματα ως συστήματα ανελαστικού πραγματικού χρόνου. Κάτι τέτοιο δεν θα ήταν επιθυμητό καθώς τα συστήματα ανελαστικού πραγματικού χρόνου σχεδιάζονται και υλοποιούνται βάσει των χειρότερων δυνατών συνθηκών, κάτι που οδηγεί σε υπερδέσμευση (overprovisioning) πόρων. Επίσης, πρέπει να τονιστεί ότι ο όρος πραγματικός χρόνος δεν έχει να κάνει με την ταχύτητα ενός συστήματος, αλλά με την προβλεψιμότητα (predictability) και την καθοριστικότητα (determinism) που αυτά παρουσιάζουν.

Ο όρος «πραγματικού χρόνου» αναφέρεται στο ότι ο χρόνος του συστήματος (δηλαδή, η αναπαράσταση του χρόνου στο εσωτερικό του συστήματος), θα πρέπει πάντα να συγχρονίζεται με τον αντίστοιχο εξωτερικό χρόνο, με βάση τον οποίον μετρώνται όλα τα χρονικά διαστήματα στο περιβάλλον.

2.3 Συστήματα Πραγματικού Χρόνου και Υπολογιστικό Νέφος

Όπως έχει ήδη αναφερθεί, πολλές από τις εφαρμογές που περιγράφηκαν μπορούν να επωφεληθούν από τις υποδομές υπολογιστικού Νέφους. Παράλληλα όμως δημιουργούνται κάποιες νέες προκλήσεις που θα πρέπει να αντιμετωπιστούν. Αυτό συμβαίνει διότι τα υπολογιστικά Νέφη βασίζονται σε τεχνικές εικονικοποίησης και μερισμού των υπολογιστικών πόρων γεγονός που πολλές φόρες έχει οδηγήσει σε ασταθείς καταστάσεις (Durkee, 2010),

(Williamson, 2010). Έτσι, τόσο η υποδομή όσο και το Σύστημα Διαχείρισης Ροών Εργασίας (ΣΔΡΕ) θα πρέπει να μπορούν να λαμβάνουν υπόψη τους τις απαιτήσεις των εφαρμογών ως προς το παρεχόμενο επίπεδο υπηρεσίας.

3 Απαιτήσεις

Για να μπορέσουμε να αξιολογήσουμε καλύτερα τις απαιτήσεις που έχουν οι τελικοί χρήστες από ένα Σύστημα Διαχείρισης Ροών Εργασίας, εκπονήθηκε μία μελέτη αγοράς και τεχνικών απαιτήσεων με τη συμμετοχή 42 εταιρειών (*Market and Technical Requirements Analysis, IRMOS Project*, 2010). Η ανάλυση των απαντήσεων των ερωτηθέντων οδήγησε στα εξής αποτελέσματα:

- 60% τον χρηστών απάντησε ότι χρησιμοποιεί ή σκοπεύει να χρησιμοποιήσει στο άμεσο μέλλον τεχνολογίες SOA για να αποκτήσει μεγαλύτερη ευελιξία.
- Οι ισχυρότεροι ανασταλτικοί παράγοντες για την υιοθέτηση τεχνολογιών Υπολογιστικού Νέφους ήταν θέματα Ασφάλειας (21%), Απόδοσης (15%) και Υποστήριξης (15%).
- Το 70% αναγνώρισε την δυνατότητα αλληλεπίδρασης με τον χρήστη σε πραγματικό χρόνο ως σημαντικό στοιχείο ενώ το 15% απάντησε ότι δεν υπάρχει τέτοια ανάγκη και το 15% δεν έδωσε απάντηση (με αναγωγή στα συμπληρωμένα ερωτηματολόγια).

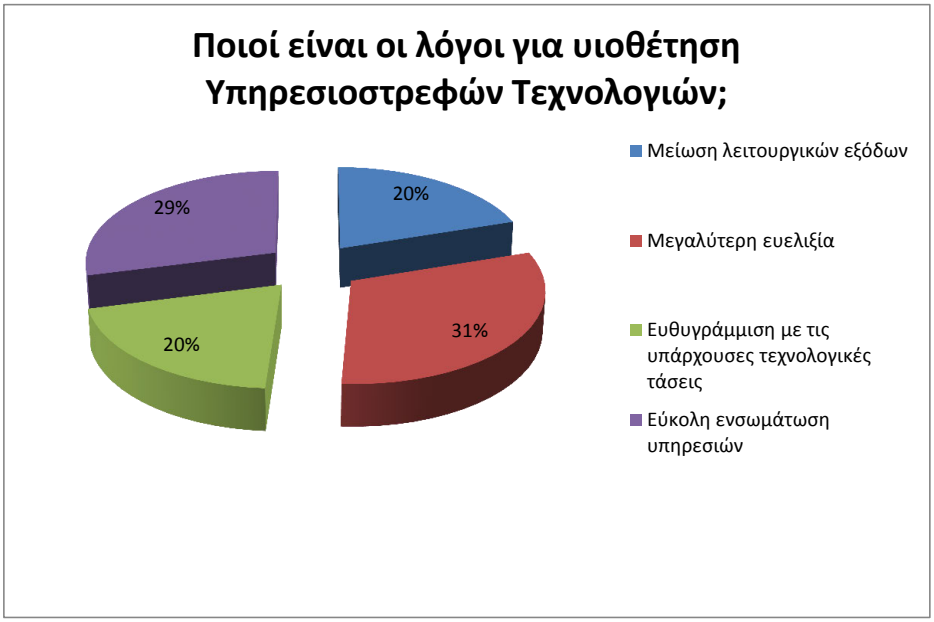
Η έρευνα αυτή, μαζί με στοιχεία από την βιβλιογραφία (Lin κ. συν., 2009), (Korkhov κ. συν., 2007), οδηγεί σε μία σειρά από απαιτήσεις που πρέπει να καλύπτει ένα ΣΔΡΕ για να είναι επιτυχημένο.

- *Επεκτασιμότητα/Κλιμακωσιμότητα.* Το ΣΔΡΕ θα πρέπει να μπορεί να δέχεται πολλαπλές ταυτόχρονες αιτήσεις εξυπηρέτησης χωρίς να επηρεάζεται η απόδοσή του ούτε η δυνατότητα απόκρισης εντός σαφώς προδιαγεγραμμένων χρονικών ορίων. Παράλληλα το σύστημα πρέπει να μπορεί να χρησιμοποιεί αποδοτικά νέους πόρους που προστίθενται στο σύστημα και να παρουσιάζει αύξηση της απόδοσής του. Έτσι είναι δυνατό να αντεπεξέλθει στην αυξανόμενη ζήτηση από τις εφαρμογές.

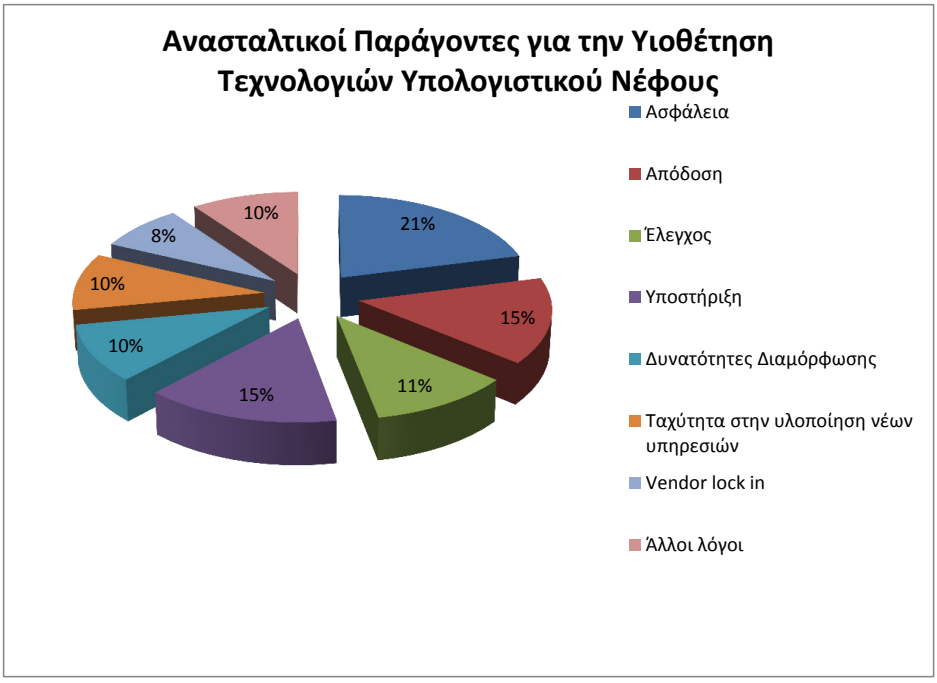
- *Διαδραστικότητα.* Ένα ΣΔΡΕ θα πρέπει να παρέχει τη δυνατότητα στον τελικό χρήστη να αλληλεπιδρά με μία ροή εργασίας που εκτελείται. Αυτό σημαίνει ότι ο χρήστης θα πρέπει να μπορεί να αλλάζει τον έλεγχο ροής μίας εφαρμογής καθώς και τις απαιτήσεις επιπέδου υπηρεσίας αυτής. Αυτό θα πρέπει να γίνεται πάντα στα πλαίσια ενός Συμβολαίου Επίπεδου Υπηρεσίας (ΣΕΥ - Service Level Agreement - SLA).

- *Διαχείριση σφαλμάτων.* Σφάλματα είναι πιθανό να συμβούν τόσο σε επίπεδο λογισμικού όσο και σε επίπεδο υλισμικού. Για το λόγο αυτό, το ΣΔΡΕ θα πρέπει να μπορεί όχι μόνο να συλλάβει ότι κάτι τέτοιο έχει συμβεί, αλλά και να παρέχει την δυνατότητα ορισμού ενεργειών που θα πρέπει να εκτελεστούν ώστε, αν είναι δυνατόν, να μην παραβιαστεί το επιθυμητό επίπεδο υπηρεσίας της εφαρμογής.

- *Δυνατότητα ορισμού επιπέδου υπηρεσίας σε όρους φιλικούς προς τον*



Σχήμα 3.1: *Αίτια υιοθέτησης Υπηρεσιοστρεφών Τεχνολογιών*



Σχήμα 3.2: *Ανασταλτικοί παράγοντες για την υιοθέτηση τεχνολογιών Υπολογιστικού Νέφους*

Σχήμα 3.3: *Ανάγκη για υποστήριξη αλληλεπίδρασης σε πραγματικό χρόνο*

χρήστη. Υπάρχουν πολλές περιπτώσεις όπου είναι επιθυμητό ο χρήστης μίας εφαρμογής να μπορεί να ορίσει το επίπεδο υπηρεσίας με όρους που καταλαβαίνει, όπως παραδείγματος χάριν, «αριθμός χαμένων καρέ ενός βίντεο» και όχι με τεχνικούς όρους όπως «ταχύτητα του κεντρικού επεξεργαστή».

- *Παρακολούθηση Ροών Εργασίας*. Ένα ΣΔΡΕ θα πρέπει να μπορεί να παρακολουθεί την τρέχουσα κατάσταση μίας ροής εργασίας και να μεταφέρει την πληροφορία αυτή στο χρήστη.

- *Ασφάλεια*. Ένα ΣΔΡΕ θα πρέπει να παρέχει και να υλοποιεί τους κατάλληλους μηχανισμούς ασφάλειας ώστε να μπορεί να χρησιμοποιηθεί σε εταιρικά περιβάλλοντα.

- *Υποστήριξη κληροδοτημένου (legacy) κώδικα.* Παρότι τα υπό εξέταση ΣΔΡΕ αναφέρονται σε Υπηρεσιοστρεφείς Αρχιτεκτονικές, θα πρέπει να μπορούν να εκτελούν διεργασίες που βασίζονται σε κληροδοτημένο, μη υπηρεσιοστρεφή κώδικα. Αυτό βοηθάει ώστε να μπορούν να χρησιμοποιηθούν άμεσα υπάρχουσες εφαρμογές που είναι δαπανηρό να αναπτυχθούν από την αρχή ως υπηρεσίες. Έτσι θα πρέπει να παρέχονται κατάλληλα εργαλεία τα οποία θα «περιτυλίξουν» (wrap) τον κληροδοτημένο κώδικα ώστε να επιτρέψουν τη χρήση του σε ένα υπηρεσιοστρεφές περιβάλλον.

Οι απαιτήσεις αυτές είναι καθοριστικές και συνέβαλλαν τα μέγιστα στην διαμόρφωση του προτεινόμενου ΣΔΡΕ.

4 Συστήματα Διαχείρισης Ροών Εργασίας

4.1 Βιβλιογραφική Ανασκόπηση των ΣΔΡΕ

Ο τομέας των ΣΔΡΕ είναι εκτενής και έχουν προταθεί τόσο κατανεμημένες λύσεις (Ranno, Shrivastava, & Wheater, 1997), (Papazoglou, Dells, Bouguettaya, & Haghjoo, 1997), (Joeris & Herzog, 1999) όσο και λύσεις που ασχολούνται με τις εγγυήσεις επίπεδου υπηρεσίας (Zeng κ. συν., 2004), (Kyriazis, Tserpes, Menychtas, Litke, & Varvarigou, 2008). Στις επόμενες παραγράφους αναφέρονται προτάσεις κυρίως από τον τομέα των Πλεγμάτων (Grids) καθώς εμφανίζουν τις περισσότερες ομοιότητες σε σχέση με τα Υπολογιστικά Νέφη.

Το Taverna (Oinn κ. συν., 2004) είναι ένα ΣΔΡΕ που μπορεί να διαχειρισθεί SOAP/WSDL και REST υπηρεσίες ιστού (web services) και ακολουθεί μία ιεραρχική αρχιτεκτονική. Το Taverna δεν παρέχει εγγυήσεις επιπέδου υπηρεσίας. Παρ' όλα αυτά, το σύστημα επιτρέπει τον διαρκή έλεγχο των ροών που εκτελούνται, ενώ σε περίπτωση σφάλματος μπορεί να προσπαθήσει να εκτελέσει είτε την ίδια διεργασία είτε μία παραπλήσιά της.

Το πρόγραμμα Askalon (Fahringer κ. συν., 2005) εστιάζει σε εφαρμογές που είναι απαιτητικές αναφορικά με την απόδοση. Παρότι η αρχιτεκτονική της πλατφόρμας είναι γενικά αποκεντρωμένη, οι αποφάσεις λαμβάνονται από έναν κεντρικό μηχανισμό. Οι χρήστες μπορούν να ορίσουν χρονικούς περιορισμούς και το σύστημα μπορεί να λαμβάνει αποφάσεις χρονοπρογραμματισμού βασιζόμενο σε προβλέψεις για την απόδοση του συστήματος και τον χρόνο που απαιτείται για την εκτέλεση κάθε εργασίας. Το Askalon παρέχει την δυνατότητα εποπτείας της ροής εργασίας που εκτελείται, χωρίς όμως να είναι δυνατή η αλλαγή αυτής. Σε περίπτωση σφάλματος το σύστημα είναι δυνατόν να μετακινήσει μία διεργασία σε άλλο εξυπηρετητή. Παράλληλα παρέχει την δυνατότητα να ορίζονται σημεία κατά τα οποία σώζεται η κατάσταση της ροής εργασίας καθώς και τα ενδιάμεσα αποτελέσματα. Έτσι σε περίπτωση σφάλματος η ροή εργασίας μπορεί να συνεχίσει από το τελευταίο σημείο κατά το οποίο σώθηκε η κατάσταση του και όχι από την αρχή.

Το περιβάλλον Amadeus (Brandic, Pllana, & Benkner, 2008) ακολουθεί μία κεντρικοποιημένη αρχιτεκτονική και παρέχει τη δυνατότητα ορισμού περιορισμών σχετικά με τον χρόνο εκτέλεσης των εφαρμογών και το κόστος των εκτελέσεων. Το σύστημα υπολογίζει τους βέλτιστους φυσικούς πόρους βασιζόμενο σε προβλέψεις για την απόδοση τους. Ο χρήστης του συστήματος καλύπτεται από SLAs, αλλά δεν παρέχεται κανένας έλεγχος πάνω στην ροή εργασίας που εκτελείται, ούτε χρησιμοποιούνται μηχανισμοί ανοχής σε σφάλματα.

Το ερευνητικό πρόγραμμα GrADS (Berman κ. συν., 2001) είναι βασισμένο στην πλατφόρμα Globus Toolkit (GT) (Globus, α.χ.) και στοχεύει σε εφαρμογές με μεγάλο υπολογιστικό και επικοινωνιακό φορτίο. Υποστηρίζει την

προδιαγραφή ροών εργασίας που στη συνέχεια αναλύονται και οι εξαρτήσεις μεταξύ των επιμέρους εργασιών προσδιορίζονται. Αυτό βοηθά ώστε, όπου αυτό είναι εφικτό, οι διεργασίες να παραλληλοποιούνται. Το GrADS υποστηρίζει επίσης περιορισμούς επιπέδου υπηρεσίας μέσω του υπολογισμού του χρόνου εκτέλεσης μίας εφαρμογής με χρήση ιστορικών στοιχείων και αναλυτικής μοντελοποίησης.

Το ΣΔΡΕ Kepler (Ludascher κ. συν., 2006) είναι μία εφαρμογή ανοικτού κώδικα που επεκτείνει τις δυνατότητες του συστήματοςPtolemy II (Eker κ. συν., 2003). Το κύριο χαρακτηριστικό του είναι ότι βασίζεται σε υπολογιστικά «βήματα», τα οποία αποκαλούνται «δράστες» (actors), και έχουν σαφώς προσδιορισμένα στοιχεία εισόδου και εξόδου. Οι χρήστες μπορούν να ορίσουν μία ροή εργασίας μέσω ενός γραφικού περιβάλλοντος, στο οποίο επιλέγουν διαφορετικούς δράστες και τους συνδέουν μεταξύ τους. Ένας «διευθυντής» (director) είναι υπεύθυνος να επιβλέπει την εκτέλεση της ροής εργασίας ως σύνολο. Ο κύριος μηχανισμός ανοχής στα σφάλματα που υλοποιεί το σύστημα βασίζεται στον ορισμό «δραστών» οι οποίοι είναι υπεύθυνοι να αναγνωρίζουν λανθασμένες καταστάσεις και να δρουν αναλόγως.

Το πρόγραμμα GRIDCC (A. S. McGough κ. συν., 2007) έχει ως στόχο να παρέχει την δυνατότητα χρησιμοποίησης ερευνητικών οργάνων μέσω υποδομών πλέγματος. Το ΣΔΡΕ που χρησιμοποιείται από το πρόγραμμα GRIDCC είναι υπεύθυνο αφενός για την βελτιστοποίηση των ροών εργασίας και αφετέρου να εξασφαλίζει την επίτευξη των απαιτήσεων ποιότητας υπηρεσίας που ορίζονται. Για να το επιτύχει αυτό ορίζει μία καινούρια γλώσσα η οποία μπορεί να χρησιμοποιηθεί σε συνδυασμό με την BPEL η οποία αποτελεί μία από τις πιο διαδεδομένες γλώσσες για τον ορισμό ροών εργασίας. Έτσι, για

τον ορισμό μία ροής εργασίας στο GRIDCC χρησιμοποιείται ένα κείμενο BPEL για να περιγράψει τη ροή καθώς και ένα δεύτερο κείμενο το οποίο περιέχει αναφορές στο πρώτο και ορίζει απαιτήσεις ποιότητας υπηρεσίας για όποια διεργασία είναι απαραίτητο.

Το VLAM-G (Korkhov κ. συν., 2007) είναι το ΣΔΡΕ του προγράμματος VL-e. Είναι ένα αποκεντρωμένο ΣΔΡΕ που βασίζεται στον έλεγχο της ροής των δεδομένων ανάμεσα στις διεργασίες και σκοπό έχει να εξυπηρετήσει την κοινότητα της «ηλεκτρονικής επιστήμης» (e-science), την κοινότητα, δηλαδή, που χρησιμοποιεί εντατικά υπολογιστικές δομές για την διενέργεια επιστημονικών πειραμάτων. Το ΣΔΡΕ έχει την δυνατότητα να διευθύνει τόσο διεργασίες που έχουν αναπτυχθεί συγκεκριμένα για το VLAM-G, όσο και γενικές υπηρεσίες διαδικτύου ή legacy εφαρμογές. Οι διεργασίες συνδυάζονται μέσω των εξαρτήσεων που έχουν, βάσει των δεδομένων πάνω στα οποία τρέχουν. Το σύστημα επιτρέπει τον έλεγχο μίας ροής εργασίας που βρίσκεται σε εξέλιξη, όσο και την αλληλεπίδραση μαζί της.

Μία υπηρεσιοστρεφής πρόταση για ανάλυση πολυμεσικών εφαρμογών είναι αυτή των (Heinzl κ. συν., 2009), όπου προτείνεται η χρήση του Flexible SOAP πρωτοκόλλου για την μοντελοποίηση ροών δεδομένων με χρήση της γλώσσας BPEL. Έτσι ελαχιστοποιούνται οι μεταφορές πακέτων μεταξύ υπηρεσιών που ανταλλάσσουν μεγάλες ποσότητες δεδομένων. Οι συγγραφείς υποστηρίζουν ότι επιτυγχάνουν επεκτασιμότητα για το σύστημα με το να χρησιμοποιούν μία μηχανή BPEL που είναι ικανή να χρησιμοποιήσει τόσο πόρους πλέγματος όσο και πόρους από το Amazon EC2. Η πρόταση αυτή, όμως, δεν προσφέρει καμία εγγύηση ποιότητας υπηρεσίας.

Στη δημοσίευση των (Kirschnick, Calero, Wilcock, & Edwards, 2010),

οι συγγραφείς προτείνουν μία αρχιτεκτονική για την αυτοματοποιημένη παροχή υπηρεσιών σε περιβάλλοντα Υπολογιστικού Νέφους. Ο χρήστης μπορεί να επιλέξει ήδη υπάρχουσες υπηρεσίες και να τις τροποποιήσει ώστε να καλύπτουν τις ανάγκες του. Ταυτόχρονα η πλατφόρμα επιτρέπει τον επαναπροσδιορισμό των υπηρεσιών κατά τον χρόνο εκτέλεσης βάσει πληροφοριών που λαμβάνονται από το ίδιο το σύστημα για την κατάσταση στην οποία βρίσκεται. Η πρόταση, όπως παραδέχονται και οι ίδιοι οι συγγραφείς, δεν παρέχει μηχανισμούς ανοχής σε σφάλματα, ούτε εγγυήσεις ποιότητας υπηρεσίας ή SLA.

Στο (Rodero-Merino κ. συν., 2010) προτείνεται το Claudia που είναι ένα επίπεδο αφαίρεσης (abstraction layer) πάνω από το επίπεδο υποδομής το οποίο επιτρέπει τον έλεγχο του κύκλου ζωής των υπηρεσιών. Κάθε υπηρεσία στο Claudia καθορίζεται από ένα αρχείο περιγραφής υπηρεσιών που περιλαμβάνει πληροφορίες προσαρμογής, όπως τις IP διευθύνσεις άλλων υπηρεσιών, οι οποίες δεν είναι γνωστές πριν την εκτέλεση. Επιπλέον, μπορούν να περιγραφούν καταστάσεις όπου το σύστημα θα επεκτείνεται αυτόματα, όπως παραδείγματος χάριν μέσω της ενεργοποίησης νέων στιγμιοτύπων (instances). Αυτό καθορίζεται από τον χρήστη μέσω σαφών κανόνων επέκτασης.

Στο (Y. Zhang, Koelbel, & Cooper, 2009), οι συντάκτες ερευνούν έναν μηχανισμό χρονικού προγραμματισμού για εφαρμογές ροής εργασίας σε υπολογιστικά πλέγματα πολλαπλών συστάδων (multi-cluster Grids). Ο αλγόριθμος χρησιμοποιεί πληροφορίες πραγματικού χρόνου για να επανεκτιμήσει τις προβλέψεις που έγιναν για την αντιστοίχιση διεργασιών της ροής στους διαθέσιμους πόρους.

Μια ενδιαφέρουσα πρόταση προέρχεται από την περιοχή των

βιομηχανικών κατασκευών (Y. Zhang, Huang, Qu, & Ho, 2010), όπου οι συντάκτες προτείνουν τη χρήση ενός ΣΔΡΕ που βασίζεται σε πράκτορες (agents) για τη βιομηχανική αυτοματοποίηση. Ο εξοπλισμός καθώς και έξυπνα αντικείμενα «περιβάλλονται» (wrapped) με πράκτορες (agents) και εκτίθενται ως υπηρεσίες Ιστού που περιέχουν πληροφορίες θέσης σε πραγματικό χρόνο. Οι πράκτορες μπορούν στη συνέχεια να χρησιμοποιηθούν για να διαμορφώσουν μια ροή εργασίας που περιγράφει μια διαδικασία κατασκευής. Η αρχιτεκτονική επιτρέπει σε πραγματικό χρόνο την παρακολούθηση και τον έλεγχο της διαδικασίας.

Τεχνικές ροής εργασίας έχουν χρησιμοποιηθεί επίσης σε ενσωματωμένα συστήματα όπως συμβαίνει με το (Chou κ. συν., 2009), όπου ένα ΣΔΡΕ προτείνεται για βοηθητικές (assistive) συσκευές. Οι συντάκτες προτείνουν το πρωτόκολλο SISARL-XPDL που είναι μια επέκταση του XPDL. Η πρόταση στοχεύει να χρησιμοποιηθεί στα ενσωματωμένα συστήματα για την εκτέλεση διαδικασιών όπως η αποφυγή αντικειμένων από ρομποτικούς μηχανισμούς ή για ευφυή συστήματα αποθήκευσης φαρμάκων. Παρότι η πρόταση είναι πολύ ενδιαφέρουσα δεν μπορεί να συγκριθεί άμεσα με τις προηγούμενες εργασίες, καθώς έχει συγκεκριμένο πεδίο εφαρμογής και δεν επεκτείνεται στον τομέα το υπηρεσιών διαδικτύου.

Οι συγγραφείς του (Weber, Partsch, Scheller-Huoy, Schweitzer, & Schneider, 1997) προτείνουν ένα πλαίσιο που μπορεί να χρησιμοποιηθεί για να μοντελοποιηθούν συσκέψεις οι οποίες εν συνεχεία θα ενσωματωθούν σε ροές εργασίας. Ένας «μεσίτης συσκέψεως» ενεργεί ως μεσολαβητής μεταξύ ενός ΣΔΡΕ και ενός συστήματος για την απομακρυσμένη συνεργασία (telecooperation). Αυτό επιτρέπει σε γεωγραφικά διασκορπισμένες ομάδες

να συνεργαστούν από μακριά και να πραγματοποιήσουν συνεδριάσεις, οι εκβάσεις των οποίων μπορούν να τροφοδοτήσουν μία επιχειρησιακή διαδικασία.

Μία σύγκριση των διαφορετικών προτάσεων σε σχέση με τις απαιτήσεις που τέθηκαν στο κεφάλαιο 3 φαίνεται στον Πίνακα 4.1.

	Επεκτασιμότητα	Ασφάλεια	Εικονικοποίηση	QoS	Επισκόπηση (Monitoring)	Διαδραστικότητα	Έλεγχος Λαθών	Legacy κώδικας
Taverna	Κεντρικοποιημένο	ΝΑΙ (WS-Security)	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ (retry and alternate location)	Ν/Α
Askalon	Αποκεντρικοποιημένο	ΝΑΙ(GSI)	ΟΧΙ	ΝΑΙ (time constraints)	ΝΑΙ	ΟΧΙ	ΝΑΙ	Ν/Α
Amadeus	Κεντρικοποιημένο	ΝΑΙ (WS-Security)	ΟΧΙ	ΝΑΙ (χρόνος και κόστος)	ΟΧΙ	ΟΧΙ	ΟΧΙ	Ν/Α
GrADS	Κεντρικοποιημένο	ΝΑΙ (WS-Security)	ΟΧΙ	ΝΑΙ (χρόνος και κόστος)	ΝΑΙ	ΟΧΙ	ΝΑΙ (επιπέδου task)	Ν/Α
Kepler	Κεντρικοποιημένο	ΝΑΙ (Apache Rampart)	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	Ν/Α
GRIDCC	Κεντρικοποιημένο	ΝΑΙ (GSI)	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ
VLAM-G	Αποκεντρικοποιημένο	ΝΑΙ (Globus Security)	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	Ν/Α	ΝΑΙ
Heinzl	Αποκεντρικοποιημένο	ΝΑΙ (GSI)	ΝΑΙ (EC2)	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ
Kirschnik	Κεντρικοποιημένο	ΝΑΙ (Εικονικοποίηση)	ΝΑΙ	ΟΧΙ	ΝΑΙ	Μερικώς (reconfiguration)	ΟΧΙ	ΟΧΙ
Claudia	Κεντρικοποιημένο	ΝΑΙ (Εικονικοποίηση)	ΝΑΙ	ΝΑΙ	ΝΑΙ	Μερικώς (reconfiguration)	ΟΧΙ	ΟΧΙ
Zhang	Κεντρικοποιημένο	Ν/Α	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	Ν/Α	Ν/Α
EMWF	Ν/Α	Ν/Α	Ν/Α	Ν/Α	Ν/Α	ΝΑΙ (restart)	Ν/Α	Ν/Α

Πίνακας 4.1: *Σύγκριση διαφόρων ΣΔΡΕ*

4.2 Γλώσσες Περιγραφής Ροών Εργασίας

Οι ευρύτερα χρησιμοποιημένες προδιαγραφές για την περιγραφή διαδικαστικών ροών εργασίας από επιχειρήσεις είναι η XPDL (*XML Process Definition Language (XPDL)*, 2008) και η ebXML (OASIS, 2006) ενώ για υπηρεσιοστραφείς αρχιτεκτονικές είναι η BPEL (*Web Services Business Process Execution Language Version 2.0*, 2007). Εκτός από τη συνηθισμένη υποστήριξη για ενορχήστρωση (orchestration), η BPEL παρέχει μηχανισμούς για τον ορισμό επιχειρησιακών ρόλων και την μοντελοποίηση της πραγματικής συμπεριφοράς των συμμετεχόντων σε μια επιχειρησιακή αλληλεπίδραση. Παράλληλα το W3C προτείνει την WS-CDL (*Web Services Choreography Description Language, Version 1.0, W3C candidate recommendation*, 2005), (Kavantzas κ. συν., 2005) για την περιγραφή της συνεργασίας μεταξύ ισότιμων μελών. Η BPEL, καθώς και οι περισσότερες γλώσσες περιγραφής ροών εργασίας για επιχειρήσεις, εστιάζουν στη ροή ελέγχου.

Εντούτοις, εκτενής έρευνα για τα πρότυπα ελέγχου ροής εργασίας έχει δείξει ότι όλες οι γλώσσες έχουν περιορισμούς από την άποψη του τι μπορεί να εκφραστεί εύκολα (Aalst, Hofstede, Kiepuszewski, & Barros, 2003). Η ανεπάρκεια στην εκφραστικότητα και η έλλειψη αυστηρής σημασιολογίας δεν επιτρέπει αυτοματοποιημένους ελέγχους για την ακρίβεια και την πληρότητα μίας προδιαγραφής ροής εργασίας. Έτσι η BPEL και τα παρόμοια πρότυπα ροής εργασίας υπηρεσιών Ιστού έχουν περιορισμούς όταν εφαρμόζονται σε ένα κατανεμημένο περιβάλλον. Η εργασία για τα πρότυπα ροών εργασίας οδήγησε τον Van der Aalst (Van der Aalst, 2011) να παρέχει μια ενημερωμένη και εκτενή σύγκριση των γλωσσών και των εφαρμογών ροής εργασίας (τόσο

ανοικτού κώδικα όσο και εμπορικών εφαρμογών). Ο Van der Aalst έχει προσδιορίσει συγκεκριμένα πρότυπα που ομαδοποιούνται στο χειρισμό ελέγχου ροής, ελέγχου δεδομένων, ελέγχου των πόρων και των εξαιρέσεων.

Παρόλα αυτά, πρέπει να σημειωθεί ότι τα πρότυπα ροών εργασίας είναι μια ταχέως εξελισσόμενη ερευνητική περιοχή και οι αξιολογήσεις από την ομάδα του Van der Aalst μπορούν, σε μερικές περιπτώσεις, να μην λαμβάνουν υπ'οψη τους τις πιο πρόσφατες διαθέσιμες προδιαγραφές. Παραδείγματος χάριν, η WS-BPEL 2.0 έγινε διαθέσιμη το καλοκαίρι του 2007 και επιτρέπει τον ορισμό διαδικασιών μαζί με χειριστές αποζημιώσεων και εξαιρέσεων. Η έκδοση 2.0 υποστηρίζεται ήδη ευρέως.

Ενώ η αξιολόγηση των υπαρχουσών γλωσσών ροών εργασίας σε σχέση με τα πρότυπα ροών εργασίας με την καλά καθορισμένη σημασιολογία τους επιτρέπει τη χρήσιμη σύγκριση των προτύπων ή των εφαρμογών, δεν εξετάζει το πρόβλημα της έμφυτης έλλειψης αυστηρής σημασιολογίας, ερμηνεύσιμης από μηχανές μέσα σε κάθε γλώσσα ροής εργασίας.

Η κοινότητα σημασιολογικών υπηρεσιών Ιστού, παράγει αυστηρά πρότυπα για τη σημασιολογική περιγραφή των υπηρεσιών Ιστού «από κάτω προς τα επάνω» (bottom-up) . Διάφορα ευρωπαϊκά προγράμματα όπως τα (SEKT, α.χ.), (DIP, α.χ.), (SUPER, α.χ.) και (ASG, α.χ.) εργάστηκαν μαζί μέσω της Ευρωπαϊκής Πρωτοβουλίας Σημασιολογικών Συστημάτων (ESSI) και ανέπτυξαν την WSMO (*Web Service Modeling Ontology WSMO*, 2005) για την περιγραφή οντολογιών καθώς και την WSML (*Web Service Modeling Language WSML*, 2008) για την περιγραφή υπηρεσιών, η οποία έχει υποβληθεί για τυποποίηση στο W3C. Εν τω μεταξύ η εργασία που γίνεται από τον ακαδημαϊκό κόσμο και τη βιομηχανία μέσω του SWSI έχει οδηγήσει στο

Πλαίσιο Σημασιολογικών Υπηρεσιών Ιστού SWSF, το οποίο έχει και μια γλώσσα την SWSL, που έχει υποβληθεί επίσης στο W3C για τυποποίηση, και μια οντολογία, την SWSO που περιλαμβάνει ένα μοντέλο διαδικασιών. Η πολυπλοκότητα και η έλλειψη υποστηρικτικών εργαλείων αποτελεί εμπόδιο στη υιοθέτηση παρόμοιων λύσεων και είναι ένας παράγοντας που πρέπει να εξεταστεί κατά την εφαρμογή τους σε κατανεμημένα περιβάλλοντα.

Από την άλλη πλευρά, ερευνητικά προγράμματα στον τομέα της μηχανικής και των επιστημών έχουν αναπτύξει δικές τους λύσεις για την διαχείριση ροών εργασίας. Οι εφαρμογές που απαντώνται σε αυτούς τους τομείς είναι συχνά εντατικές σχετικά με τα δεδομένα και του υπολογισμούς και χρησιμοποιούν υποδομές πλέγματος. Ένα χαρακτηριστικό παράδειγμα είναι η ενορχήστρωση υπολογιστικών πόρων για την εκτέλεση μαθηματικών προσομοιώσεων μεγάλης κλίμακας, όπως παραδείγματος χάριν η μοντελοποίηση μεγάλων βιολογικών συστημάτων (Lloyd κ. συν., 2007). Μερικές λύσεις στρέφονται στην παροχή υποστήριξης για επιστημονικές εφαρμογές, π.χ. το Taverna (Oinn κ. συν., 2006), Pegasus (Deelman κ. συν., 2004) και Kepler (Bowers, Ludascher, Ngu, & Critchlow, 2006), ενώ άλλες αντιμετωπίζουν περισσότερο αρχιτεκτονικά ζητήματα όπως τα KWF-Grid (Neubauer, Hoheisel, & Geiler, 2006), Triana (Taylor, Shields, Wang, & Harrison, 2007) και Askalon (Fahringer κ. συν., 2007), ή οι μηχανισμοί ροής εργασίας στο Unicore (Sild, Maran, Romberg, Schuller, & Benfenati, 2005) και το Globus (Globus, α.χ.). Η μέθοδος που υιοθετείται από αυτές τις λύσεις είναι συχνά επί παραγγελία, συμπεριλαμβανομένης της προσαρμογής του ελέγχου ροής εργασίας σε συγκεκριμένες περιοχές όπως π.χ. επεξεργασία δεδομένων βιοπληροφορικής, υποστήριξη συγκεκριμένων απαιτήσεων που δεν

καλύπτονται από τα υπάρχοντα πρότυπα, όπως μεταφορά μεγάλου όγκου δεδομένων μεταξύ υπηρεσιών ή την αλληλεπίδραση του χρήστη κατά την εκτέλεσης πειραμάτων (Coles κ. συν., 2006).

Ενώ πολλά από αυτά τα προαναφερθέντα ερευνητικά προγράμματα έχουν αναπτύξει δικές τους γλώσσες και μηχανισμούς εκτέλεσης ροών εργασίας αντί να υιοθετήσουν υπάρχοντα πρότυπα, εντούτοις πολλά από αυτά έχουν βασιστεί σε υπάρχοντες φορμαλισμούς και επομένως είναι υποψήφια για επέκταση ή και επαναχρησιμοποίηση. Παραδείγματος χάριν, το Triana είναι βασισμένο σε δίκτυα Petri, το Akogrimo χρησιμοποιεί BPEL και η γλώσσα OWL-WS (Beco, Cantalupo, Giammarino, Matskanis, & Surridge, 2005) που αναπτύχθηκε στα πλαίσια του NextGrid είναι μια επέκταση του OWL-S (Burstein κ. συν., 2004). Εντούτοις, αν και κάποια πρότυπα χρησιμοποιούνται, είναι αναγνωρισμένο πρόβλημα το ότι προηγμένες αρχιτεκτονικές πλέγματος, όπως το NextGrid και Akogrimo, δεν έχουν επιτύχει μία κοινή προσέγγιση (Beco, Cantalupo, & Terracina, 2006).

4.3 Χρονοπρογραμματισμός Ροών Εργασίας

Ο χρονοπρογραμματισμός ροών εργασίας μπορεί να θεωρηθεί ως μία μορφή καθολικού χρονοπρογραμματισμού διεργασιών καθώς εστιάζει στην λήψη αποφάσεων για το που θα πρέπει να εκτελεσθεί κάθε διεργασία καθώς και στο να διαχειριστεί τις αλληλεξαρτήσεις μεταξύ των διεργασιών λαμβάνοντας υπ'οψη τους κοινούς πόρους που οφείλουν να διαμοιράζονται οι διεργασίες. Οι πόροι είναι δυνατόν να είναι ετερογενείς τόσο σε επίπεδο τοπικής διαμόρφωσης όσο και σε επίπεδο πολιτικών που εφαρμόζονται. Ο

χρονοπρογραμματισμός οφείλει να λάβει υπ'οψη τις απαιτήσεις ποιότητας υπηρεσίας των χρηστών κατά την λήψη των αποφάσεων.

Ο χρονοπρογραμματισμός ροών εργασίας αποτελείται από τις εξής κατηγορίες:

- *Αρχιτεκτονική χρονοπρογραμματισμού.* Οι αρχιτεκτονικές χρονοπρογραμματισμού ροών εργασίας μπορούν να διαχωρισθούν σε τρεις κύριες κατηγορίες: κεντρικοποιημένες, ιεραρχικές και αποκεντρικοποιημένες. Όταν μία μηχανή εκτέλεσης ροών εργασίας ακολουθεί μία κεντρικοποιημένη αρχιτεκτονική, όλες οι αποφάσεις λαμβάνονται από έναν κεντρικό χρονοπρογραμματιστή. Σε αυτές τις περιπτώσεις οι αποφάσεις είναι βέλτιστες αλλά υπάρχουν σοβαρά προβλήματα στην επεκτασιμότητα του συστήματος. Σε αντιδιαστολή, τόσο οι ιεραρχικές όσο και οι αποκεντρικοποιημένες αρχιτεκτονικές μοιράζουν την ευθύνη των αποφάσεων σε πολλαπλούς χρονοπρογραμματιστές. Και στις δύο περιπτώσεις το σύστημα εμφανίζει πολύ καλή επεκτασιμότητα αλλά πρέπει να δίνεται μεγάλη προσοχή ώστε να μην υπάρχει υποβάθμιση της απόδοσης του συστήματος.

- *Λήψη αποφάσεων.* Η λήψη αποφάσεων αναφέρεται κυρίως στις αποφάσεις που πρέπει να ληφθούν για την αντιστοίχιση των διαφορετικών διεργασιών μίας ροής εργασίας με πόρους και υπηρεσίες. Αυτό μπορεί να επιτευχθεί είτε σε τοπικό επίπεδο, όπου η απόφαση για μία διεργασία λαμβάνει υπόψη μόνο δεδομένα για την συγκεκριμένη διεργασία, είτε σε συνολικό επίπεδο, όπου λαμβάνονται υπόψη δεδομένα για όλες τις διεργασίες.

- *Σχεδιασμός προγραμματισμού.* Οι αφηρημένες ροές εργασίας οφείλουν να μετατραπούν σε μία μορφή που αναφέρεται σε συγκεκριμένους πόρους και υπηρεσίες. Ο προγραμματισμός μπορεί να είναι είτε στατικός (όπου η δυναμικά μεταβαλλόμενη κατάσταση των πόρων δεν λαμβάνεται υπόψη) είτε δυναμικός (όπου, σε αντίθεση με τον στατικό, τόσο στατικές όσο και δυναμικές πληροφορίες για τους πόρους χρησιμοποιούνται για τον σχεδιασμό και τη λήψη των αποφάσεων κατά το χρόνο εκτέλεσης).

- *Στρατηγική χρονοπρογραμματισμού.* Ο χρονοπρογραμματισμός ροών εργασίας σε κατανεμημένα συστήματα είναι ένα NP-complete πρόβλημα (Fernandez-Baca, 1989). Για αυτό το λόγο έχουν προταθεί διάφοροι ευριστικοί (heuristic) αλγόριθμοι που οδηγούν σε πολύ καλές (near-optimal) λύσεις για να επιτευχθούν οι απαιτήσεις του χρήστη. Οι περισσότερες τέτοιες λύσεις είναι επικεντρωμένες στην απόδοση καθώς προσπαθούν να μειώσουν τον συνολικό χρόνο εκτέλεσης. Υπάρχουν και λύσεις που λαμβάνουν υπ'οψη τους άλλες παραμέτρους, όπως το συνολικό κόστος (market-driven) ή την ασφάλεια και την εμπιστοσύνη (trust-driven).

- *Εκτίμηση απόδοσης.* Η εκτίμηση της απόδοσης είναι ένας σημαντικός παράγοντας κατά τον χρονοπρογραμματισμό και την εκτέλεση μίας ροής εργασίας. Οι χρονοπρογραμματιστές χρησιμοποιούν τεχνικές εκτίμησης της απόδοσης για να προβλέψουν την απόδοση των διαφορετικών διεργασιών μίας ροής εργασίας σε ένα κατανεμημένο περιβάλλον προκειμένου να ληφθούν αποφάσεις σχετικά με το πού (σε ποιον πόρο) θα λάβει χώρα η εκτέλεση μίας διεργασίας. Τεχνικές εκτίμησης της

απόδοσης αποτελούν η προσομοίωση, η αναλυτική μοντελοποίηση, τα ιστορικά δεδομένα, η μηχανική μάθηση και άλλες.

4.4 Ανοχή σε Σφάλματα

Μία δυναμική και ετερογενής υπηρεσιοστρεφής υποδομή μπορεί να είναι γεωγραφικά και διοικητικά διασκορπισμένη και να αποτελείται από διαφορετικούς πόρους όπως υπολογιστικά συστήματα, αποθηκευτικά μέσα, επιστημονικά όργανα, συστήματα επικοινωνιών καθώς και ανθρώπινους πόρους. Σε ένα τέτοιο ετερογενές περιβάλλον οι αλλαγές είναι πολυάριθμες, ιδιαίτερα μεταβλητές και με απρόβλεπτα αποτελέσματα. Αυτές οι αλλαγές μπορούν να οδηγήσουν σε σφάλματα για διάφορους λόγους: μη διαθεσιμότητα των απαραίτητων υπηρεσιών ή τμημάτων λογισμικού, υπερφορτωμένοι πόροι, έλλειψη μνήμης και αποτυχία της δικτυακής υποδομής. Για αυτούς τους λόγους, η διαχείριση ροών εργασίας σε υπηρεσιοστρεφής υποδομές πρέπει να είναι σε θέση να ανιχνεύσει και να διαχειριστεί τα σφάλματα προκειμένου να εξασφαλίσει αξιόπιστη υποστήριξη στο περιβάλλον εκτέλεσης.

Οι τεχνικές διαχείρισης σφαλμάτων ροών εργασίας μπορούν να διαιρεθούν σε δύο επίπεδα: επιπέδου εργασίας (task) και επιπέδου ροής (Soonwook & Kesselman, 2003). Ενώ οι τεχνικές επιπέδου εργασίας προσπαθούν να μετριάσουν την επίπτωση των σφαλμάτων μίας υπηρεσίας σε ολόκληρη τη ροή εργασίας, οι τεχνικές επιπέδου ροής προσπαθούν να αλλάξουν την δομή της ροής εργασίας για να αντεπεξέλθουν σε αυτά.

Οι τεχνικές επιπέδου *εργασίας* μπορούν να διαιρεθούν στις εξής κατηγορίες:

- *Επαναπροσπάθεια.* Μετά από μία αποτυχία επιχειρείται να εκτελεστεί πάλι η ίδια εργασία στον ίδιο πόρο.

- *Εναλλακτικός πόρος.* Μετά από μία αποτυχία επιχειρείται να εκτελεστεί η εργασία σε διαφορετικό πόρο.

- *Σημεία ελέγχου.* Οι κατάσταση μία εργασίας διατηρείται συνεχώς σε άλλον πόρο. Έτσι σε περίπτωση σφάλματος η εργασία μπορεί να εκτελεστεί σε άλλον πόρο από το σημείο του τελευταίου σημείου ελέγχου.

- *Αντιγραφή.* Η εργασία εκτελείται ταυτόχρονα σε πολλούς πόρους.

Οι τεχνικές επιπέδου *ροής* μπορούν να διαιρεθούν στις εξής κατηγορίες:

- *Εναλλακτική εργασία.* Μετά από μία αποτυχία επιχειρείται να εκτελεστεί μία διαφορετική εργασία που παρέχει την ίδια λειτουργικότητα.

- *Πολλαπλές εργασίες.* Πολλαπλές εναλλακτικές εργασίες που παρέχουν την ίδια λειτουργικότητα εκτελούνται ταυτόχρονα.

- *Καθορισμένος από τον χρήστη χειρισμός εξαιρέσεων.* Οι χρήστες είναι υπεύθυνοι να ορίσουν διορθωτικές ενέργειες για συγκεκριμένους τύπους λαθών.

- *Διάσωση της ροής.* Αυτή η τεχνική αγνοεί τα σφάλματα και προσπαθεί, αν είναι δυνατόν, να εκτελέσει τις υπόλοιπες εργασίες της ροής. Στη συνέχεια στατιστικά στοιχεία συλλέγονται και από το σύστημα για επεξεργασία.

- *Σημεία ελέγχου.* Οι κατάσταση όλων των εργασιών διατηρείται συνεχώς σε κάποιο κεντρικό πόρο. Έτσι, σε περίπτωση σφάλματος η εκτέλεση της ροής μπορεί να συνεχιστεί.

4.5 Τεχνικές Δέσμευσης Πόρων για Μελλοντική Χρήση (Advance Reservation)

Η έλλειψη εγγυήσεων ποιότητας υπηρεσίας, ειδικά για εφαρμογές που βασίζονται σε ροές εργασίας, θεωρείται ως ένας από τους πιο κρίσιμους αποτρεπτικούς παράγοντες για την υιοθέτηση των Υπηρεσιοστρεφών Αρχιτεκτονικών από την βιομηχανία, καθώς αποθαρρύνει τόσο τους παρόχους υπηρεσιών όσο και τους πιθανούς πελάτες από το να κάνουν εκτεταμένη χρήση των λειτουργιών που παρέχει μία Υπηρεσιοστρεφή Υποδομή. Η δέσμευση πόρων για μελλοντική χρήση θεωρείται ένας μηχανισμός που επιλύει αυτό το πρόβλημα και παρέχει στους χρήστες τις απαιτούμενες εγγυήσεις ποιότητας υπηρεσίας (Schwiegelshohn, Yahyapour, & Wieder, 2006). Κατ' ουσίαν, η δέσμευση πόρων επιτρέπει σε έναν χρήστη να ζητήσει πόρους για συγκεκριμένο χρονικό διάστημα δηλώνοντας τον χρόνο έναρξης και λήξης της κράτησης και να λάβει έναν ικανοποιητικό αριθμό πόρων για την υποστήριξη της εκτέλεσης της εφαρμογής κατά τη διάρκεια αυτού του διαστήματος.

4.5.1 Ντετερμινιστική Δέσμευση Πόρων

Η ντετερμινιστική δέσμευση πόρων αποτελείται από μαθηματικές μεθόδους για την δέσμευση των πόρων. Βασίζεται σε ντετερμινιστικούς

αλγορίθμους για την ανάλυση των περιορισμών απόδοσης με την εφαρμογή μαθηματικών διαδικασιών στα διάφορα στρώματα και τις υποθέσεις που σχετίζονται με κάθε υποψήφιο πόρο. Οι τεχνικές αυτές μειώνουν τον κίνδυνο να χαθούν οι προθεσμίες εκτέλεσης και αυξάνουν την αξιοπιστία του συστήματος ενώ μπορούν να εφαρμοστούν αποτελεσματικά και σε συστήματα πραγματικού χρόνου. Οι μελέτες που έχουν διεξαχθεί για την απόδοση αυτών των μηχανισμών (Netto, Bubendorfer, & Buyya, 2007) έχουν καταδείξει ικανοποιητικά αποτελέσματα. Εντούτοις, τέτοιες μέθοδοι έχουν και περιορισμούς. Οι υπάρχοντες ντετερμινιστικοί αλγόριθμοι χρονοπρογραμματισμού υποθέτουν ότι οι παράμετροι για την ευελιξία είναι στατικές (Farooq, Majumdar, & Parsons, 2005), ενώ επικεντρώνονται στην ελαχιστοποίηση του χρόνου απόκρισης του συστήματος και δεν προωθούν την κοινή εκμετάλλευση των πόρων από πολλαπλές υπηρεσίες. Έτσι οδηγούν στον τεμαχισμό (fragmentation) και την υπο-χρησιμοποίηση των πόρων.

4.5.2 Πιθανοτική Δέσμευση Πόρων

Το βασικό κίνητρο για τη πιθανοτική δέσμευση πόρων είναι το γεγονός ότι σε πολλές περιπτώσεις οι παράμετροι εισαγωγής και οι συνθήκες κατά το χρόνο εκτέλεσης είναι τόσες πολλές, που καθίσταται ανεπαρκές και μερικές φορές αδύνατο να εξεταστεί αναλυτικά ολόκληρος ο χώρος λύσεων του προβλήματος. Γενικά, οι πιθανοτικές μέθοδοι υιοθετούν τις αρχές της θεωρίας αποφάσεων και επηρεάζονται από προγενέστερες πιθανότητες που προέρχονται από τον συνδυασμό αναλυτικών μοντέλων και μετρήσεων αξιολόγησης. Αυτές οι προγενέστερες πιθανότητες χρησιμοποιούνται για να

καθορίσουν τη μεταγενέστερη πιθανότητα μιας απόφασης δέσμευσης πόρων. Αυτοί οι αλγόριθμοι επιδεικνύουν καλύτερα αποτελέσματα αναφορικά με την χρησιμοποίηση των πόρων και οδηγούν σε χαμηλότερο κόστος, αλλά εμφανίζονται λιγότερο αξιόπιστοι από τους ντετερμινιστικούς αλγορίθμους αναφορικά με την απόδοση που προσφέρεται στις εφαρμογές.

4.5.3 Υλοποιήσεις Μηχανισμών Δέσμευσης Πόρων

Το Askalon είναι ένα περιβάλλον ανάπτυξης εφαρμογών Πλέγματος με κύριο στόχο την βελτιστοποίηση των εφαρμογών. Το περιβάλλον εκτιμά την απόδοση της εφαρμογής βασιζόμενο σε ιστορικά στοιχεία που λαμβάνονται κατά την διάρκεια μίας φάσης εκμάθησης καθώς επίσης και μέσω αναλυτικής μοντελοποίησης. Επιπροσθέτως, κατά την φάση εύρεσης των υπηρεσιών που μπορούν να εκτελέσουν τη ροή εργασίας χρησιμοποιεί μία σειρά από παραμέτρους ποιότητας υπηρεσίας. Οι παράμετροι αυτοί μπορούν να είναι είτε προκαθορισμένες από το σύστημα είτε να δοθούν από τον χρήστη. Ο Wieczorek (Wieczorek, Siddiqui, Villazon, Prodan, & Fahringer, 2006) προτείνει μία επέκταση του μηχανισμού χρονοπρογραμματισμού του Askalon ώστε να υποστηρίζεται δέσμευση πόρων. Η προσέγγιση αυτή εκτελεί μία χαρτογράφηση των πόρων που βασίζεται στην αρχή της δίκαιης διανομής των πόρων η οποία περιορίζει τον αριθμό των πόρων που προσφέρονται σε κάποιο χρήστη μέσα σε μία χρονική περίοδο. Ο χρονοπρογραμματισμός βασίζεται στον ευριστικό αλγόριθμο HEFT (Topcuoglu, Hariri, & Min-You, 2002) ο οποίος έχει αποδειχθεί ότι λειτουργεί ικανοποιητικά με χαμηλή χρονική πολυπλοκότητα.

Σε αντίθεση με τις περισσότερες μηχανές εκτέλεσης ροών εργασίας, το

ICENI (S. McGough, Young, Afzal, Newhouse, & Darlington, 2004) είναι σε θέση να υποστηρίξει την δέσμευση πόρων σε ορισμένες περιπτώσεις. Συγκεκριμένα, όταν καθοριστεί ότι ένας πόρος θα απαιτηθεί για ένα ορισμένο διάστημα χρόνου, μια δέσμευση εκφράζεται μέσω μιας διαδικασίας WS-Agreement (Battre, Wieder, & Ziegler, 2008). Προκειμένου να καθοριστεί ο χρόνος έναρξης εκτέλεσης κάθε συστατικού της ροής εργασίας χρησιμοποιείται ένα μητρώο όπου αποθηκεύονται πληροφορίες για την απόδοση του συγκεκριμένου συστατικού σε διαφορετικούς πόρους. Οι πληροφορίες αυτές χρησιμοποιούνται κατά την φάση δέσμευσης πόρων ώστε να εκτιμηθεί ο χρόνος έναρξης κάθε συστατικού της ροής εργασίας. Εάν επιτευχθεί συμφωνία για την δέσμευση ενός πόρου τότε τα συστατικά που θα τρέξουν σε αυτόν τον πόρο τοποθετούνται σε μία ουρά αναμονής μέχρι τον χρόνο έναρξής τους. Εντούτοις, η εργασία αυτή δεν εξετάζει το πρόβλημα ολιστικά, ώστε να βρεθεί η συνολικά βέλτιστη ροή εργασίας, καθώς αυτό αποτελεί ένα NP-hard πρόβλημα (McMahon & Florian, 1975). Αντ' αυτού, χρησιμοποιούνται ευριστικοί αλγόριθμοι για να προσεγγιστεί η βέλτιστη λύση, ενώ αλγόριθμοι χρονοπρογραμματισμού, όπως ο τυχαίος και ο bestof-n τυχαίος, έχουν τροποποιηθεί ώστε να λαμβάνουν υπόψη όλη τη ροή εργασίας κατά τον χρονοπρογραμματισμό κάθε συστατικού της. Επιπλέον, ο χρόνος έναρξης των εκτελέσεων είναι σαφώς και αυστηρά ορισμένος και το σύστημα είναι υπεύθυνο να ξεκινήσει την εκτέλεση της ροής εργασίας κατά την διάρκεια της δέσμευσης.

Σχετική είναι και η εργασία των (Claris, George, & Khaled, 2007) όπου περιγράφεται μία εφαρμογή του αλγορίθμου χρονοπρογραμματισμού καλύτερης τοποθέτησης (best-fit) με περιορισμούς προθεσμίας, ώστε να ελαχιστοποιηθεί ο κατακερματισμός και να επιτευχθεί η μέγιστη εκμετάλλευση

των πόρων. Ο αλγόριθμος επαναχρησιμοποιεί έννοιες της υπολογιστικής γεωμετρίας για να αντιμετωπίσει αποτελεσματικά τον κατακερματισμό των πόρων και να ικανοποιήσει τις απαιτήσεις του χρήστη, εντασσόμενος σε έναν ευρύτερο μηχανισμό ο οποίος μπορεί να τροποποιηθεί και να προσαρμοστεί τόσο σε πόρους δικτύου όσο και σε υπολογιστικούς πόρους. Συγκεκριμένα, καθορίζει εάν είναι εφικτό να χρονοπρογραμματιστεί η εργασία κατά τρόπο τέτοιο ώστε να τηρηθεί η προθεσμία της. Σε αυτή την περίπτωση, χρησιμοποιεί ένα σύνολο κριτηρίων για να επιλέξει έναν από πιθανώς πολλαπλούς πόρους, ενημερώνει το πρόγραμμά του και επιστρέφει στον χρήστη μία αναφορά στον πόρο. Διαφορετικά η εργασία απορρίπτεται. Εντούτοις, ο χρόνος έναρξης της εργασίας καθορίζεται αυστηρά και ο χρήστης μπορεί να τρέξει την εργασία μόνο μία φορά ανά δέσμευση. Επιπλέον, ο χρονοπρογραμματισμός που παρουσιάζεται σε αυτήν την εργασία εστιάζει στην προσπάθεια να ελαχιστοποιηθεί ο χρόνος απόκρισης αντί να μεγιστοποιηθεί η χρησιμοποίηση των πόρων. Τέλος, ο αλγόριθμος εξαρτάται από την ύπαρξη περιόδων πού δεν έχουν δεσμευτεί μεταξύ των ήδη υπαρχόντων δεσμεύσεων σε κάθε πόρο.

Ομοίως, η λύση που προσφέρεται από τον αλγόριθμο SSS (Farooq κ. συν., 2005) αποτελείται από περιόδους συνεχούς χρησιμοποίησης του πόρου, που ονομάζονται blocks, οι οποίες διαχωρίζονται μεταξύ τους με ανενεργές περιόδους. Για κάθε εργασία είναι γνωστός ο χρόνος έναρξής της βάσει του χρονοπρογραμματισμού που έχει υπολογιστεί. Όταν υπάρξει ένα καινούριο αίτημα, ο αλγόριθμος προσαρμόζει το πρόγραμμα ώστε να συμπεριλάβει τη νέα εργασία, αν αυτό είναι εφικτό. Ο αλγόριθμος προσδιορίζει αρχικά όλες εκείνες τις εργασίες στο πρόγραμμα του πόρου, που μπορούν να έχουν επιπτώσεις στη δυνατότητα πραγματοποίησης του νέου προγράμματος με το νέο αίτημα

και έπειτα προσπαθεί να υπολογίσει ένα εφικτό πρόγραμμα μόνο για αυτό το υποσύνολο των εργασιών. Όπως και στο (Claris κ. συν., 2007) η λύση αυτή δεν προωθεί την κοινή εκμετάλλευση των πόρων και βασίζεται σε ακριβείς χρόνους έναρξης της εκτέλεσης.

Οι Netto, Bubendorfer και Buyya (Netto κ. συν., 2007) ερεύνησαν την απόδοση του χρονοπρογραμματισμού με δεσμεύσεις. Για αυτόν το λόγο ανέπτυξαν έναν χρονοπρογραμματιστή Ποιότητας Υπηρεσίας που χρησιμοποιεί SLA για να σχεδιάσει αποτελεσματικά τις δεσμεύσεις που πρέπει να γίνουν σε υπολογιστικούς πόρους βάσει της ευελιξίας τους. Εισάγουν επίσης την έννοια των προσαρμοστικών χρονικών παραμέτρων Επιπέδου Υπηρεσίας, οι οποίες δεν είναι στατικές, αλλά ευέλικτες και προσαρμόζονται σύμφωνα με τις ανάγκες των χρηστών και τις πολιτικές των παρόχων. Εντούτοις, όπως και με τα (Claris κ. συν., 2007) και (Farooq κ. συν., 2005) ο σχεδιασμός βασίζεται στην εύρεση μίας διαθέσιμης χρονοσχισμής που καλύπτει τις ανάγκες τις εργασίας και της οποίας ο χρόνος έναρξης είναι γνωστός. Κατά συνέπεια, όλα τα εισερχόμενα αιτήματα για δέσμευση πόρων που επικαλύπτονται με κάποια πρότερη δέσμευση απορρίπτονται.

Στο (Singh, Kesselman, & Deelman, 2006) προτείνεται ένα μοντέλο παροχής πόρων κατά το οποίο η διαθεσιμότητα των πόρων περιγράφεται με τη μορφή σχισμών (slots), όπου κάθε σχισμή αντιπροσωπεύει των αριθμό των επεξεργαστών που είναι διαθέσιμοι για ένα ορισμένο χρονικό διάστημα με ορισμένο κόστος. Με βάση τα στοιχεία αυτά, διατυπώνεται ένα σχέδιο κατά το οποίο δεσμεύονται χρονικές σχισμές στους πόρους με στόχο την βελτιστοποίηση μίας μετρικής η οποία συνδυάζει το αναμενόμενο κόστος και τον χρόνο εκτέλεσης της εφαρμογής. Αυτό συνδυάζεται επίσης με δύο

ευριστικούς αλγορίθμους, τον Min-Min και έναν γενετικό αλγόριθμο. Στη συνέχεια η λύση ελέγχεται βάσει εφαρμογών που έχουν εκφραστεί ως ροές εργασίας. Εντούτοις, η εργασία αυτή επικεντρώνεται στη βελτιστοποίηση από την πλευρά του χρήστη και αγνοεί τη χρησιμοποίηση των πόρων, κάτι για το οποίο ενδιαφέρονται συνήθως οι πάροχοι. Επιπλέον, η προσέγγιση αυτή δεν εξετάζει τις προθεσμίες που μπορεί να έχουν οι εργασίες της ροής.

4.6 Η Γλώσσα BPEL

Η BPEL είναι μια γλώσσα ενορχήστρωσης (orchestration), όχι μια γλώσσα χορογραφίας (choreography). Η κύρια διαφορά μεταξύ ενορχήστρωσης και χορογραφίας είναι δυνατότητα εκτέλεσης και ελέγχου. Μια ενορχήστρωση καθορίζει μία εκτελέσιμη διαδικασία που περιλαμβάνει ανταλλαγές μηνυμάτων με άλλα συστήματα, έτσι ώστε οι ακολουθίες ανταλλαγής μηνυμάτων να ελέγχονται από το σχεδιαστή της ενορχήστρωσης. Μια χορογραφία καθορίζει ένα πρωτόκολλο για ομότιμες αλληλεπιδράσεις. Ένα τέτοιο πρωτόκολλο δεν είναι άμεσα εκτελέσιμο, καθώς επιτρέπει πολλές διαφορετικές υλοποιήσεις (διαδικασίες που συμμορφώνονται με αυτό). Η χορογραφία μπορεί να πραγματοποιηθεί, αφού τροποποιηθεί σε ενορχήστρωση, με τη σύνταξη μίας ενορχήστρωσης (π.χ. με τη μορφή μιας διαδικασίας BPEL) για κάθε μέλος που εμπλέκεται σε αυτό. Η ενορχήστρωση και η χορογραφία είναι διακρίσεις που βασίζονται στις αναλογίες: η ενορχήστρωση παραπέμπει στον κεντρικό έλεγχο (τον μαέστρο) της συμπεριφοράς ενός κατανεμημένου συστήματος (η ορχήστρα που αποτελείται από πολλούς μουσικούς), ενώ η χορογραφία αναφέρεται σε ένα κατανεμημένο σύστημα (η ομάδα χορού) που λειτουργεί σύμφωνα με κανόνες αλλά δεν έχει κεντρικό έλεγχο.

Η εστίαση της BPEL για τις σύγχρονες επιχειρηματικές διαδικασίες, καθώς και η ιστορία των WSFL και XLANG, οδήγησε την BPEL να υιοθετήσει τις διαδικτυακές υπηρεσίες ως έναν εξωτερικό μηχανισμό ανακοίνωσής της. Έτσι τα μηνύματα της BPEL εξαρτώνται από τη χρήση του Web Services Description Language (WSDL) 1.1 για την περιγραφή των εξερχόμενων και

εισερχόμενων μηνυμάτων.

Όπως αναφέρθηκε προηγουμένως, η BPEL είναι μια γλώσσα που χρησιμοποιείται για να περιγράψει τη λογική εκτέλεση των υπηρεσιών εφαρμογών διαδικτύου με τον καθορισμό των ροών ελέγχου τους και παρέχει έναν τρόπο για τις συμμετέχουσες υπηρεσίες να μοιράζονται ένα κοινό πλαίσιο. Ως συμμετέχουσες υπηρεσίες ορίζουμε αυτές που αλληλεπιδρούν με την BPEL διαδικασία.

Η BPEL βασίζεται σε διάφορες προδιαγραφές, όπως SOAP, WSDL και XML Schema, όπου η WSDL είναι ίσως η πιο σημαντική. Η WSDL είναι αυτό που κάνει μια υπηρεσία να μπορεί να χρησιμοποιηθεί εντός των σύνθετων υπηρεσιών που βασίζονται στην BPEL. Η BPEL επιτρέπει να οριστούν επιχειρησιακές διαδικασίες που αλληλεπιδρούν με συνεργαζόμενες υπηρεσίες μέσω WSDL περιγραφών.

Στη BPEL, η ροή ελέγχου (control flow) καθορίζει την ροή εκτέλεσης (execution flow) ως ένας κατευθυνόμενος ακυκλικός γράφος. Η γλώσσα έχει σχεδιαστεί έτσι ώστε να συνδυάζει τον συμβολισμό με χρήση μπλοκ (block oriented notation) με τον συμβολισμό των γράφων. Περιέχει κατασκευές που βοηθούν στην μοντελοποίηση δομημένων δραστηριοτήτων όπως: συνάθροιση, διακλάδωση, ταυτοχρονισμός, βρόχοι, εξαιρέσεις και χρονικοί περιορισμοί. Σύνδεσμοι χρησιμοποιούνται για να ορίσουν τις εξαρτήσεις ελέγχου μεταξύ δύο μπλοκ ορισμών: μία δραστηριότητα πηγή και μία δραστηριότητα στόχος. Οι δραστηριότητες μπορούν να ομαδοποιηθούν σε ένα πεδίο και με κάθε πεδίο συνδέονται τρεις τύποι χειριστών (handlers): χειριστές σφαλμάτων (fault handlers), χειριστές αποζημίωσης (compensation handlers) και χειριστές συμβάντων (event handlers). Όταν παρουσιαστεί ένα σφάλμα η κανονική ροή

εκτέλεσης σταματάει και ο έλεγχος περνάει στον κατάλληλο χειριστή.

Οι βασικές δραστηριότητες της BPEL μπορούν να περιγραφούν με τρεις τύπους μηνυμάτων: *invoke* για να κληθεί μία εργασία (operation) σε κάποιον *εταίρο* (partner), *receive* για να γίνει λήψη ενός μηνύματος εκτέλεσης από κάποιον εταίρο και *reply* για να σταλεί ένα μήνυμα απάντησης σε ένα ληφθέν μήνυμα εκτέλεσης. Για κάθε μήνυμα πρέπει κανείς να συνδέσει έναν εταίρο, το οποίο απαγορεύει την ανταλλαγή μηνυμάτων μεταξύ εσωτερικών στοιχείων. Επιπλέον, δεν υπάρχει η δυνατότητα συσχέτισης ενός χρονικού ορίου με την δραστηριότητα *invoke*, πράγμα που σημαίνει ότι μπορεί να μπλοκαριστεί ολόκληρο το σύστημα σε περίπτωση που δε δοθεί απάντηση σε μία κλήση.

Η ροή δεδομένων μπορεί να διαχειρισθεί με χρήση μεταβλητών συγκεκριμένης εμβέλειας (scoped variables). Η είσοδος και η έξοδος των δραστηριοτήτων φυλάσσεται σε μεταβλητές και τα δεδομένα μεταφέρονται μεταξύ δύο (ή περισσοτέρων) δραστηριοτήτων χάρη σε κοινούς χώρους δεδομένων μεταξύ των Υπηρεσιών Ιστού. Η δραστηριότητα *assign* χρησιμοποιείται για την αντιγραφή δεδομένων μεταξύ δύο μεταβλητών.

Η BPEL προτείνει επίσης ένα πρωτόκολλο αποζημίωσης για τον χειρισμό της ροής συναλλαγών σε ιδιαιτέρως μακροχρόνιες συναλλαγές. Μπορεί κανείς να ορίσει είτε ένα χειριστή σφάλματος είτε έναν χειριστή αποζημίωσης. Οι χειριστές ορίζονται εντός ενός πεδίου (scope) όπου ένας χειριστής σφάλματος ορίζει εναλλακτικές διαδρομές εντός του πεδίου, ενώ ένας χειριστής αποζημίωσης χρησιμοποιείται για να αντιστραφεί η εργασία που εκτελέστηκε εντός ενός τερματισθέντος πεδίου.

Η γλώσσα είναι δυνατόν να χρησιμοποιηθεί για την μοντελοποίηση διαφόρων ειδών διαδραστικότητας μεταξύ υπηρεσιών από απλές

αλληλεπιδράσεις χωρίς την αποθήκευση της κατάστασης μέχρι μακροχρόνιες ασύγχρονες επικοινωνίες όπου η κατάσταση αποθηκεύεται από κλήση σε κλήση. Διαφορετικοί τύποι *Partner Link* μπορούν να χρησιμοποιηθούν για την μοντελοποίηση διαφορετικών εταίρων και ομάδες συσχέτισης (correlation sets) αντιπροσωπεύουν τις συνομιλίες, διατηρώντας έτσι την κατάσταση της επικοινωνίας. Η χορογραφία των συνεργατικών επιχειρηματικών διαδικασιών ορίζεται ως μια αφηρημένη διαδικασία.

5 Λειτουργικά Συστήματα Πραγματικού Χρόνου

5.1 Εισαγωγή στα Λειτουργικά Συστήματα Πραγματικού Χρόνου

Οι διαδραστικές εφαρμογές νέας γενιάς έχουν υψηλές απαιτήσεις σε υπολογιστική ισχύ, αποθηκευτικό χώρο και δυνατότητες δικτύωσης καθώς και αυστηρούς περιορισμούς χρονισμού. Στο παρελθόν τέτοιες εφαρμογές ήταν διαθέσιμες ως επαγγελματικές με υψηλό κόστος κτήσης. Η ανάπτυξη όμως των δυνατοτήτων των ευρείας κυκλοφορίας υπολογιστών έχει ωθήσει πολλές εταιρείες να προσφέρουν παρόμοιες λύσεις ως εφαρμογές που είναι δυνατόν να εκτελεστούν με την χρήση Λειτουργικών Συστημάτων Γενικής Χρήσης (ΛΣ-ΓΧ) (General Purpose Operating Systems). Αυτό επιτρέπει την ταυτόχρονη χρήση των φυσικών πόρων από πολλαπλές εφαρμογές, μειώνοντας το κόστος λειτουργίας.

Δυστυχώς τα Λειτουργικά Συστήματα Γενικής Χρήσης δεν είναι σχεδιασμένα να παρέχουν την δυνατότητα σε εφαρμογές να ορίζουν σαφείς

χρονικούς περιορισμούς κατά την εκτέλεσή τους. Σε τέτοιες περιπτώσεις απαιτείται η χρήση εξειδικευμένων ΛΣ, γνωστά ως Λειτουργικά Συστήματα Πραγματικού Χρόνου (ΛΣ-ΠΧ) (Real-Time Operating Systems). Ένα ΛΣ-ΠΧ παρέχει τα χαρακτηριστικά εκείνα τα οποία επιτρέπουν σε μία εφαρμογή να εκτελεστεί με σαφώς καθορισμένο και προβλέψιμο τρόπο αναφορικά με τον χρόνο. Έτσι, όλα τα τμήματα του πυρήνα χαρακτηρίζονται από μία γνωστή διάρκεια χειρότερης περίπτωσης ενώ είναι δυνατόν να ορισθούν οι καθυστερήσεις χρονοπρογραμματισμού και διακοπής. Παράλληλα, ένα σύνολο από πολιτικές χρονοπρογραμματισμού πραγματικού χρόνου είναι διαθέσιμο στον σχεδιαστή του συστήματος. Ο χρόνος μπορεί να μετρηθεί με υψηλή ακρίβεια (συνήθως σε μεγέθη μικρότερα του millisecond), ενώ υπάρχουν εργαλεία που επιτρέπουν την εκτίμηση του χρόνου εκτέλεσης χειρίστων συνθηκών καθώς και την ανάλυση των επιλογών χρονοπρογραμματισμού. Δυστυχώς τα ΛΣ-ΠΧ είναι συνήθως σχεδιασμένα για ενσωματωμένα συστήματα και έτσι παρουσιάζουν αδυναμίες στην υποστήριξη γενικού υλισμικού καθώς και λογισμικού.

Για το λόγο αυτό, τα τελευταία χρόνια έχουν υπάρξει πολλές προσπάθειες να ενταχθούν τεχνολογίες πραγματικού χρόνου σε ΛΣ-ΓΧ, ιδιαιτέρως στο Linux λόγω της ανοιχτής φύσης του ως ελεύθερου λογισμικού.

Διάφορα μειονεκτήματα καθιστούν τον πυρήνα του Linux, καθώς και τα περισσότερα από τα ΛΣ-ΓΣ, ιδιαίτερα ακατάλληλο για τη λειτουργία εφαρμογών πραγματικού χρόνου: η μονολιθική δομή του πυρήνα και η μεγάλη ποικιλία των οδηγών, η αδυναμία να κρατηθούν υπό έλεγχο όλα τα μη προεκτοπίσιμα τμήματα που ενδεχομένως έχουν προσθέσει οι οδηγοί, η γενική δομή του πυρήνα διαχείρισης διακοπών που στοχεύει στην φορητότητα

σε βάρος της καθυστέρησης και άλλα.

Ως εκ τούτου, η καθυστέρηση του χρόνου που αντιμετωπίζουν οι ευαίσθητες δραστηριότητες μπορεί να φτάνει τα δεκάδες ή εκατοντάδες χιλιοστά του δευτερολέπτου. Αυτό καθιστά τα ΛΣ-ΓΧ ακατάλληλα για εφαρμογές ελαστικού πραγματικού χρόνου. Αντίθετα, εφαρμογές ελαστικού πραγματικού χρόνου μπορούν να τρέξουν πολύ καλά μέσα σε ένα τέτοιο περιβάλλον, ειδικά όταν ο αρχικός πυρήνας τροποποιηθεί ούτως ώστε να ενσωματώσει τις αναγκαίες δυνατότητες πραγματικού χρόνου.

Κατά την ανάπτυξη του προτεινόμενου ΣΔΡΕ έγινε φανερό ότι για να μπορεί να παρέχει εγγυήσεις ως προς την ανάγκες πραγματικού χρόνου των εφαρμογών, θα πρέπει να υπάρχει κατάλληλη υποστήριξη από το λειτουργικό σύστημα στο οποίο εδράζεται. Έτσι, σκοπός του κεφαλαίου αυτού είναι να παρέχει μια επισκόπηση των προσεγγίσεων που έχουν παρουσιαστεί στη βιβλιογραφία με στόχο την ενσωμάτωση δυνατοτήτων πραγματικού χρόνου σε Λειτουργικά Συστήματα Γενικής Χρήσης και να παρουσιάσει την λύση η οποία επιλέχτηκε να χρησιμοποιηθεί στο προτεινόμενο ΣΔΡΕ.

5.2 Τεχνολογικό Υπόβαθρο

Παρότι δεν αποτελεί ένα σύστημα πραγματικού χρόνου, ο πυρήνας 2.6 του Linux περιλαμβάνει ένα σύνολο χαρακτηριστικών που τον καθιστούν ιδιαίτερα κατάλληλο για εφαρμογές ελαστικού πραγματικού χρόνου. Πρώτον, είναι ένας πλήρως προεκτοπίσιμος πυρήνας, όπως και τα περισσότερα από τα υπάρχοντα συστήματα πραγματικού χρόνου και πολλή προσπάθεια έχει δαπανηθεί για τη μείωση των μη προεκτοπίσιμων τμημάτων του (η

κύρια πηγή των καθυστερήσεων του πυρήνα). Αξίζει να σημειωθεί ότι ο πυρήνας 2.6 εισήγαγε ένα νέο μηχανισμό χρονοπρογραμματισμού που είναι οριοθετημένος ως προς τον χρόνο εκτέλεσης, με αποτέλεσμα μια ιδιαίτερα μειωμένη καθυστέρηση χρονοπρογραμματισμού. Επίσης, στην τελευταία έκδοση του πυρήνα έχει εισαχθεί ένα αρθρωτό πλαίσιο που επιτρέπει την εύκολη ενσωμάτωση άλλων πολιτικών χρονοπρογραμματισμού. Δεύτερον, μολονότι είναι ένας γενικής χρήσης πυρήνας που βασίζεται στον καταμερισμό χρόνου, περιλαμβάνει επεκτάσεις πραγματικού χρόνου στο πρότυπο POSIX ("IEEE Standard for Information Technology- Portable Operating System Interface (POSIX) Base Specifications, Issue 7", 2008) όπως: τις SCHED_FIFO και SCHED_RR πολιτικές δρομολόγησης των διεργασιών, το πρωτόκολλο κληρονομικής προτεραιότητας για την αποφυγή του γνωστού προβλήματος αναστροφής προτεραιότητας καθώς και επεκτάσεις πραγματικού χρόνου που σχετίζονται με τα σήματα και τα χρονόμετρα. Επίσης, φαίνεται να υπάρχει ένα αυξανόμενο ενδιαφέρον για την εφαρμογή της SCHED_SPORADIC κλάσης χρονοπρογραμματισμού πραγματικού χρόνου. Αυτά τα χαρακτηριστικά μπορεί να φανούν χρήσιμα για την ανάπτυξη συστημάτων πραγματικού χρόνου. Τρίτον, η υποστήριξη που εισήχθη πρόσφατα στον πυρήνα για υψηλής ανάλυσης χρονόμετρα είναι υψίστης σημασίας για την υλοποίηση μηχανισμών χρονοπρογραμματισμού υψηλής ακρίβειας, καθώς και για τη γενική απόδοση των εφαρμογών ελαστικού πραγματικού χρόνου.

Επιπλέον, οι πρόσφατες διορθώσεις που πρότεινε η ομάδα του Ingo Molnar (Mckenney, Molnar, Sarma, & Bhattacharya, 2006) στο πλαίσιο διαχείρισης διακοπών του πυρήνα, με στόχο την ενσωμάτωση των οδηγών συσκευών σε νήματα του πυρήνα, είναι ιδιαίτερα σημαντικές καθώς τέτοιες

προσεγγίσεις θα αυξήσουν σε μεγάλο βαθμό την προβλεψιμότητα της συμπεριφοράς του πυρήνα.

Όσον αφορά τον χρονοπρογραμματιστή των διαδικασιών, πρόσφατα έχει υποστεί δύο σημαντικές αλλαγές: από την έκδοση 2.4 στην έκδοση 2.6, ο χρονοπρογραμματιστής αναδιοργανώθηκε έτσι ώστε να μειώσει την υπολογιστική του πολυπλοκότητα, από γραμμική ως προς τον αριθμό των εργασιών του σε O(1). Από την έκδοση 2.6.23, ένα νέο αρθρωτό πλαίσιο έχει εισαχθεί από τον Ingo Molnar, μαζί με μια πλήρη επανασυγγραφή της πολιτικής χρονοπρογραμματισμού (η SCHEDOTHER_POSIX κατηγορία), που ονομάζεται πλέον Απολύτως Δίκαιος Χρονοπρογραμματιστής - Completely Fair Scheduler (CFS). Ο CFS παρουσιάζει ιδιότητες ισότητας μεταξύ διεργασιών που εκτελούνται, μοιάζοντας με ένα σύστημα Γενικευμένης Αναλογιστικής Δικαιοσύνης Generalized Proportional Fair (GPF) λόγω της χρήσης του μηχανισμού timestamping, αλλά δυστυχώς δεν ανταποκρίνεται σε κανένα αλγόριθμο με γνωστές ιδιότητες. Οι εργασίες χαρακτηρίζονται από ένα ρυθμιζόμενο βάρος και μία ποσότητα που ονομάζεται vruntime. Κάθε φορά που μία εργασία εκτελείται για ένα χρονικό διάστημα t, η τιμή του vruntime αυξάνεται κατά $t * task\_weight / total\_weight$, όπου *total_weight* είναι το συνολικό βάρος των εργασιών που τρέχουν τη δεδομένη στιγμή. Όταν μια εργασία έχει ενεργοποιηθεί, τότε εκχωρείται μια vruntime που ποικίλλει ανάλογα με τη διαμόρφωση του συστήματος καθώς και βάσει ευριστικών μηχανισμών (heuristics), οι οποίοι τείνουν να δώσουν μια μικρή vruntime σε διαδραστικές διαδικασίες. Τέλος, οι εργασίες χρονοπρογραμματίζονται βάσει της τιμής vruntime. Μέχρι σήμερα, δεν έχουν υπάρξει επίσημες εγγυήσεις για την υπηρεσία που παρέχεται στις διεργασίες που έχουν προγραμματιστεί με CFS.

Επίσης, ο τρέχων πυρήνας του Linux υποστηρίζει μερικώς τον ιεραρχικό χρονοπρογραμματισμό, ένα γνώρισμα χαρακτηριστικό σε ΛΣ πραγματικού χρόνου. Τόσο εργασίες πραγματικού χρόνου όσο και SCHED_OTHER μπορούν να οργανωθούν κατά ομάδες και για κάθε ομάδα είναι δυνατό να περιοριστεί ο χρόνος κεντρικής μονάδας επεξεργασίας που καταναλώνεται, καθώς επίσης και να διαμορφωθεί το μερίδιο ΚΜΕ που χρησιμοποιείται από τις διεργασίες SCHED_OTHER της.

Λόγω της εσωτερικής αρχιτεκτονικής του χρονοπρογραμματιστή (που δεν είναι πλήρως ιεραρχικός υπό μια παραδοσιακή έννοια) ένας συνδυασμός των δύο αποτελεσμάτων δεν μπορεί να ληφθεί. Με άλλα λόγια, δεν είναι δυνατό να περιοριστεί ο χρόνος κεντρικής μονάδας επεξεργασίας μιας ολόκληρης ομάδας (διεργασίες πραγματικού χρόνου + SCHED_OTHER) και δεν είναι δυνατό να περιοριστεί το μερίδιο της Κεντρικής Μονάδας Επεξεργασίας (ΚΜΕ) που ορίζεται σε μια ολόκληρη ομάδα. Η πρόθεση να ενισχυθεί ο μηχανισμός περιορισμού πραγματικού χρόνου (throttling) έχει εκφραστεί στις δημόσιες συζητήσεις, αλλά κανένας κώδικας δεν έχει γραφτεί προς αυτή την κατεύθυνση μέχρι τώρα.

Συμπερασματικά, ο πυρήνας του Linux, παρά τις πρόσφατες βελτιώσεις στο πεδίο της υποστήριξης πραγματικού χρόνου, στερείται αυτήν την περίοδο ενός χρονοπρογραμματιστή ΚΜΕ ικανού να ικανοποιήσει τις απαιτήσεις συγχρονισμού των όλο και περισσότερο αναγκαίων κατανεμημένων εφαρμογών πραγματικού χρόνου.

Στη συνέχεια παρουσιάζεται η σχετική εργασία στον τομέα της υποστήριξης πραγματικού χρόνου για ΛΣ-ΓΧ. Διάφορες σχετικές τροποποιήσεις στα ΛΣ-ΓΧ έχουν εμφανιστεί στη βιβλιογραφία. Έτσι,

ερευνάται η σχετική εργασία στον τομέα του προγραμματισμού πραγματικού χρόνου για την ΚΜΕ. Κατόπιν, παρουσιάζονται προσεγγίσεις στις οποίες ο χρονοπρογραμματισμός πολλαπλών, ετερογενών και ενδεχομένως κατανεμημένων πόρων είναι ενσωματωμένος σε ένα κοινό πλαίσιο. Κατόπιν, χάριν πληρότητας, οι πιο σχετικές και πρόσφατες προσεγγίσεις, μεταξύ των λύσεων που έχουν προταθεί, περιγράφονται με περισσότερες λεπτομέρειες.

5.3 Χρονοπρογραμματισμός ΚΜΕ Πραγματικού Χρόνου σε ΛΣ-ΓΧ

Υπάρχουν διάφοροι πυρήνες ΛΣ-ΓΧ που είναι συμβατοί με τις επεκτάσεις πραγματικού χρόνου του POSIX ("IEEE Standard for Information Technology- Standardized Application Environment Profile (AEP)-POSIX Realtime and Embedded Application Support", 2004). Εντούτοις, το κύριο μειονέκτημα τέτοιων επεκτάσεων είναι ότι υπάρχουν διάφορα προαιρετικά μέρη και οι περισσότερες υλοποιήσεις του πρωτοτύπου περιορίζονται στην υλοποίηση του χρονοπρογραμματισμού σταθερής προτεραιότητας, μερικές φορές με την προσθήκη υψηλής ευκρίνειας χρονομέτρων και του πρωτοκόλλου κληρονομικής προτεραιότητας για την αποφυγή της αντιστροφής προτεραιότητας (Sha κ. συν., 2004). Επίσης, ένα κύριο χαρακτηριστικό που συνήθως δεν εφαρμόζεται είναι η χρονική ιδιοκτησία απομόνωσης (Buttazzo κ. συν., 2005), όπως παρέχεται από τον χρονοπρογραμματιστή Σποραδικού Διακομιστή (Sporadic Server) ("IEEE Standard for Information Technology-Standardized Application Environment Profile (AEP)-POSIX Realtime and

Embedded Application Support", 2004). Χωρίς έναν τέτοιο μηχανισμό, μία διεργασία υψηλότερης προτεραιότητας τρέχει ανενόχλητη έως ότου εμποδιστεί, ανεξάρτητα από το χρόνο υπολογισμού που μπορεί να είχε προβλεφθεί κατά την ανάλυση του συστήματος. Αυτό οδηγεί στην πιθανή διάσπαση των εγγυήσεων που προσφέρονται στις διεργασίες χαμηλότερης προτεραιότητας. Επομένως, τέτοιες προσεγγίσεις είναι κατάλληλες για τις παραδοσιακές εφαρμογές ανελαστικού πραγματικού χρόνου όπου όλες οι διεργασίες που εκτελούνται στο σύστημα έχουν ελεγχθεί λεπτομερώς ή έχουν αποδειχθεί τυπικά.

Για τον χρονοπρογραμματιστή πραγματικού χρόνου της ΚΜΕ έχουν προταθεί τροποποιήσεις ανελαστικού πραγματικού χρόνου στον πυρήνα του Linux, όπως το rT-Linux (`http://www.rtlinuxfree.com`), που προτείνεται από τον Yodaiken και λοιπούς (Ayers & Barabanov, 1997) και RTAI (`http://www.rtai.org`), που προτείνεται από τον Dozio και λοιπούς (Dozio & Mantegazza, 2003). Σε αυτές τις προσεγγίσεις, ένας μικρο-πυρήνας προστίθεται μεταξύ του υλισμικού και του Linux, ο οποίος τρέχει ως υπόβαθρο όποτε δεν υπάρχει ενεργή διαδικασία πραγματικού χρόνου στο σύστημα.

Αυτό επιτρέπει το σεβασμό των πολύ σφιχτών περιορισμών χρονισμού (επιπέδου μικροδευτερόλεπτων) που χαρακτηρίζουν τη βιομηχανική αυτοματοποίηση και τις ρομποτικές εφαρμογές. Το κύριο μειονέκτημα τέτοιων προσεγγίσεων είναι ότι οι εφαρμογές θα πρέπει να γράφονται ως κομμάτι του ίδιου του πυρήνα. Αυτό αποτελεί σοβαρό εμπόδιο για την μεγάλη κατηγορία πολυμεσικών εφαρμογών που θα επωφελούνταν πολύ από πολιτικές πραγματικού χρόνου. Εντούτοις, μια λεπτομερής περιγραφή αυτών των προσεγγίσεων ακολουθεί κατωτέρω, λόγω της ιστορικής σημασίας τους.

Προκειμένου να υπερνικηθούν αυτοί οι περιορισμοί, άλλες προσεγγίσεις

στόχευαν ρητά στις εφαρμογές ελαστικού πραγματικού χρόνου, με την προσθήκη μιας πολιτικής χρονοπρογραμματισμού ως επέκταση στον ίδιο τον πυρήνα, περιλαμβάνοντας συνήθως έναν μηχανισμό χρονικής απομόνωσης. Μια τέτοια προσέγγιση, επιτρέπει τη συνύπαρξη των εφαρμογών ελαστικού πραγματικού χρόνου και καλύτερης-προσπάθειας (best-effort), μέσα σε έναν πυρήνα ενός ΛΣ-ΓΧ με ενδεχομένως μακροχρόνια, μη-προεκτοπίσιμα τμήματα, καθιστώντας αδύνατο να παρασχεθούν εγγυήσεις μη ελαστικού πραγματικού χρόνου. Εντούτοις, για τις εφαρμογές ελαστικού πραγματικού χρόνου όπως οι πολυμεσικές, αυτή η προσέγγιση επιτρέπει την προβλεψιμότητα στη χρονική συμπεριφορά των εφαρμογών. Το γεγονός αυτό οδηγεί σε σημαντικές βελτιώσεις στην ποιότητα υπηρεσίας που παρέχεται προς τον τελικό χρήστη. Μια επισκόπηση αυτών των προσεγγίσεων έχει πραγματοποιηθεί από τον Gopalan (Gopalan, 2001).

Αξιοπρόσεκτες εργασίες είναι αυτή για την προσθήκη κράτησης πόρων (Mercer, Savage, & Tokuda, 1993) στο λειτουργικό σύστημα Microsoft Windows NT από τον Jones (Jones & Regehr, 1999) και αυτή από τον Rajkumar (Rajkumar, Juvva, Molano, & Oikawa, 1997). Η τελευταία αποτελεί μια τροποποίηση στον πυρήνα του Linux, που εμπνέεται κατά ένα μεγάλο μέρος από την προγενέστερη εργασία των ίδιων συντακτών πάνω στο rT-Mach (Tokuda, Nakajima, & Rao, 1990). Υπάρχουν επίσης προσπάθειες για την φορητότητα παρόμοιων λύσεων σε διαφορετικούς πυρήνες ΛΣ-ΓΧ (Oikawa & Rajkumar, 1999). Μια προσπάθεια στη φορητότητα ενός χρονοπρογραμματιστή πραγματικού χρόνου σε διαφορετικά λειτουργικά συστήματα (όπως της Microsoft, το Unix και τις οικογένειες Linux), είναι και ο χρονοπρογραμματιστής DSRT (`http://cairo.cs.uiuc.edu/software/DSRT-2/dsrt-2.html`) (Yuan & Nahrstedt, 2003).

Εντούτοις, όπως και στη σειρά αρχιτεκτονικών GRACE (Vardhan κ. συν., 2005), η εστίαση φαίνεται να περιορίζεται αποκλειστικά στο χρονοπρογραμματισμό της ΚΜΕ.

Επίσης, οι χρονοπρογραμματιστές ελαστικού πραγματικού χρόνου για Linux έχουν ερευνηθεί και έχουν εφαρμοστεί στα πλαίσια διάφορων ευρωπαϊκών προγραμμάτων, όπως η εφαρμογή CBS για Linux που αναπτύχθηκε κατά τη διάρκεια του προγράμματος OCERA (Open Components for Embedded Real-time Applications) καθώς και η εξέλιξή του, ο χρονοπρογραμματιστής AQuoSA για Linux (Palopoli, Cucinotta, Marzario, & Lipari, 2009), που αναπτύχθηκε από τον Tommaso Cucinotta κατά τη διάρκεια του προγράμματος FRESCOR.

Στο (Faggioli, Mancina, Checconi, & Lipari, 2008) προτείνεται μια POSIX-συμβατή εφαρμογή του αλγορίθμου σποραδικού εξυπηρετητή (και παραλλαγών αυτού) για την ενίσχυση του τρέχοντος μηχανισμού throttling του πυρήνα του Linux με βελτιωμένη δυνατότητα προβλεψιμότητας, χρονικής απομόνωσης και ανάλυσης. Στο (Checconi, Cucinotta, & Faggioli, 2009) προτείνεται ένα νέο ιεραρχικό υβριδικό πλαίσιο σχεδιασμού, βασισμένο σε έναν συνδυασμό του αλγορίθμου μερικού EDF και σφαιρικού FP, σχεδιασμένο να ταιριάζει όσο το δυνατόν περισσότερο (και να προσκρούει όσο το δυνατόν λιγότερο) στον τρέχοντα κώδικα της κλάσης πραγματικού χρόνου. Μια παρόμοια εργασία γίνεται στα πλαίσια του προγράμματος ACTORS (προσαρμοστικότητα και έλεγχος των πόρων στα ενσωματωμένα συστήματα), αλλά με μια εστίαση στα ενσωματωμένα συστήματα ενός κόμβου χωρίς οποιαδήποτε κατανεμημένα γνωρίσματα. Εδώ ερευνώνται τόσο ο αλγόριθμος μερικού EDF όσο και ολικού EDF.

Υπάρχουν προγενέστερες εργασίες που ενσωματώνουν τον χρονοπρογραμματισμό πραγματικού χρόνου ετερογενών πόρων σε μια αρχιτεκτονική για τη διαχείρισή τους, όπως αυτή από τον Zhang (R. Zhang, Lu, Abdelzaher, & Stankovic, 2002) ή το Hola QoS (Valls, Alonso, Ruiz, & Groba, 2003). Το τελευταίο είναι μια αρχιτεκτονική που στοχεύει συγκεκριμένα στις ανάγκες των καταναλωτικών ενσωματωμένων ηλεκτρονικών συστημάτων πολυμέσων, προσφέροντας εύκαμπτη διαχείριση των πόρων καθώς και προσαρμοστικότητα. Το πρόγραμμα Eclipse/BSD (Blanquer κ. συν., 1999) ενσωματώνει χρονοπρογραμματισμό πραγματικού χρόνου της ΚΜΕ, του δικτύου και της προσπέλασης στο δίσκο και εκθέτει στις εφαρμογές μια διεπαφή βασισμένη σε αρχεία. Εντούτοις, το πρόγραμμα δεν εξετάζει τις κατανεμημένες εφαρμογές πραγματικού χρόνου. Πιο πρόσφατα, προτάθηκε το MURALS (Gopalan & Kang, 2007), μια κατανεμημένη αρχιτεκτονική πραγματικού χρόνου που βασίζεται στο TimeSys Linux (`http://www.timesys.com`), η οποία υποστηρίζει εφαρμογές πραγματικού χρόνου με περιορισμούς που χρησιμοποιούν κατανεμημένους ετερογενείς πόρους, όπως οι δίσκοι, η ΚΜΕ και οι συνδέσεις δικτύων. Η αρχιτεκτονική ενσωματώνει ένα σφαιρικό σχεδιασμό ελέγχου αποδοχής που λαμβάνει υπόψη ολόκληρο τον γράφο εξάρτησης της εφαρμογής.

Η ερευνητική ομάδα Nahrstedt εργάστηκε επίσης στο QualMan (Nahrstedt, Chu, & Narayan, 1998), μία κατανεμημένη αρχιτεκτονική πραγματικού χρόνου κατανομής των πόρων που υποστηρίζει επίσης την κατανομή δικτύων, δίσκων και μνήμης, με την εφαρμογή πρωτοτύπων στο Solaris OS. Εντούτοις, οι ίδιοι συντάκτες τονίζουν ότι ο βαθμός διαμορφωσιμότητας της αρχιτεκτονικής τους είναι κάπως περιορισμένος και

ότι θα ωφελούταν από μία σχεδίαση προσανατολισμένη προς το CORBA.

Στην πραγματικότητα, η προδιαγραφή CORBA έχει επεκταθεί για να προσφέρει την ικανότητα επαναχρησιμοποίησης στο CORBA Component Model (CCM), το οποίο καλύπτει επίσης πτυχές ποιότητας υπηρεσίας. Παραδείγματος χάριν, αυτό έχει εφαρμοστεί στο Component-Integrated ACE ORB (Schantz κ. συν., 2003). Το TAO (D. C. Schmidt, Levine, & Mungee, 1998) αποτελεί μια υλοποίηση σε γλώσσα C++ της προδιαγραφής CORBA πραγματικού χρόνου (Fay-Wolfe κ. συν., 2000), η οποία εκθέτει κάποιες θεμελιώδεις λειτουργίες των κατανεμημένων εφαρμογών πραγματικού χρόνου μέσω του CORBA.

Επίσης, το TAO έχει ενσωματωθεί στο QuO (Schantz, Loyall, Rodrigues, & Schmidt, 2006), ένα πλαίσιο που εκμεταλλεύεται τις ικανότητες του CORBA για να μειωθεί ο αντίκτυπος της διαχείρισης της Ποιότητας Υπηρεσίας στον κώδικα εφαρμογής. Το αποτέλεσμα είναι ένα υλικολογισμικό για τον προσαρμοστικό έλεγχο Ποιότητας Υπηρεσίας που χρησιμοποιεί χρονοπρογραμματισμό πραγματικού χρόνου στα επίπεδα υπολογισμού και δικτύου. Πρόσφατα, μια παρόμοια προσέγγιση (Shankaran, Koutsoukos, Schmidt, Xue, & Lu, 2006) έχει ακολουθηθεί στην αρχιτεκτονική HiDRA για την ιεραρχική διαχείριση πολλαπλών πόρων σε κατανεμημένα συστήματα πραγματικού χρόνου. Εντούτοις, αυτές οι εργασίες ασχολούνται με ζητήματα σχετικά με την παρακολούθηση της συμπεριφοράς της εφαρμογής κατά τον χρόνο εκτέλεσης και τη δυναμική προσαρμογή της κατανομής των πόρων ή/και τη συμπεριφορά της εφαρμογής στις συνεχώς μεταβαλλόμενες ανάγκες τους, με αποτέλεσμα να εξετάζουν μόνο οριακά ζητήματα σχετικά με τους χαμηλού επιπέδου μηχανισμούς που είναι απαραίτητοι για την

εγγύηση των περιορισμών συγχρονισμού που απαιτούνται από τις εφαρμογές πραγματικού χρόνου. Στο (Eide, Stack, Regehr, & Lepreau, 2004), παρότι παρουσιάζεται ένα CORBA-βασισμένο υλικολογισμικό για τη διαχείριση της ΚΜΕ σε κατανεμημένα συστήματα, εντούτοις δεν εξετάζονται άλλοι πόροι.

Επιπλέον, αξίζει να αναφερθεί η αρχιτεκτονική (Cucinotta κ. συν., 2009) που αναπτύχθηκε στα πλαίσια του προγράμματος ri-MACS για κατανεμημένες εφαρμογές πραγματικού χρόνου στην περιοχή της αυτοματοποίησης εργοστασίων. Η αρχιτεκτονική στηρίζεται στις δυνατότητες που παρέχουν οι υπηρεσιοστραφείς υποδομές για ανακάλυψη των πόρων και τις δυνατότητες πραγματικού χρόνου που αυτοί έχουν, για αυτοδιαμόρφωση, για την ανοχή τους σε σφάλματα και τη δυνατότητα διαπραγμάτευσης των παραμέτρων χρονοπρογραμματισμού. Η αρχιτεκτονική ri-MACS εκμεταλλεύεται επίσης το AQuoSA (Palopoli κ. συν., 2009)) ως χαμηλού επιπέδου χρονοπρογραμματιστή ΚΜΕ, αλλά στερείται υποστήριξης για πολλαπλάσιους ετερογενείς πόρους καθώς και ένα σαφώς καθορισμένο και ενοποιημένο API που να επιτρέπει στις εφαρμογές να χρησιμοποιήσουν ικανότητες πραγματικού χρόνου.

Το 2002, ο Ravindran εισήγαγε ένα εννοιολογικό πλαίσιο (Ravindran, 2002) όπου περιγράφονται τα στοιχεία που είναι απαραίτητα σε ένα υλικολογισμικό για κατανεμημένες εφαρμογές πραγματικού χρόνου: μια γλώσσα περιγραφής συστημάτων που περιλαμβάνει την προδιαγραφή απαιτήσεων ποιότητας υπηρεσίας και πραγματικού χρόνου, εφαρμογές χρόνου εκτέλεσης για τη διαχείριση των πόρων, την ανίχνευση και αποκατάσταση των σφαλμάτων, την ανίχνευση των παραβιάσεων των περιορισμών πραγματικού χρόνου και των ενεργειών αποκατάστασης που απαιτούνται (δηλ. δυναμικές αλλαγές στην κατανομή των πόρων). Εντούτοις, η εργασία είναι αρκετά

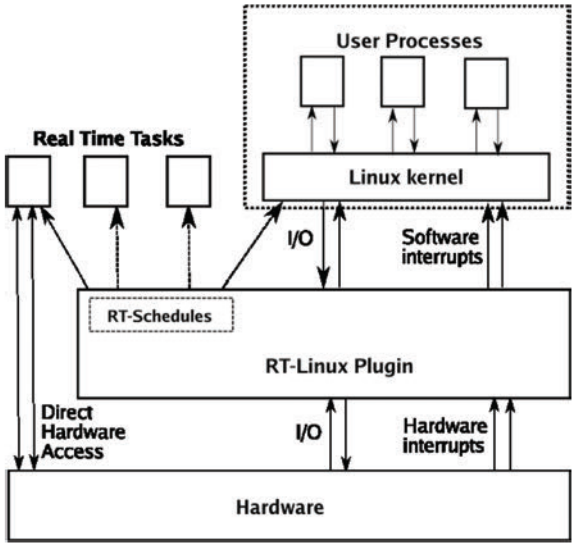
αφηρημένη και, ακόμα κι αν αναφέρεται σε μία πρωτότυπη υλοποίηση σε C στο ΛΣ Solaris, δεν εξετάζει μερικές ουσιαστικές λεπτομέρειες όπως τον τύπο σχεδιασμού που χρησιμοποιείται στο επίπεδο των πόρων ή εάν η εφαρμογή είναι ή όχι φορητή. Επιπλέον, το πλαίσιο Ravindran δίνει έμφαση στην ανάγκη για την μετανάστευση των εφαρμογών στους κόμβους ώστε να είναι εφικτή η ανοχή σε σφάλματα υλικού ή για την ενίσχυση της απόδοσης των εφαρμογών των οποίων οι απαιτήσεις φόρτου εργασίας παρεκκλίνουν από αυτήν που προβλέπεται, αλλά δεν δίνει οποιαδήποτε λεπτομέρεια στον τρόπο με τον οποίο ένας τέτοιος μηχανισμός θα μπορούσε να εφαρμοστεί και να υποστηριχθεί από το υλικολογισμικό (δηλαδή, δεν υπάρχει καμία συζήτηση για την πιθανή χρήση τεχνολογιών εικονικοποίησης (virtualization) ή check-pointing). Επίσης, το πλαίσιο χρησιμοποιεί το TCP/IP για τις επικοινωνίες στο επίπεδο υλικολογισμικού. Έτσι έχει υλοποιήσει από την αρχή χαρακτηριστικά γνωρίσματα, όπως την εγγραφή και την ανακάλυψη υπηρεσιών, τα οποία είναι τυποποιημένα σε ένα σύγχρονο κατανεμημένο υλικολογισμικό όπως το CORBA.

Πιο πρόσφατα, προτάθηκαν οι Κατανεμημένοι Πυρήνες Πόρων (Distributed Resource Kernels) (Lakshmanan & Rajkumar, 2008), μια επέκταση του Linux/RK που προσθέτει υποστήριξη για κατανεμημένες εφαρμογές πραγματικού χρόνου. Το Linux/RK στοχεύει προς ένα πρότυπο μη-τροποποίησης των εφαρμογών, καθώς η υποστήριξη πραγματικού χρόνου για κλασσικές εφαρμογές είναι ένα από τα κρίσιμα χαρακτηριστικά γνωρίσματά του. Έτσι, οι εφαρμογές δεν γνωρίζουν απαραιτήτως ότι λαμβάνουν εγγυήσεις από το ΛΣ, αλλά αντίθετα το σύστημα αντιμετωπίζεται και διαμορφώνεται κατά τέτοιο τρόπο ώστε υπηρεσίες χρονοπρογραμματισμού

πραγματικού χρόνου, για τους διάφορους πόρους, ενεργοποιούνται αυτόματα κάθε φορά που προωθείται μια δεδομένη εφαρμογή στο σύστημα. Σημαντικά μειονεκτήματα του Linux/RK είναι ότι, για τη βέλτιστη αποδοτικότητα, ο πυρήνας περιλαμβάνει λειτουργίες για τη διαχείριση και την κατανομή των πόρων με τη βοήθεια αφιερωμένων νημάτων πυρήνα που ανταλλάσσουν τα μηνύματα UDP προκειμένου να επικοινωνήσουν το ένα με το άλλο. Επομένως, ο πυρήνας περιλαμβάνει διάφορα συστατικά που είναι παραδοσιακά παρόντα σε υψηλότερο επίπεδο σε άλλα ΛΣ, όπως έναν κεντρικό εξυπηρετητή HTTP, DNS, και NTP. Αντίθετα η υλοποίηση τέτοιων υπηρεσιών στο χώρο του χρήστη είναι ιδιαίτερα ευεργετική τόσο για λόγους ασφάλειας όσο και ευρωστίας.

Το RTLinuxFree και το RTAI RTLinuxFree είναι η εξέλιξη του RT-Linux, μια τροποποίηση στον πυρήνα του Linux που στοχεύει να υποστηρίξει τις εφαρμογές μη ελαστικού πραγματικού χρόνου. Αρχικά αναπτύχθηκε από τον Victor Yodaiken στο πανεπιστήμιο του New Mexico και κατόπιν από το FSMLabs, το οποίο αποκτήθηκε από την WindRiver το 2007. Λειτουργεί ως μικρός εκτελεστής με έναν χρονοπρογραμματιστή μη ελαστικού πραγματικού χρόνου που, εκτός από διεργασίες πραγματικού χρόνου, εκτελεί ολόκληρο το Linux [τόσο τον πυρήνα όσο και τον χώρο χρήστη (user space)] ως διεργασίες χαμηλής προτεραιότητας (Setz, 2007).

Το RTLinuxFree προσθέτει έναν διαχειριστή πραγματικού χρόνου μεταξύ του υλικού και του πυρήνα του Linux, ο οποίος παίρνει τον άμεσο έλεγχο των διακοπών του υλικού και τις προωθεί στο Linux μόνο όταν δεν είναι διεργασίες πραγματικού χρόνου. Με αυτόν τον τρόπο οι διεργασίες πραγματικού χρόνου τρέχουν ανενόχλητες και χωρίς παρεμβάσεις από ολόκληρο το Linux.



Σχήμα 5.1: *Η αρχιτεκτονική του RT-Linux*

Το RTAI (Διεπαφή Εφαρμογής Πραγματικού Ξρόνου - Real Time Application Interface) είναι μια τροποποίηση του πυρήνα Linux από τον καθ. Paolo Mantegazza του Dipartimento Di Ingegneria Aerospaziale Politecnico του Μιλάνου (DIAPM). Το RTAI είναι ένα πρόγραμμα ανοικτού κώδικα που στηρίζεται μεν στην αρχική ιδέα του RT-Linux, αλλά έχει ενισχυθεί αρκετά. Το RTAI επιτρέπει να αναμιχθούν ομοιόμορφα τεχνικές ελαστικού και μη ελαστικού πραγματικού χρόνου με την ενσωμάτωση του χρονοπρογραμματισμού των διεργασιών του πυρήνα του RTAI, των νημάτων του πυρήνα και των διεργασιών του χώρου χρήστη. Όπως και στο RT-Linux, οι διεργασίες τρέχουν στο υπόβαθρο όσον αφορά τον πυρήνα πραγματικού χρόνου. Το Linux εκτελείται μόνο όταν δεν υπάρχει καμία διεργασία πραγματικού χρόνου για να εκτελεσθεί και ο πυρήνας πραγματικού χρόνου είναι ανενεργός. Επιπλέον, το Linux δεν μπορεί ποτέ να εμποδίσει τις διακοπές ή να αποτρέψει την προεκτόπισή του από τον πυρήνα πραγματικού χρόνου

(Ripoll κ. συν., 2002).

Και οι δύο προσεγγίσεις στοχεύουν να ενσωματώσουν διεργασίες πραγματικού χρόνου και διεργασίες του Linux στο ίδιο σύστημα. Αυτό επιτρέπει, παραδείγματος χάριν, να εκτελεστούν ταυτόχρονα ένα σύνολο από διεργασίες πραγματικού χρόνου που ελέγχουν βιομηχανικές εγκαταστάσεις μαζί με την υψηλού επιπέδου υποδομή και τις εφαρμογές λογισμικού που επιτρέπουν τον έλεγχο των εγκαταστάσεων από μακρινές ή από σύνθετες διεπαφές. Το κύριο μειονέκτημα αυτών των προσεγγίσεων είναι ότι οι διεργασίες πραγματικού χρόνου δεν μπορούν πραγματικά να εκμεταλλευτούν την πλήρη δύναμη του ΛΣ-ΓΧ που τις φιλοξενεί και μπορούν μόνο να έχουν πρόσβαση σε ένα πολύ περιορισμένο σύνολο περιφερειακών μονάδων (π.χ. σειριακές θύρες). Οι σύνθετες εφαρμογές πολυμέσων που επιτρέπουν ροές εικόνας που χρειάζονται τουλάχιστον τη δικτύωση TCP/IP, αυξημένη χρήση του δίσκου και πρόσβαση στους ακουστικούς και τηλεοπτικούς προσαρμοστές δεν θα ήταν κατάλληλες να τρέξουν σε ένα τέτοιο σύστημα.

Μια διαφορετική προσέγγιση ακολουθείται στην παραλλαγή του Linux που αναπτύσσεται ως συνέπεια το ευρωπαϊκού προγράμματος OCERA, το οποίο ενσωματώνει μέσα στο Linux πολιτική χρονοπρογραμματισμού βασισμένη σε δεσμεύσεις (Mercer κ. συν., 1993) χαρακτηριστική των συστημάτων πραγματικού χρόνου. Με αυτό τον τρόπο, τυποποιημένες εφαρμογές Linux, που χρησιμοποιούν το πλήρες σύνολο των διαθέσιμων πόρων μέσω των υπηρεσιών και των βιβλιοθηκών του ΛΣ, μπορούν συγχρόνως να εκμεταλλευτούν τα οφέλη της προβλέψιμης συμπεριφοράς συγχρονισμού και της χρονικής απομόνωσης για εκείνα τα νήματα που είναι περισσότερο εντατικά υπολογιστικά. Στην πραγματικότητα, το OCERA παρείχε μια

προσαρμοσμένη διανομή του πυρήνα του Linux κατάλληλη και για τις εφαρμογές ελαστικού πραγματικού χρόνου και για τις εφαρμογές ανελαστικού πραγματικού χρόνου ενσωματώνοντας: το RT-Linux, ένα σύνολο οδηγών χρήσιμων στα πλαίσια του βιομηχανικού ελέγχου (δηλ., για το διάδρομο CAN) και το Linux με τις επεκτάσεις ελαστικού πραγματικού χρόνου στο επίπεδο χρονοπρογραμματιστών που εμπλουτίζεται με έναν ελεγκτή ποιότητας υπηρεσίας βασισμένο σε ανατροφοδότηση, χρήσιμο για τη συνεχή προσαρμογή των παραμέτρων σχεδιασμού στις παρούσες απαιτήσεις των διεργασιών. Μια ενοποιημένη διεπαφή διαμόρφωσης του πυρήνα επιτρέπει να επιλέγονται τα συστατικά που θα συμπεριληφθούν στον πυρήνα του OCERA.

Μια προσέγγιση παρόμοια με το OCERA έχει υιοθετηθεί επίσης από το Linux/RT, μια εμπορική παραλλαγή του Linux που υποστηρίζεται από την TimeSys Inc., και είναι βασισμένη στον αρχικό πυρήνα του Linux (Linux/RK) από το πανεπιστήμιο Carnegie Mellon (Lakshmanan & Rajkumar, 2008). Στο Linux/RT ο πυρήνας έχει τροποποιηθεί άμεσα έτσι ώστε να παρέχει άμεσα στις διαδικασίες χρηστών δεσμεύσεις ΚΜΕ, δικτύου και δίσκων. Αυτό επιτρέπει την παροχή εγγυήσεων συγχρονισμού στις παραδοσιακές εφαρμογές του Linux με έναν διαφανή τρόπο. Επιπλέον, είναι δυνατό να προσεγγιστεί μία συγκεκριμένη διεπαφή για να γίνει εκμετάλλευση των παρεχόμενων δεσμεύσεων και υπηρεσιών διαχείρισης της ποιότητας της υπηρεσίας.

Ο χρονοπρογραμματιστής του OCERA μέσα στον πυρήνα του Linux έχει ξαναγραφεί σχεδόν εξολοκλήρου στα πλαίσια του ευρωπαϊκού προγράμματος FRESCOR, κυρίως λόγω των αλλαγών του πυρήνα από την σειρά 2.4 στην 2.6, με αποτέλεσμα το πλαίσιο AQuoSA, που περιγράφεται κατωτέρω. Επιπλέον, το πρόγραμμα FRESCOR εστίασε στην ανάπτυξη των αναγκαίων συστατικών για

την υποστήριξη κατανεμημένων εφαρμογών και την υλοποίηση πιό σύνθετων στρατηγικών ελέγχου ποιότητας υπηρεσίας οι οποίες είναι επίσης κατάλληλες για τις κατανεμημένες και ενσωματωμένες εφαρμογές. Στην πραγματικότητα, στο FRESCOR υπάρχουν τμήματα πυρήνων που προφέρουν δεσμεύσεις πόρων στο επίπεδο της ΚΜΕ, των δίσκων και του δικτύου, που μπορούν να προσεγγιστούν μέσω μίας ομοιόμορφης διεπαφής που σχεδιάστηκε γύρω από τις επεκτάσεις πραγματικού χρόνου του POSIX.

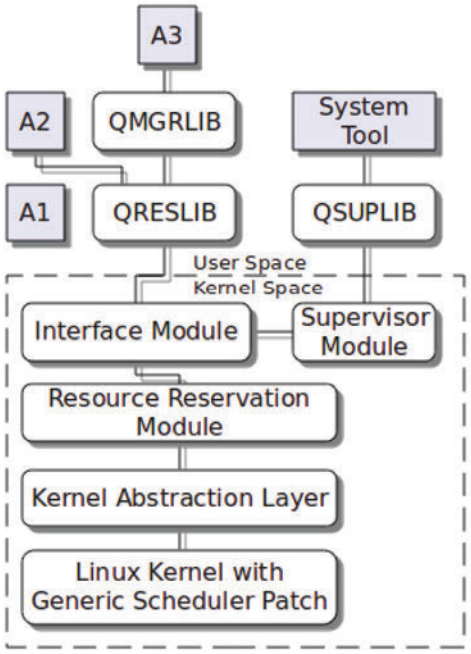
Γενικά, ο κύριος στόχος του προγράμματος FRESCOR είναι να αναπτυχθούν η τεχνολογία και η υποδομή που απαιτούνται για να χρησιμοποιηθούν αποτελεσματικά οι πιο προηγμένες τεχνικές που αναπτύσσονται για τις εφαρμογές πραγματικού χρόνου με εύκαμπτες απαιτήσεις χρονοπρογραμματισμού στις ενσωματωμένες μεθοδολογίες σχεδιασμού συστημάτων και τα εργαλεία εκείνα που επιτρέπουν την στόχευση επαναδιαμορφώσιμων κατανεμημένων αρχιτεκτονικών.

Η προσέγγιση ενσωματώνει προηγμένες εύκαμπτες τεχνικές χρονοπρογραμματισμού σε μια ενσωματωμένη μεθοδολογία σχεδιασμού συστημάτων, που καλύπτει όλα τα επίπεδα που περιλαμβάνονται στην εφαρμογή, από τις αρχές (primitives) του ΛΣ, μέσω του υλικολογισμικού, μέχρι το επίπεδο εφαρμογής. Αυτό επιτυγχάνεται μέσω ενός προτύπου συμβάσεων που διευκρινίζει ποιες είναι οι απαιτήσεις της εφαρμογής όσον αφορά την εύκαμπτη χρήση των πόρων επεξεργασίας στο σύστημα, ποιοι είναι οι πόροι που πρέπει να είναι εγγυημένοι εάν το συστατικό πρόκειται να εγκατασταθεί στο σύστημα και πώς το σύστημα μπορεί να διανείμει οποιαδήποτε εφεδρική ικανότητα που έχει για να επιτύχει την υψηλότερη χρήση των διαθέσιμων πόρων.

Το κύριο μειονέκτημα του FRESCOR είναι η εστίασή του στα ενσωματωμένα συστήματα. Έτσι, η ικανότητα χρήσης πολυεπεξεργαστών στην πλατφόρμα εξετάζεται μόνο περιθωριακά και τα συστατικά του Linux που αναπτύσσονται στο επίπεδο χρονοπρογραμματιστών δεν υποστηρίζουν αυτή τη λειτουργία. Αντ' αυτού, στα πλαίσια του ευρωπαϊκού προγράμματος IRMOS, έχει αναπτυχθεί ένας χρονοπρογραμματιστής πραγματικού χρόνου για Linux με πλήρη υποστήριξη για πολυεπεξεργαστικές και πολυπύρηνες πλατφόρμες (Checconi κ. συν., 2009).

Το πλαίσιο του AQuoSA (`http://aquosa.sourceforge.net`) ενισχύει ένα τυποποιημένο σύστημα GNU/Linux με στρατηγικές χρονοπρογραμματισμού βασισμένες σε τεχνικές δέσμευσης πόρων. Το AQuoSA έχει μια διαστρωματωμένη αρχιτεκτονική όπως απεικονίζεται στο σχήμα 5.2.

Στο χαμηλότερο επίπεδο, υπάρχει ένα μικρό patch (Generic Scheduler Patch, GSP) στον πυρήνα του Linux, που επιτρέπει στις δυναμικά φορτωμένες ενότητες να προσαρμόζουν τη συμπεριφορά του χρονοπρογραμματιστή της ΚΜΕ, με την παρεμπόδιση και αντίδραση σε γεγονότα όπως: η δημιουργία και η καταστροφή tasks, το φράξιμο και η απελευθέρωση tasks σε primitives συγχρονισμού και λήψη των ειδικών σημάτων SIGSTOP και SIGCONT από tasks. Ένα Στρώμα Αφαίρεσης του Πυρήνα (Kernel Abstraction Layer - KAL) στοχεύει στην αφαίρεση των υψηλότερων στρωμάτων από τις πολύ χαμηλού επιπέδου λεπτομέρειες του πυρήνα του Linux (που μπορεί να αλλάξουν από έκδοση σε έκδοση του πυρήνα), με την παροχή ενός συνόλου λειτουργιών και μακροεντολών σε γλώσσα C που αφαιρούν τις αναγκαίες λειτουργίες του πυρήνα. Η ενότητα δέσμευσης των πόρων (Resource Reservation



Σχήμα 5.2: *Η αρχιτεκτονική του AQuoSA*

Module) εφαρμόζει μια παραλλαγή της πολιτικής χρονοπρογραμματισμού CBS πάνω από έναν εσωτερικό χρονοπρογραμματιστή EDF. Η ενότητα διεπαφών Interface Module επιτρέπει σε εφαρμογές που τρέχουν στον χώρο χρήστη (user-space) να ζητήσουν δεσμεύσεις της ΚΜΕ, αφήνοντας όλα τα αιτήματα να περάσουν από την ενότητα εποπτών (Supervisor Module). Η τελευταία εφαρμόζει ένα κατάλληλα σχεδιασμένο πρότυπο ελέγχου πρόσβασης (Cucinotta, 2008), με τη βοήθεια του οποίου το AQuoSA είναι διαθέσιμο στους μη εξουσιοδοτημένους χρήστες υπό μια πολιτική ασφάλειας που μπορεί να διαμορφωθεί από το διαχειριστή του συστήματος. Οι περισσότερες από τις άλλες επεκτάσεις πραγματικού χρόνου του Linux, αντ' αυτού, επιτρέπουν μόνο στους προνομιούχους χρήστες να εκμεταλλευτούν τη διαθέσιμη λειτουργία πραγματικού χρόνου.

Μια βιβλιοθήκη στον χώρο χρήστη (user-space) επιτρέπει στις εφαρμογές να εκμεταλλευτούν τη λειτουργία του AQuoSA. Εκτός από τη βιβλιοθήκη δέσμευσης των πόρων (QRESLIB), που χρησιμοποιείται από μη προνομιούχες εφαρμογές για να δεσμευθεί η ΚΜΕ, μια βιβλιοθήκη εποπτών (QSUPLIB) χρησιμοποιείται από ένα εργαλείο του συστήματος προκειμένου να παρασχεθεί στον πυρήνα η πολιτική ελέγχου πρόσβασης που διαμορφώνεται από το διαχειριστή του συστήματος. Τέλος, μια βιβλιοθήκη διαχείρισης Ποιότητας Υπηρεσίας (QMGRLIB) είναι διαθέσιμη η οποία υλοποιεί προσαρμοστικές δεσμεύσεις πόρων και είναι ιδιαιτέρως χρήσιμη για την ανάπτυξη περιοδικών εφαρμογών.

Ένα ενδιαφέρον χαρακτηριστικό γνώρισμα της αρχιτεκτονικής AQuoSA είναι ότι δεν αντικαθιστά τον προεπιλεγμένο χρονοπρογραμματιστή του Linux, αλλά συνυπάρχει με αυτόν, δίνοντας στους στόχους ελαστικού πραγματικού χρόνου πιο υψηλή προτεραιότητα από οποιοδήποτε στόχο του Linux. Επιπλέον, η αρχιτεκτονική AQuoSA ακολουθεί μια μη-παρεισφρητική προσέγγιση διατηρώντας στο ελάχιστο τις τροποποιήσεις που απαιτούνται στον πυρήνα του Linux. Στην τρέχουσα έκδοση το AQuoSA υποστηρίζει μόνο συστήματα ενός επεξεργαστή.

Το Solaris 10 διαθέτει ικανότητες πραγματικού χρόνου (McDougall & Mauro, 2006). Τα κύρια χαρακτηριστικά του περιλαμβάνουν:

- Πλήρως προεκτοπίσιμος πυρήνας: Εάν μια διαδικασία πραγματικού χρόνου γίνει εκτελέσιμη, θα τοποθετηθεί αμέσως σε μια ΚΜΕ εάν η προτεραιότητά της είναι υψηλότερη από το νήμα που τρέχει σε εκείνη την ΚΜΕ.

- Διακοπές ως νήματα: Οι διακοπές μετατρέπονται σε νήματα με τις δικές τους δομές δεδομένων. Οι διακοπές μπορούν να μπλοκαριστούν σε primitives συγχρονισμού του πυρήνα. Αυτό επιτρέπει να προστατευθούν οι δομές δεδομένων στον πυρήνα του Solaris με primitives συγχρονισμού αντί να αυξάνεται ή να μειώνεται το επίπεδο προτεραιότητας των διακοπών.

- Χρονοπρογραμματισμός πραγματικού χρόνου: Οι οντότητες που χρειάζονται χρονοπρογραμματισμό πραγματικού χρόνου μπορούν να χρησιμοποιήσουν τον χρονοπρογραμματιστή που προσφέρει δύο επιλογές: round-robin ή FIFO.

- Primitives συγχρονισμού με κληρονομική προτεραιότητα: Το βασικό primitive συγχρονισμού στο Solaris είναι το mutex. Τα mutexes είναι προσαρμοστικά καθώς εάν η οντότητα που κατέχει το mutex δεν τρέχει αυτήν την περίοδο τότε η οντότητα που προσπαθεί να το κερδίσει τίθεται σε ύπνο δεδομένου ότι δεν υπάρχει καμία πιθανότητα να το αναλάβει ενώ ο τρέχων ιδιοκτήτης κοιμάται. Τα mutexes (και άλλοι primitives) προσφέρουν τη δυνατότητα της «αντιστροφής προτεραιότητας». Τα mutexes στο Solaris εφαρμόζουν το βασικό πρωτόκολλο κληρονομικής προτεραιότητας. Όταν η οντότητα υψηλού επιπέδου μπλοκαριστεί, όλες οι οντότητες που την εμποδίζουν αποκτούν την προτεραιότητα της οντότητας υψηλού επιπέδου. Όταν παύουν να εμποδίζουν το νήμα, οι προτεραιότητές τους επανέρχονται στο προηγούμενο επίπεδό τους.

- Συμμόρφωση με το POSIX: Το Solaris 10 έχει σχεδόν πλήρη υποστήριξη για το POSIX 1003.1b και πλήρη υποστήριξη για το POSIX 1003.1c.

5.4 Ο Χρονοπρογραμματιστής Πραγματικού Χρόνου του IRMOS

Ο χρονοπρογραμματιστής πραγματικού χρόνου που αναπτύχθηκε στα πλαίσια του ευρωπαϊκού προγράμματος IRMOS επιτρέπει την δέσμευση μιας «φέτας» (slice) της ικανότητας επεξεργασίας ενός συστήματος από μια ομάδα νημάτων ή/και διαδικασιών. Αυτό επιτυγχάνεται με τον ορισμό δύο παραμέτρων χρονοπρογραμματισμού για κάθε ομάδα: ένας προϋπολογισμός Q και μια περίοδος P, με την έννοια ότι οι στόχοι στην ομάδα έχουν δικαίωμα να τρέξουν σε κάθε μια από τις ΚΜΕ (επεξεργαστής ή πυρήνας) που είναι διαθέσιμες στο ΛΣ, για Q χρονικές μονάδες κάθε περίοδο P χρονικών μονάδων. Αυτό αποτελεί συγχρόνως μια εγγύηση χρονοπρογραμματισμού και έναν περιορισμό. Αυτό επιτυγχάνεται από μια παραλλαγή του αλγορίθμου EDF, τον χρονοπρογραμματιστή Κεντρικού Υπολογιστή Σταθερού Εύρους Ζώνης (Constant Bandwidth Server - CBS) (Abeni & Buttazzo, 1998), όπου κάθε ΚΜΕ έχει μια ιδιωτική ουρά αναμονής στόχων η οποία χρονοπρογραμματίζεται ανεξάρτητα. Εντούτοις, όταν μια ομάδα έχει το δικαίωμα να τρέξει σε κάθε ΚΜΕ, ο χρονοπρογραμματιστής του IRMOS υιοθετεί μια στρατηγική χρονοπρογραμματισμού πραγματικού χρόνου μεταξύ των στόχων του που βασίζεται σε προτεραιότητες κατά το POSIX ("IEEE Standard for Information Technology- Standardized Application Environment Profile (AEP)-POSIX Realtime and Embedded Application Support", 2004), κατά τέτοιο τρόπο ώστε, εάν υπάρχουν m ΚΜΕ, τότε m στόχοι με την πιο υψηλή προτεραιότητα είναι αυτοί που τρέχουν πραγματικά. Το σύστημα

εκτελεί έλεγχο αποδοχής, έτσι ώστε η γενική χωρητικότητα του συστήματος να μπορεί να χωριστεί κατάλληλα μεταξύ δραστηριοτήτων που εκτελούνται ταυτόχρονα στο σύστημα, χωρίς την υπερφόρτωσή του.

Επίσης, ο χρονοπρογραμματιστής έχει ικανότητα ιεραρχικής διαμόρφωσης, μέσω της οποίας είναι δυνατό να καθοριστούν ομάδες και ένθετες υποομάδες στόχων πραγματικού χρόνου με συγκεκριμένες παραμέτρους χρονοπρογραμματισμού.

5.5 Σύνοψη του Κεφαλαίου

Σε αυτό το κεφάλαιο έγινε μία επισκόπηση για την υποστήριξη πραγματικού χρόνου που παρέχουν τα ΛΣ-ΓΧ, με μια ιδιαίτερη έμφαση στο Linux. Είναι εμφανές ότι υπάρχει ένα αυξανόμενο ενδιαφέρον για την ενσωμάτωση υποστήριξης πραγματικού χρόνου στο επίπεδο του πυρήνα στα ΛΣ-ΓΧ, όπως το Linux. Το Linux αποτελεί άριστη πλατφόρμα ανάπτυξης για μια ευρεία ποικιλία εφαρμογών ελαστικού πραγματικού χρόνου, όπως οι διαδραστικές εφαρμογές πολυμέσων, χάρη στην ευρεία διαθεσιμότητα βιβλιοθηκών συμπίεσης και πολυμέσων και την υποστήριξη για σωρεία συσκευών πολυμέσων. Εξετάζοντας την τάση ανάπτυξης του πυρήνα του Linux, είναι εμφανές πώς στα τελευταία έτη, παρά τις αυξανόμενες υπολογιστικές ικανότητες του υλικού, ζητήματα όπως οι καθυστερήσεις λόγω διακοπών, η δυνατότητα για προεκτοπίσιμα (preemtable) τμήματα κώδικα, η ακρίβεια των χρονομέτρων και οι πολιτικές χρονοπρογραμματισμού κερδίζουν όλο και περισσότερο ενδιαφέρον στην κοινότητα των χρηστών και των υπεύθυνων για την ανάπτυξη λογισμικού. Ο τρέχων χρονοπρογραμματιστής

του Linux για τις διαδικασίες μη πραγματικού χρόνου ενσωματώνει ήδη έννοιες που προέρχονται από τον κόσμο του χρονοπρογραμματισμού πραγματικού χρόνου, ακόμα κι αν όχι με έναν απολύτως αποτελεσματικό τρόπο για τις εφαρμογές πραγματικού χρόνου.

Αναμένεται ότι στα επόμενα έτη κάθε ΛΣ-ΓΧ θα ενσωματώσει πιο πολλές δυνατότητες χρονοπρογραμματισμού πραγματικού χρόνου, προκειμένου να καλυφθούν καλύτερα οι αυξανόμενες απαιτήσεις των χρονικά ευαίσθητων εφαρμογών. Έτσι, παρότι η προτεινόμενη λύση για το ΣΔΡΕ βασίζεται στις επεκτάσεις που υλοποιήθηκαν στα πλαίσια του ερευνητικού έργου IRMOS, αυτό δεν είναι δεσμευτικό.

6 Προτεινόμενο ΣΔΡΕ

6.1 Σχεδιαστικές Αποφάσεις

Κατά την σχεδίαση του συστήματος ελήφθησαν κάποιες αποφάσεις με γνώμονα την ικανοποίηση των απαιτήσεων που περιγράφηκαν στο κεφάλαιο 3.

Καταρχάς, το σύστημα ακολουθεί υπηρεσιοστρεφή αρχιτεκτονική και υλοποιεί το Μοντέλο Υπηρεσιών Νέφους όπως φαίνεται και στο Σχ. 6.1. Κάθε εφαρμογή αποτελείται από διαφορετικές υπηρεσίες οι οποίες δημιουργούν ένα Εικονικοποιημένο Δίκτυο Υπηρεσιών (ΕΔΥ).

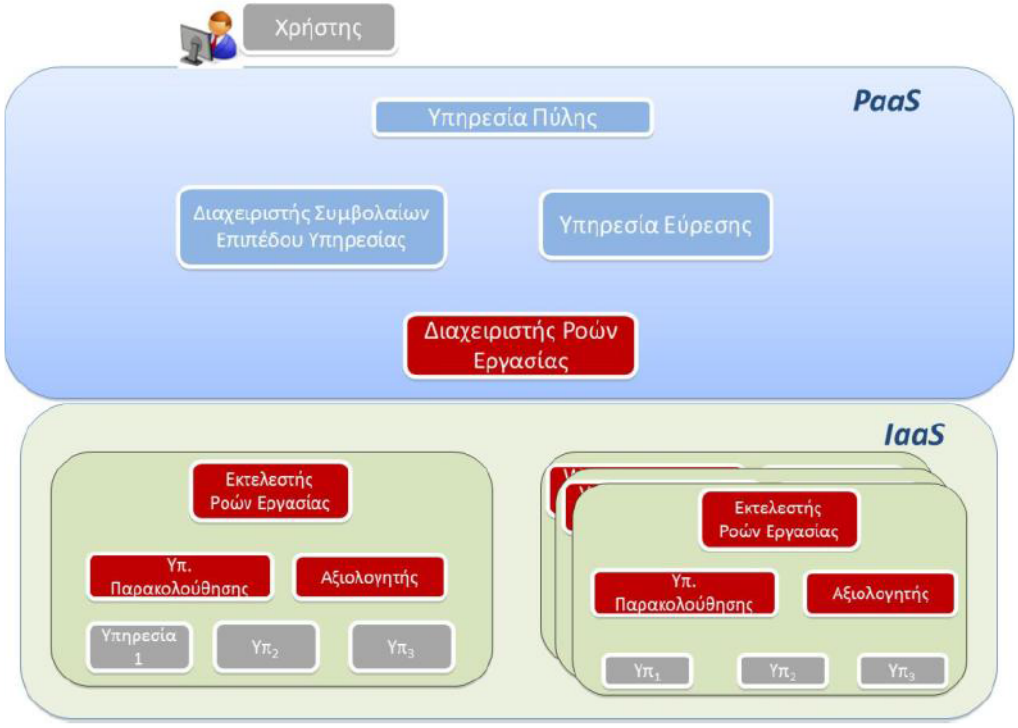
Για να επιτευχθεί η απαίτηση της επεκτασιμότητας, το προτεινόμενο ΣΔΡΕ έχει πολυεπίπεδη δομή. Ένα μέρος του μηχανισμού διαχείρισης ροών εργασίας μοντελοποιείται όπως όλες οι υπηρεσίες που αποτελούν την εφαρμογή και φιλοξενείται εντός ενός Εικονικοποιημένου Περιβάλλοντος - ενός Νέφους - του οποίου οι πόροι (υπολογιστικοί, δικτυακοί και αποθήκευτικοί) δύνανται να αυξηθούν και να μειωθούν κατά το δοκούν. Επομένως, το ΣΔΡΕ έχει την ίδια μεταχείριση, σχετικά με την διαχείρισή του από το Νέφος, όπως κάθε

άλλη υπηρεσία. Έτσι οι απαιτήσεις σε πόρους της ίδιας της πλατφόρμας δεν είναι στατικές, αλλά υπολογίζονται διαρκώς και είναι δυνατόν να αλλάξουν όταν υπάρχει ανάγκη. Αυτό όχι μόνο βοηθάει ώστε να καλύπτονται οι απαιτήσεις ποιότητας υπηρεσίας του τελικού χρήστη, αλλά οδηγεί και σε καλύτερη διαχείριση της υποδομής από τον πάροχο.

Οι απαίτηση για διαδραστικότητα καλύπτεται μέσω μίας διεπαφής την οποία μπορεί να χρησιμοποιήσει ο χρήστης για να σταματήσει, να αναστείλει ή να επαναδιαμορφώσει μία ροή εργασίας που εκτελείται, αρκεί αυτό να μην αντιβαίνει του υπογραφέντος Συμβολαίου Επιπέδου Υπηρεσίας. Η ανοχή στα σφάλματα υλοποιείται μέσω κανόνων που μπορεί να ορίσει ο χρήστης, οι οποίοι αξιολογούνται κατά την εκτέλεση.

Η απαίτηση για προσδιορισμό απαιτήσεων σε φιλικούς προς τον χρήστη όρους ικανοποιείται μέσω ενός μηχανισμού που αναλαμβάνει να μεταφράσει παραμέτρους υψηλού επιπέδου σε απαίτησης πόρων χαμηλού επιπέδου.

Η ασφάλεια του συστήματος είναι πολύ σημαντική για να μπορεί να υιοθετηθεί από εμπορικές εφαρμογές. Για τον λόγο αυτό, κάθε χρήστης χρησιμοποιεί ψηφιακά πιστοποιητικά για να επικοινωνήσει με την πλατφόρμα. Επίσης ψηφιακά πιστοποιητικά χρησιμοποιούνται και μεταξύ των διαφορετικών υπηρεσιών της πλατφόρμας, ώστε να εξασφαλίζεται η ασφάλεια των επικοινωνιών. Τα πιστοποιητικά αυτά εκδίδονται από τον Διαχειριστή Ροών Εργασίας, που ενεργεί ως υπηρεσία πιστοποίησης. Επίσης η γλώσσα που χρησιμοποιείται για τον ορισμό μίας ροής εργασίας περιέχει διαφορετικές δομές που μπορούν να χρησιμοποιηθούν για να ορίσουν το επιθυμητό επίπεδο ασφάλειας. Έτσι μπορεί να ορισθεί ασφάλεια επιπέδου μηνύματος ή επιπέδου μεταφοράς. Επίσης όλες οι υπηρεσίες μίας εφαρμογής



Σχήμα 6.1: *Αρχιτεκτονική του προτεινόμενου συστήματος*

χρησιμοποιούν προσωπικές διευθύνσεις IP δημιουργώντας ένα intranet, ενώ εξωτερικές IP χρησιμοποιούνται μόνο όπου ζητηθεί από τον χρήστη. Έτσι επιτυγχάνεται απομόνωση σε επίπεδο δικτύου (network isolation).

6.2 Αρχιτεκτονική του Συστήματος

Η προτεινόμενη αρχιτεκτονική μίας πλατφόρμας ικανής να παρέχει τις εγγυήσεις που χρειάζεται ώστε να μπορούν να εξυπηρετηθούν εφαρμογές ελαστικού πραγματικού χρόνου φαίνεται στο Σχήμα 6.1.

Η αρχιτεκτονική διαχωρίζεται στα επίπεδα των PaaS και IaaS. Στο επίπεδο PaaS υπάρχουν υπηρεσίες με τις οποίες αλληλεπιδρά ο χρήστης

και έχουν ως σκοπό την διαχείριση όλης της πλατφόρμας. Μία εφαρμογή αποτελείται από πολλές υπηρεσίες ή Application Service Components - ASCs οι οποίες μαζί με υπηρεσίες της ίδιας της πλατφόρμας δημιουργούν ένα Εικονικό Δίκτυο Υπηρεσιών (ΕΔΥ - Virtual Service Network - VSN). Κάθε ΕΔΥ τρέχει μέσα στο εικονικοποιημένο περιβάλλον που παρέχει ο πάροχος IaaS, ενώ πολλαπλά ΕΔΥ μπορούν να εκτελούνται ταυτόχρονα, μοιραζόμενα τους φυσικούς πόρους. Μετά από ανάλυση για κάθε υπηρεσία ξεχωριστά, εξάγονται οι απαιτήσεις που έχει σε φυσικούς πόρους με βάση συγκεκριμένα στοιχεία εισόδου. Έτσι είναι δυνατόν να γίνει η σωστή απαίτηση και δέσμευση των πόρων από τον πάροχο IaaS, ώστε να καλύπτονται οι ανάγκες της εφαρμογής στην ολότητά τους και να επιτρέπεται η παροχή ισχυρών δεσμεύσεων επιπέδου υπηρεσίας από τον πάροχο της πλατφόρμας.

Οι απαιτήσεις της εφαρμογής προσδιορίζονται από τον τελικό χρήστη σε υψηλό επίπεδο, δηλαδή με όρους που αυτός κατανοεί όπως ο αριθμός ταυτόχρονων χρηστών, ο χρόνος εκτέλεσης ή ο αριθμός καρέ ανά δευτερόλεπτο. Οι απαιτήσεις αυτές πρέπει να μεταφραστούν σε απαιτήσεις χαμηλού επιπέδου, όπως είναι η απαιτούμενη μνήμη RAM, ο χρονισμός του επεξεργαστή κλπ. Για να επιτευχθεί αυτή η μετάφραση χρησιμοποιούνται Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ). Έτσι, πριν το deployment μίας εφαρμογής υπάρχει μία φάση αξιολόγησης (benchmarking) κατά την οποία συλλέγεται ένα ικανοποιητικό σύνολο δεδομένων το οποίο χρησιμοποιείται προκειμένου να εκπαιδευθούν τα νευρωνικά δίκτυα. Για κάθε ASC δημιουργείται ένα ΤΝΔ και στη συνέχεια ένα αριθμητικό λογισμικό όπως το GNU Octave ή το Matlab μπορεί να χρησιμοποιηθεί άμεσα στον κύκλο ζωής των υπηρεσιών. Τα δημιουργημένα μοντέλα αποθηκεύονται για πρόσβαση από όλες τις εμπλεκόμενες υπηρεσίες.

Μετά από αυτήν την φάση, για κάθε ένα από τα ASC λαμβάνεται ένας αλγεβρικός κανόνας, υπό μορφή ΤΝΔ, που είναι συγκεκριμένος για αυτό το ASC και μπορεί να απεικονίσει την επίδραση μιας επιλογής ενός συγκεκριμένου πόρου υλικού και μιας συγκεκριμένης διαμόρφωσης για το ASC στην απόδοση, όπως αυτή εκφράζεται μέσω των συγκεκριμένων δεικτών απόδοσης. Ο λόγος για τη χρήση αυτής της τεχνικής (ΤΝΔ) είναι ότι ικανοποιεί πολλές από τις απαιτήσεις που εμφανίζονται στις Υπηρεσιοστραφείς Αρχιτεκτονικές. Παραδείγματος χάριν, απαιτείται πολύ λίγη γνώση για την υποδομή ή το ASC. Αυτές οι πληροφορίες περιλαμβάνονται στο (ASC) και είναι κυρίως οι προαναφερθείσες απαιτήσεις της εφαρμογής που υπάρχουν ήδη στο SLA. Επιπλέον, οι παράμετροι εισαγωγής και τα αποτελέσματα των προτύπων μπορούν εύκολα να ρυθμιστούν εκ νέου προκειμένου να περιληφθούν νέες παράμετροι.

Επίσης για τον απαιτούμενο χρονοπρογραμματισμό χρησιμοποιείται ο χρονοπρογραμματιστής (Checconi κ. συν., 2009) που τρέχει στους φυσικούς υπολογιστικούς πόρους του συστήματος. Ο χρονοπρογραμματιστής αυτός επιτρέπει τον ορισμό κρατήσεων με την μορφή:

$$RSV = (Q, P)$$

που σημαίνει ότι ο επεξεργαστής θα διοριστεί σε μία συγκεκριμένη Εικονική Μηχανή για Q χρονικές μονάδες μέσα σε κάθε διάστημα P χρονικών μονάδων (όπου $Q \leq P$). Αυτό επιτρέπει έναν λεπτομερή χειρισμό των υπολογιστικών πόρων που χορηγούνται σε κάθε εικονική μηχανή παρέχοντας ταυτόχρονα χρονική απομόνωση.

6.3 Τα Βασικά Μέρη του ΣΔΡΕ

Το προτεινόμενο ΣΔΡΕ ακολουθεί μία πολυεπίπεδη αρχιτεκτονική και αποτελείται από τρία κυρίως μέρη:

Διαχειριστής Ροών Εργασίας (Workflow Manager). Ο Διαχειριστής Ροών εργασίας είναι η κεντρική υπηρεσία που είναι υπεύθυνη για όλες τις ροές που εκτελούνται από την πλατφόρμα και επιβλέπει πολλαπλούς Εκτελεστές Ροών Εργασίας. Επίσης αποτελεί την υπηρεσία από την οποία διέρχονται όλες οι εντολές του χρήστη πριν μεταφερθούν στον υπεύθυνο Εκτελεστή Ροών Εργασίας.

Εκτελεστής Ροών Εργασίας (Workflow Enactor). Είναι η υπηρεσία που είναι υπεύθυνη να εκτελέσει μία ροή εργασίας βάσει του ορισμού της. Είναι επίσης υπεύθυνη για να λαμβάνει μηνύματα από τον Αξιολογητή και να αντιδρά σε αυτά. Ένας Εκτελεστής είναι υπεύθυνος για κάθε εφαρμογή-ροή εργασίας και θεωρείται μέρος αυτής κατά την εκτίμηση των απαιτούμενων πόρων. Ένας Εκτελεστής εκτελεί αποκλειστικά μία ροή εργασίας και εδράζεται μέσα στο εικονικοποιημένο περιβάλλον καθώς αποτελεί μέρος του ΕΔΥ.

Αξιολογητής (Evaluator). Ο Αξιολογητής είναι υπεύθυνος να ελέγχει διαρκώς τις παραμέτρους που έχουν τεθεί από τον χρήστη στο Συμβόλαιο Επιπέδου Υπηρεσίας και να παράγει μηνύματα που προωθούνται προς τον Εκτελεστή, βάσει κανόνων. Παραδείγματος χάριν, μπορεί να παράγει ένα μήνυμα προς τον Αξιολογητή όταν μία παράμετρος είναι κάτω από ένα κατώφλι, ώστε να πυροδοτήσει την έναρξη μίας αλληλουχίας ενεργειών από τον Εκτελεστή.

6.4 Τα Βασικά Μέρη της Πλατφόρμας

Οι βασικές υπηρεσίες της πλατφόρμας πέραν του ΣΔΡΕ είναι οι εξής:

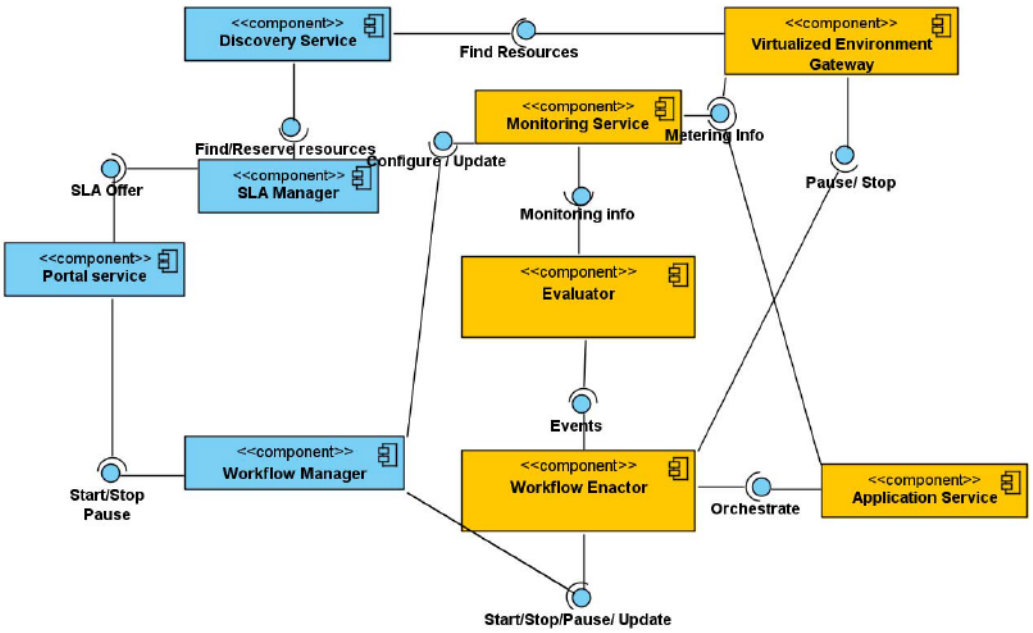
Υπηρεσία Παρακολούθησης (Monitoring). Η υπηρεσία αυτή είναι υπεύθυνη να συλλέγει όλες τις απαιτούμενες πληροφορίες που χρειάζονται από το σύστημα. Μία υπηρεσία δημιουργείται ανά εφαρμογή και αποτελεί μέρος του δημιουργούμενου δικτύου υπηρεσιών.

Διαχειριστής Συμβολαίων Επιπέδου Υπηρεσίας (SLA Manager). Η υπηρεσία είναι υπεύθυνη να υπογράφει Συμβόλαια Επιπέδου Υπηρεσίας με τους τελικούς χρήστες.

Υπηρεσία Εύρεσης (Discovery Service). Η υπηρεσία είναι υπεύθυνη να βρίσκει πόρους κατάλληλους να φιλοξενήσουν μία εφαρμογή. Καλείται από τον Διαχειριστή Συμβολαίων, στον οποίον και επιστρέφει τα αποτελέσματα της αναζήτησης.

Υπηρεσία Πύλης (Portal Service). Η υπηρεσία πύλης αποτελεί την διεπαφή του χρήστη με την πλατφόρμα. Μέσω αυτής μπορούν να δοθούν εντολές για να ξεκινήσει, να σταματήσει ή να παύσει μία ροή εργασίας.

Τα δομικά στοιχεία της πλατφόρμας φαίνονται στο Σχήμα 6.2. Τα στοιχεία με Monitoring Service, Ebaluator, Workflow Enactor, Virtualized Service Gateway και Application Service βρίσκονται εντός του εικονικοποιημένου περιβάλλοντος.



Σχήμα 6.2: *Διάγραμμα δομικών στοιχείων της πλατφόρμας*

6.5 Φάσεις Εκτέλεσης μίας Εφαρμογής στην Πλατφόρμα IRMOS

Η διαδικασία για να μεταφερθεί μία εφαρμογή στη πλατφόρμα IRMOS μπορεί να διαιρεθεί σε τρεις ευδιάκριτες φάσεις. Παρότι το προτεινόμενο ΣΔΡΕ δεν συμμετέχει ενεργά σε όλες τις φάσεις, εντούτοις κρίνεται σκόπιμο να παρουσιαστούν συνοπτικά για λόγους πληρότητας, καθώς πολλές από τις ενέργειες που πραγματοποιούνται κατά τις φάσεις αυτές επιδρούν στον τρόπο λειτουργίας του ΣΔΡΕ.

6.5.1 Φάση Δημοσίευσης.

Κατά τη φάση δημοσίευσης ο υπεύθυνος για την ανάπτυξη της εφαρμογής μοντελοποιεί την εφαρμογή ώστε να μπορεί να αναπτυχθεί σε μία υπηρεσιοστρεφή πλατφόρμα. Κάθε εφαρμογή είναι μία ροή εργασίας, αποτελούμενη από πολλαπλές υπηρεσίες, κάθε μία από τις οποίες παρέχει κάποια διακριτή λειτουργία και ονομάζεται Application Service Component - ASC. Ο υπεύθυνος ανάπτυξης της εφαρμογής αλληλεπιδρά με μία διεπαφή ανάπτυξης που επιτρέπει τον καθορισμό των διεπαφών εισόδου και εξόδου του ASC καθώς επίσης και των απαιτούμενων υπολογιστικών και δικτυακών πόρων. Οι απαιτήσεις σε πόρους ενδέχεται να εξαρτώνται από τις παραμέτρους των δεδομένων εισόδου/εξόδου και να περιέχουν και χρονικούς περιορισμούς. Για την μοντελοποίηση και περιγραφή των ASC χρησιμοποιείται το UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) (*UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems*, 2009) καθώς και το UML for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (*UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification v1.1*, 2008). Ο υπεύθυνος για την ανάπτυξη της εφαρμογής μπορεί στη συνέχεια να περιγράψει τη ροή εργασίας, τις απαιτήσεις ποιότητας υπηρεσίας που επιθυμεί για την εφαρμογή καθώς και κανόνες που χρησιμοποιούνται από τον Αξιολογητή για την παραγωγή γεγονότων. Μετά τη μοντελοποίηση ο πάροχος της πλατφόρμας είναι σε θέση να αξιολογήσει την εφαρμογή και να μετατρέψει τις απαιτήσεις της εφαρμογής σε απαιτήσεις πόρων. Κατά την διάρκεια της μοντελοποίησης ο Εκτελεστής Ροών Εργασίας,

η Υπηρεσία Παρακολούθησης και ο Αξιολογητής θεωρούνται μέρη της εφαρμογής ώστε να εξαχθούν οι απαιτήσεις σε πόρους και να παραχθούν οι αντίστοιχες εγγυήσεις ποιότητας υπηρεσίας από τον προμηθευτή IaaS.

6.5.2 Φάση Διαπραγμάτευσης.

Κατά τη φάση διαπραγμάτευσης ο χρήστης της εφαρμογής δημιουργεί ένα αίτημα προς τον πάροχο PaaS το οποίο περιέχει τη ροή εργασίας καθώς και απαιτήσεις υψηλού επιπέδου χρησιμοποιώντας πρότυπα που έχουν δημιουργηθεί από τον υπεύθυνο ανάπτυξης της εφαρμογής. Το κείμενο που δημιουργεί ο χρήστης περιέχει επίσης κανόνες που χρησιμοποιούνται από την υπηρεσία Αξιολογητή για να παράγει γεγονότα. Το αίτημα του χρήστη μετασχηματίζεται αυτόματα από τον πάροχο σε ένα αίτημα προς τον πάροχο IaaS που περιέχει απαιτήσεις πόρων χαμηλού επιπέδου και το οποίο εκφράζεται μέσω ενός κειμένου που καλείται Περιγραφή Εικονικού Δικτύου Υπηρεσιών (Virtual Service Network Description - VSND). Αν το αίτημα του χρήστη μπορεί να ικανοποιηθεί το κόστος επιστρέφεται στον χρήστη. Αν ο χρήστης αποδεχθεί το κόστος οι πόροι δεσμεύονται για το χρονικό πλαίσιο που περιγράφεται στο αίτημα. Οι δεσμεύσεις για την παροχή των υπηρεσιών περιγράφονται σε SLA τόσο μεταξύ του χρήστη και του παρόχου PaaS (SLA εφαρμογής - Application SLA - A-SLA) όσο και μεταξύ του παρόχου PaaS και του παρόχου IaaS (τεχνικό SLA - Technical SLA - T-SLA).

6.5.3 Φάση Εκτέλεσης.

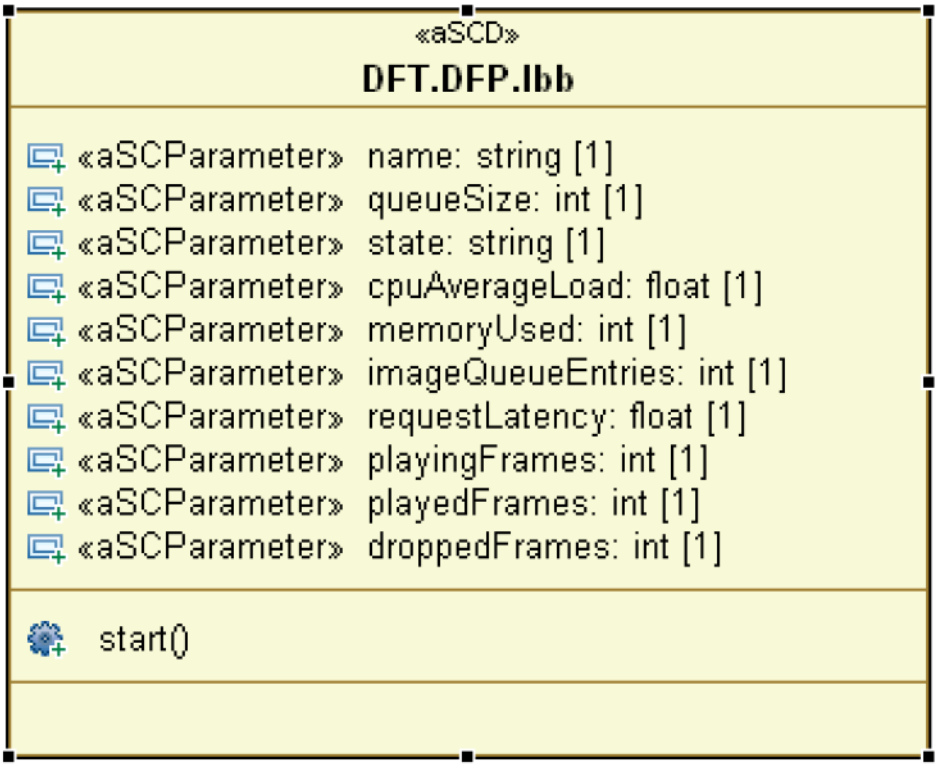
Η φάση εκτέλεσης ξεκινάει όταν ο χρήστης εισέλθει στην πλατφόρμα και ζητήσει την εκτέλεση της εφαρμογής. Όλα τα μέρη της εφαρμογής έχουν ήδη αρχικοποιηθεί ως εικονικές μηχανές (Virtual Machine Unit - VMU) με τις παραμέτρους που έχει θέσει ο χρήστης. Ο χρήστης δύναται πλέον να χρησιμοποιήσει την εφαρμογή για το χρονικό διάστημα που έχει ζητήσει την δέσμευση των πόρων. Η φάση εκτέλεσης περιγράφεται διεξοδικά στην παράγραφο 6.8.

6.6 Μοντελοποίηση της Εφαρμογής

Για την μοντελοποίηση της εφαρμογής απαιτείται πρώτα η μοντελοποίηση των ASC που αποτελούν την εφαρμογή. Η περιγραφή τους γίνεται μέσω ενός ASCD το οποίο περιέχει όλη την πληροφορία που χρειάζεται η πλατφόρμα για χειριστεί την εφαρμογή, όπως πληροφορίες για τη διαμόρφωσή της, πληροφορίες για τις ανάγκες της εφαρμογής κ.α. Παράδειγμα ενός ASCD φαίνεται στο Σχήμα 6.3.

Ο δημιουργός του ASC θα πρέπει να καθορίσει όλα τα χαρακτηριστικά των ASC όπως επίσης και τον ρόλο τους, αν δέχονται δεδομένα εισόδου ή παράγουν δεδομένα εξόδου καθώς και ποιες παράμετροι μπορούν να αλλαχθούν. Επίσης είναι δυνατόν να καθορισθεί και ο τύπος και το εύρος των επιτρεπτών τιμών.

Αφού έχουν ορισθεί οι περιγραφές των ASC είναι δυνατόν να ορισθούν στιγμιότυπα χρησιμοποιώντας διαγράμματα κλάσεων UML. Στο Σχήμα 6.4



Σχήμα 6.3: *Περιγραφή ενός ASC*

απεικονίζεται ένα στιγμιότυπο του προαναφερθέντος DFT.DFP.lbb ASC. Το στιγμιότυπο παρέχει συγκεκριμένες τιμές για τις ιδιότητες name και queueSize.

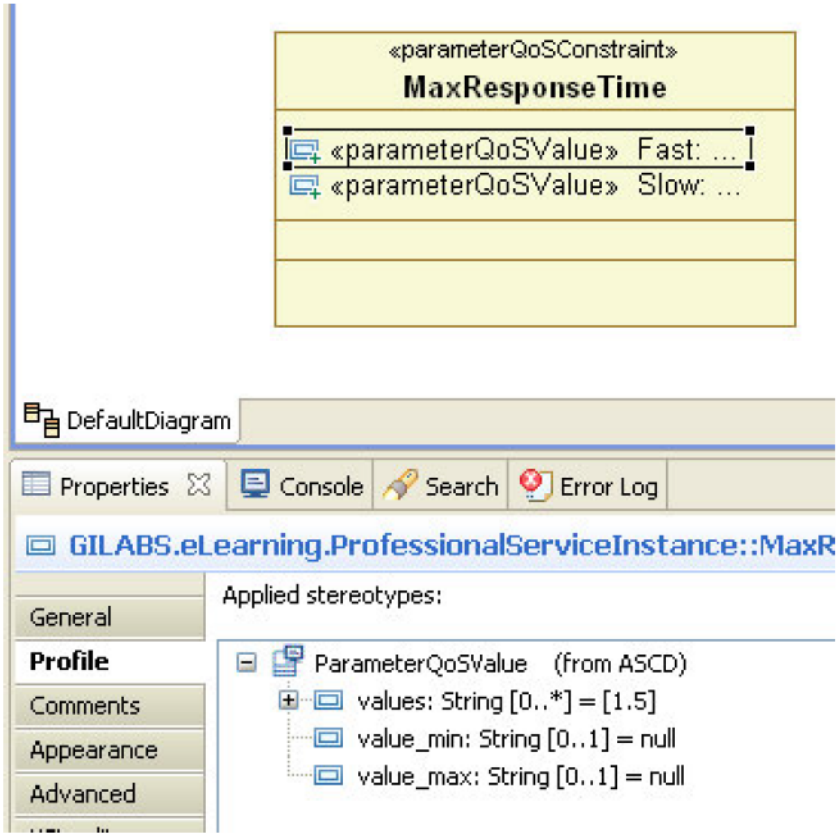
Σχήμα 6.4: *Στιγμιότυπο ενός ASC*

Στα στιγμιότυπα των ASC είναι δυνατόν να ορισθούν περιορισμοί QoS για τις διάφορες παραμέτρους. Στο Σχήμα 6.5 φαίνεται πως μπορεί να ορισθεί ότι η παράμετρος *MaxResponseTime* μπορεί να πάρει μόνο δύο τιμές *Fast* και *Slow* όπου η τιμή *Fast* ορίζεται στα 1, 5 δευτερόλεπτα.

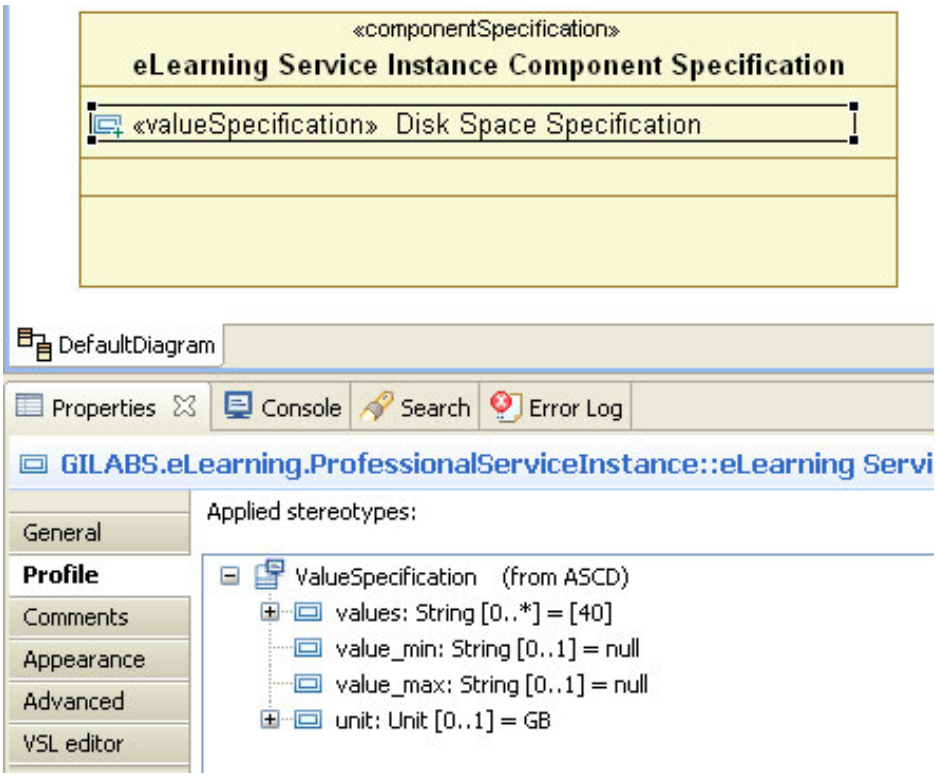
Επίσης είναι δυνατόν να ορισθούν περιορισμοί για τις ανάγκες πόρων που έχει μία υπηρεσία. Στο Σχήμα 6.6 φαίνεται ο ορισμός μίας απαίτησης για μέγεθος αποθηκευτικού χώρου ίσο με 40 GB.

Η περιγραφή μίας εφαρμογής γίνεται μέσω της σύνδεσης πολλών ASCD. Για το μοντέλο αυτό χρησιμοποιούμε το UML διάγραμμα σύνθετης δομής (composite structure) για να μοντελοποιήσουμε πως διασυνδέονται τα διαφορετικά ASC. Το Σχήμα 6.7 δείχνει μία απλή εφαρμογή διασύνδεσης του DFT.DFP.lbb με άλλα συστατικά μέρη της εφαρμογής. Στο παράδειγμα αυτό φαίνεται ότι η εφαρμογή *DFT.DFP* αποτελείται από ένα *Image Processing Controller (ipc)* συνδεδεμένο με δύο *Image Processing Units (ipu)* τα οποία είναι με την σειρά του συνδεδεμένα με ένα *Load Balancer and Broadcaster (lbb).*

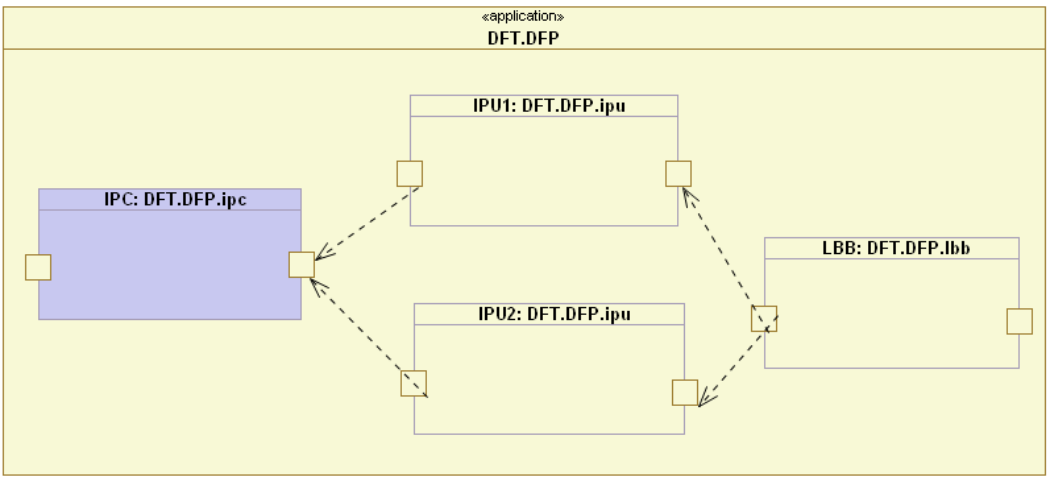
Επιπλέον, είναι δυνατό να προσδιοριστούν απαιτήσεις QoS για τις συνδέσεις. Στο Σχήμα 6.8 φαίνεται ένα παράδειγμα με προφίλ για τρεις



Σχήμα 6.5: *Ορισμός περιορισμών QoS*

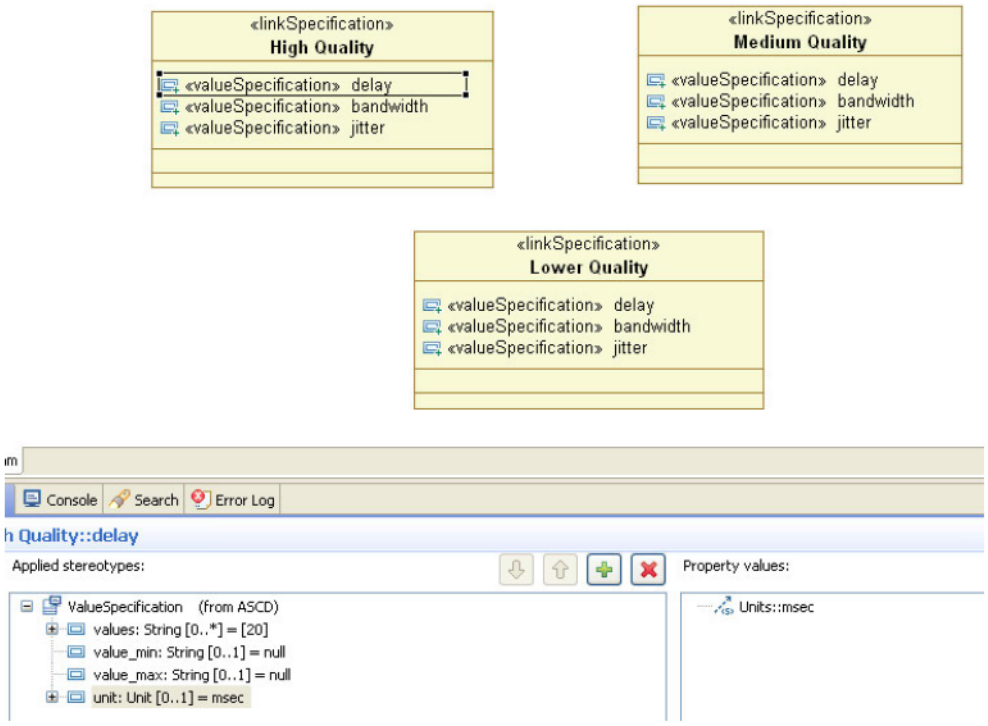


Σχήμα 6.6: *Ορισμός περιορισμού φυσικών πόρων*



Σχήμα 6.7: *Περιγραφή εφαρμογής*

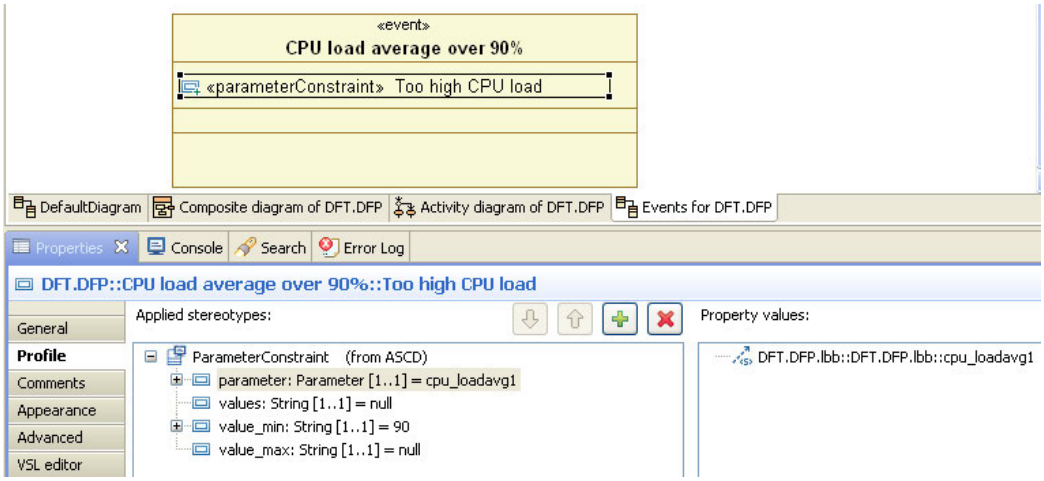
τύπους συνδέσεων (υψηλής, μεσαίας και κατώτερης ποιότητας). Αυτά τα προφίλ μπορούν να χρησιμοποιηθούν για να καθορισθούν οι απαιτήσεις QoS των συνδέσεων της εφαρμογής.



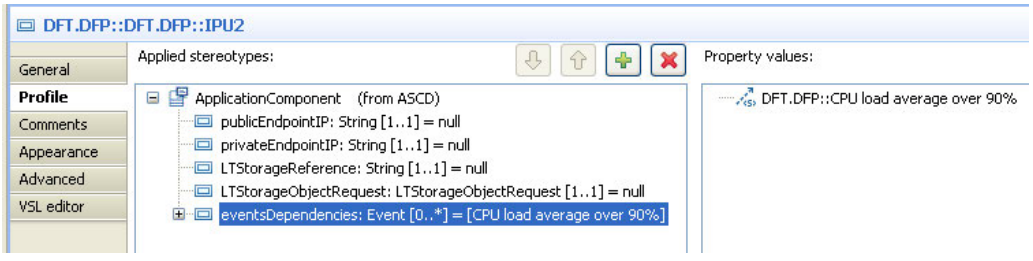
Σχήμα 6.8: *Περιγραφή απαιτήσεων QoS για τις συνδέσεις*

Η εφαρμογή μπορεί να περιέχει υπηρεσίες που ενεργοποιούνται κάτω από ορισμένες συνθήκες. Για τον λόγο αυτό είναι απαραίτητο να μπορούν να ορισθούν γεγονότα. Στο Σχήμα 6.9 ορίζεται ένα γεγονός που δηλώνει ότι ο φόρτος της ΚΜΕ είναι πάνω από 90%. Στο Σχήμα 6.10 φαίνεται πως ένα γεγονός μπορεί να συσχετισθεί με μία υπηρεσία.

Στη συνέχεια μπορεί να ορισθεί η ροή εργασίας με την χρήση ενός διαγράμματος ενεργειών UML. Στο Σχήμα 6.11 φαίνεται η ροή εργασίας που ορίζει μία απλή εφαρμογή. Πρώτα ξεκινάει η υπηρεσία *Image Processing*

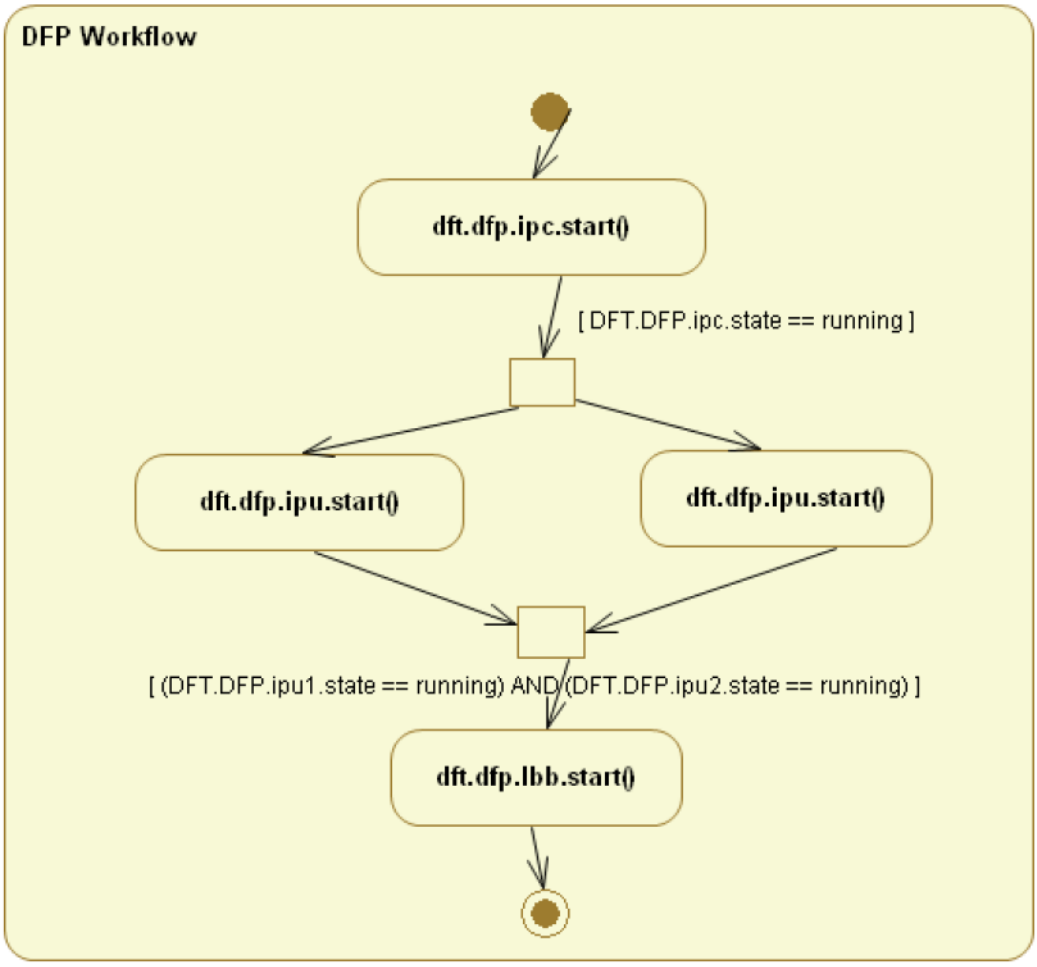


Σχήμα 6.9: *Μοντελοποίηση ενός γεγονότος*



Σχήμα 6.10: *Συσχέτιση ενός γεγονότος με μία υπηρεσία*

Controller (ipc). Όταν είναι έτοιμη, καλούνται δύο υπηρεσίες *Image Processing Units (ipu)* και τέλος καλείται η υπηρεσία *Load Balancer and Broadcaster (lbb).*



Σχήμα 6.11: *Ροή εργασίας σε UML*

Στη συνέχεια η ροή εργασίας που έχει ορισθεί με χρήση UML μετατρέπεται μέσω της εφαρμογής MOFScript (Oldevik, 2006) στην γλώσσα περιγραφής που παρουσιάζεται στην παράγραφο 6.7.

6.7 Γλώσσα Περιγραφής Ροών Εργασίας

Για να περιγραφούν ροές εργασίας με περιγραφή ποιότητας υπηρεσίας αναπτύχθηκε μία καινούρια γλώσσα η οποία είναι βασισμένη στην BPEL. Η επιλογή αυτή οφείλεται στο γεγονός ότι η BPEL έχει γίνει η *de facto* γλώσσα περιγραφής ροών εργασίας. Σκοπός των επεκτάσεων ήταν να υποστηριχθεί η περιγραφή χαρακτηριστικών ποιότητας υπηρεσίας.

6.7.1 Βασική Δομή

Η βασική δομή της δημιουργηθείσης γλώσσας είναι η invoke που χρησιμοποιείται για την κλήση μίας υπηρεσίας της ροής εργασίας. Η δομή invoke αποτελείται από τα εξής μέρη:

- Η ετικέτα name χρησιμοποιείται για να προσδιορίσει την συγκεκριμένη κλήση.
- Η ετικέτα partnerlink προσδιορίζει την υπηρεσία που θα κληθεί. Αυτό είναι απαραίτητο γιατί κατά τη φάση δημιουργίας της ροής εργασίας δεν είναι γνωστές όλες η πληροφορίες που χρειάζεται ο εκτελεστής για να κάνει την κλήση. Έτσι η περιγραφή της ροής εργασίας περιέχει την ταυτότητα της υπηρεσίας που θα πρέπει να κληθεί και κατά την φάση εκτέλεσης ο εκτελεστής χρησιμοποιεί αυτή την πληροφορία για να βρει την διεύθυνση IP της υπηρεσίας.
- Η ετικέτα legacy χρησιμοποιείται για να δηλώσει κατά πόσο το καλούμενο component έχει υλοποιηθεί ως web υπηρεσία ή είναι κληροδοτημένος κώδικας. Στη δεύτερη περίπτωση υπονοείται ότι έχει

χρησιμοποιηθεί ένας μηχανισμός κατά τον οποίο ο κληροδοτημένος κώδικας «περιβάλλεται» (wrapped) από μία υπηρεσιοστρεφή διεπαφή. Πιο συγκεκριμένα, ο υπεύθυνος για την ανάπτυξη του κώδικα έχει υλοποιήσει ένα script σε κάποια γλώσσα (perl, shell script κλπ.). Σε αυτήν την περίπτωση μία ειδική υπηρεσία που είναι μέρος του ASC καλείται από τον εκτελεστή ροών εργασίας. Η υπηρεσία αυτή αναλαμβάνει την εκτέλεση του συγκεκριμένου script. Έτσι είναι δυνατόν το ΣΔΡΕ να υποστηρίζει και κληροδοτημένο κώδικα αφού αυτός «περιβάλλεται» από μία υπηρεσιοστρεφή διεπαφή.

- Η ετικέτα security είναι προαιρετική και χρησιμοποιείται για να ορισθεί το επίπεδο ασφαλείας που απαιτείται μεταξύ του Εκτελεστή της ροής εργασίας και των υπηρεσιών. Αν δεν έχει ορισθεί, τότε δεν εφαρμόζεται κανένα μέτρο ασφαλείας. Αν έχει ορισθεί, τότε υπάρχει η επιλογή της ασφάλειας μηνύματος (επιλογή message), όπου χρησιμοποιείται ασφάλεια κατά την προδιαγραφή WS-SecureConversation (*WS-SecureConversation 1.3*, 2007), και η επιλογή transport η οποία ορίζει ασφάλεια στο επίπεδο της μεταφοράς με χρήση του πρωτοκόλλου TLS (Dierks & Rescorla, 2008).

- Η ετικέτα operation χρησιμοποιείται για να ορισθεί το ποια λειτουργία θα εκτελεσθεί στην καλούμενη υπηρεσία. Αν πρόκειται για κληροδοτημένο κώδικα, τότε ορίζεται το όνομα του script που θα εκτελεσθεί.

- Οι ετικέτες input και output χρησιμοποιούνται για τη διαχείριση δεδομένων. Οι διευθύνσεις URL της πηγής και του προορισμού ορίζονται ως διευθύνσεις GridFTP (Allcock, Bresnahan, Kettimuthu, & Link, 2005)

για απομακρυσμένα αρχεία και σαν διευθύνσεις URL αρχείων, αν πρόκειται για δεδομένα που βρίσκονται στο τοπικό σύστημα αρχείων.

- Η ετικέτα event χρησιμοποιείται για να ορισθούν γεγονότα που παράγονται από τον Evaluator και τις ενέργειες που θα εκτελεσθούν σε κάθε περίπτωση. Γεγονότα μπορούν να προέρχονται τόσο από την εφαρμογή όσο και από τον πάροχο της υποδομής. Ο Evaluator είναι υπεύθυνος να αναγνωρίζει την κάθε κατάσταση και να παράγει γεγονότα που είναι γνωστά στον Εκτελεστή Ροών Εργασίας.

- Η ετικέτα error χρησιμοποιείται για να ορισθούν οι ενέργειες που θα πρέπει να εκτελεσθούν σε περιπτώσεις σφάλματος. Υπάρχουν τρεις επιλογές:

    1. με την επιλογή retry ορίζεται ότι ο εκτελεστής θα πρέπει να δοκιμάσει να εκτελέσει ξανά μία υπηρεσία. Αυτό μπορεί να γίνει κατά μέγιστο δύο φορές πριν η εκτέλεση της ροής εργασίας αποτύχει.

    2. με την επιλογή fail ορίζεται ότι η εκτέλεση θα πρέπει να αποτύχει στο σύνολό της και θα πρέπει να ενημερωθεί ο Διαχειριστής Ροών Εργασίας.

    3. με την επιλογή custom ο υπεύθυνος για την ανάπτυξη του κώδικα έχει την δυνατότητα να ορίσει γεγονότα που καλύπτουν τις συγκεκριμένες ανάγκες του. Προς το παρόν υποστηρίζεται η κλήση άλλων υπηρεσιών.

```
<invoke>
  <name>"name of the invoke" </name>
  <partnerLink>"ID of the ASC to be executed"</partnerLink>
  <legacy>"true/false"</legacy>
  <security>"message/transport"</security >
  <operation>"operation to be called on service"</operation>
  <input>
    <sourceURL>"source of input" <sourceURL>
    <destinationURL>"destination of input"</destinationURL>
  </input>
  <output>
    <sourceURL>"source of output" </sourceURL>
    <destinationURL>"destination of output" </destinationURL>
  </output>
 <event>"event_id"
   <actions>
       <invoke> ... </invoke>
       ...
   </actions>
 </event>
 <error>"retry/fail/custom"
   <action>"retry/fail/custom"</action>
    <custom>
     <invoke> ... </invoke>
     <invoke> ... </invoke>
    </custom>
 </error>
</invoke>
```

Σχήμα 6.12: *Δομή Invoke*

6.7.2 Δομές Ελέγχου Ροής

Η υλοποίηση υποστηρίζει τις εξής δομές για την ροή του ελέγχου: sequence, while, wait και flow.

Sequence. Η δομή sequence χρησιμοποιείται για να δηλωθεί ένα σύνολο από ενέργειες που θα πρέπει να εκτελεστούν σειριακά και επιτρέπει την εμφώλευσή τους (nesting) σε άλλες δομές όπως την flow και την while.

While. Η δομή while χρησιμοποιείται για να δηλωθεί ένας βρόχος επανάληψης. Το πεδίο component_ID ορίζει το component που έχει την μεταβλητή που χρησιμοποιείται από το πεδίο ορισμού της συνθήκης του βρόχου (πεδίο condition). Η δομή παρουσιάζεται στο σχήμα 6.13.

```
<while>
   <condition> Boolean experession </condition>
   <component_ID> component_ID </component_ID>
   <invoke> ... </invoke>
   <flow> ... </flow>
    ...
</while>
```

Σχήμα 6.13: *Δομή While*

Wait. Η δομή wait χρησιμοποιείται για να σταματήσει προσωρινά η εκτέλεση της ροής εργασίας για κάποιο χρονικό διάστημα. Το διάστημα αυτό ακολουθεί την δομή Χρόνια - Μήνες - Μέρες - Ώρες - Λεπτά - Δευτερόλεπτα και φαίνεται στο σχήμα 6.14.

```
<wait>
   <name> "Wait1" </name>
   <for> P0Y0M0DT0H0M0S </for>
</wait>
```

Σχήμα 6.14: *Δομή Wait*

Flow. Η δομή flow (Σχήμα 6.15) χρησιμοποιείται για να ορισθεί μία ομάδα εργασιών που θα εκτελεσθεί παράλληλα. Η εκτέλεση της ροής σταματάει μέχρι να τελειώσουν όλες οι εργασίες του ορισμένου συνόλου, παρέχει δηλαδή ένα σημείο συγχρονισμού της ροής. Σε μία δομή flow μπορεί να εμφωλευθεί οποιαδήποτε άλλη δομή, όπως invoke, sequence και flow.

```
<flow>
   <invoke>    ... </invoke>
   <sequence> ... </sequence>
    ...
</flow>
```

Σχήμα 6.15: *Δομή Flow*

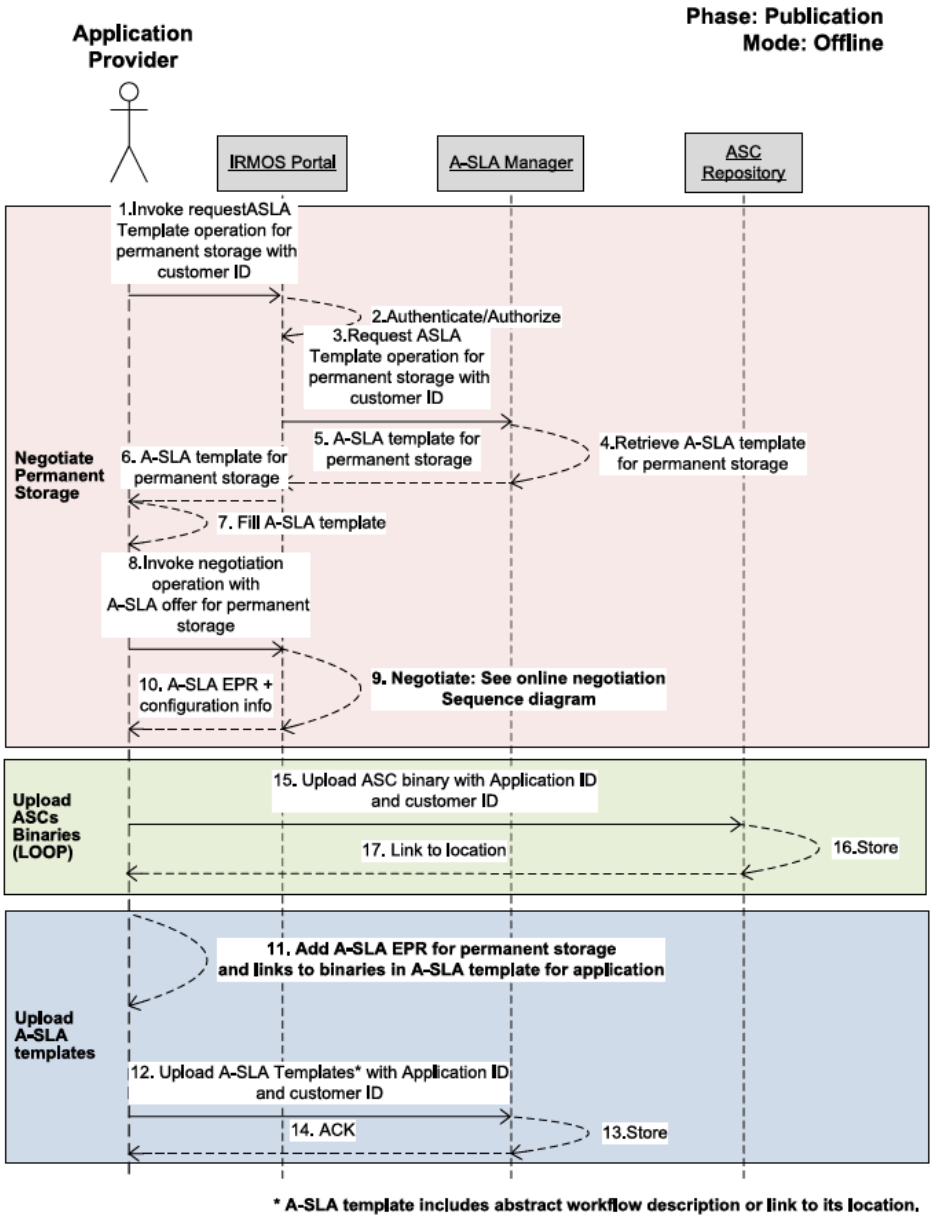
6.8 Αλληλεπιδράσεις Υπηρεσιών

Για να μπορέσει μία εφαρμογή να τρέξει στην πλατφόρμα IRMOS πρέπει να ακολουθηθούν κάποια βήματα, τα οποία περιλαμβάνουν την δημιουργία της περιγραφής της ροής εργασίας, την διαπραγμάτευση με την πλατφόρμα για τους όρους υπό τους οποίους θα εκτελεστεί η εφαρμογή και τέλος την εκτέλεση της εφαρμογής βάσει των συμφωνηθέντων.

Πιο συγκεκριμένα μία εφαρμογή πρέπει να περάσει από τις εξής φάσεις:

1. *Φάση δημοσίευσης*.

   Σε αυτή τη φάση (Σχήμα 6.16) ο δημιουργός της ροής εργασίας παράγει την περιγραφή της ροής εργασίας σε μία αόριστη μορφή. Αυτό σημαίνει ότι η ροή εργασίας δεν περιέχει τους συγκεκριμένους πόρους που θα χρησιμοποιηθούν.



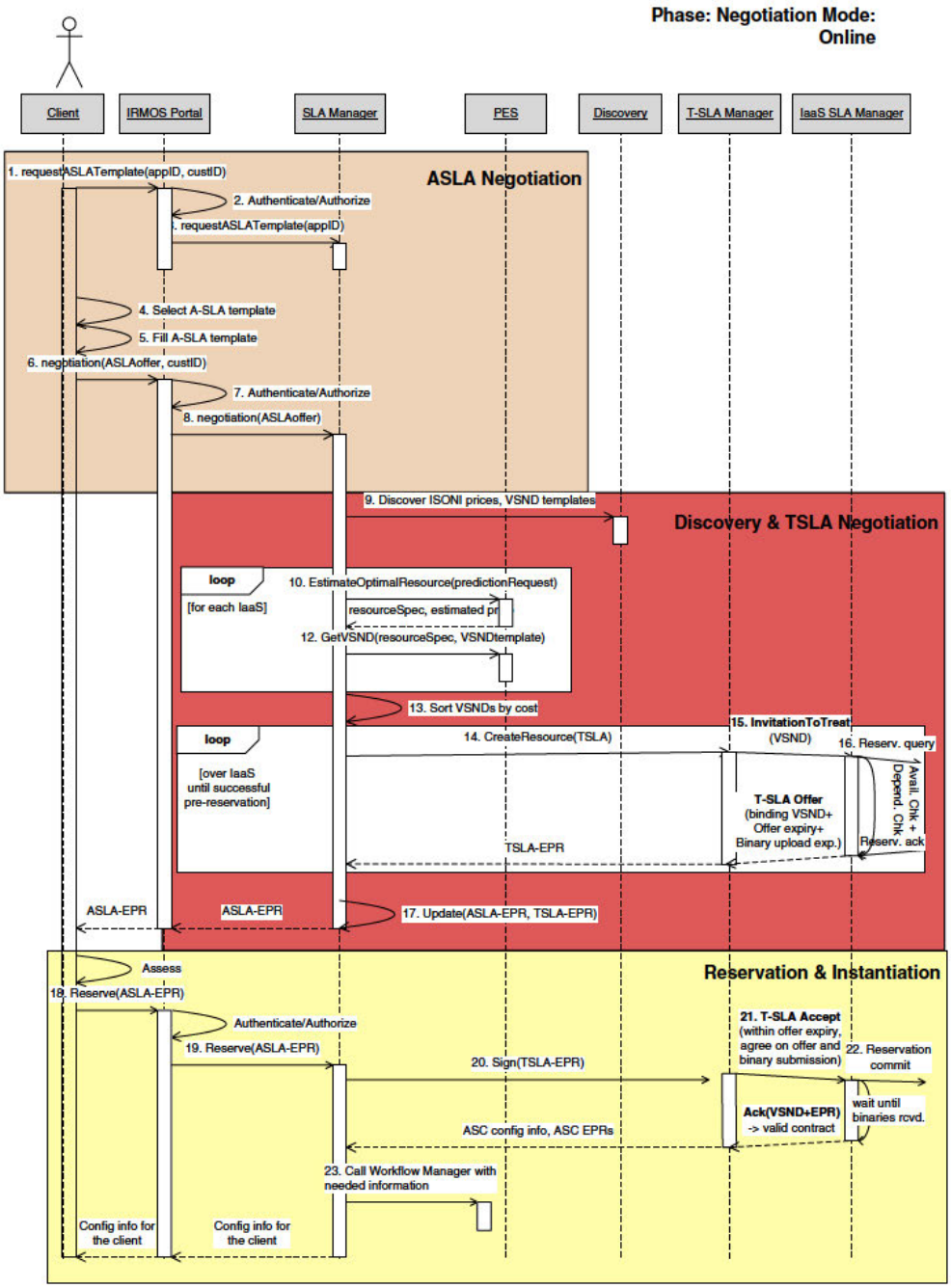
Σχήμα 6.16: *Ακολουθιακό διάγραμμα φάσης δημοσίευσης*

2. *Φάση διαπραγμάτευσης.*

   Κατά την φάση διαπραγμάτευσης (Σχήμα 6.17) ο πελάτης θέτει τις απαιτήσεις του προς την πλατφόρμα. Ο Διαχειριστής Συμβολαίων Επιπέδου Υπηρεσίας καλεί τον Διαχειριστή Ροών Εργασίας και του μεταφέρει τις εξής πληροφορίες: (α) Το κείμενο της περιγραφής της ροής εργασίας, (β) τις διευθύνσεις IP όλων των συστατικών της ροής εργασίας μαζί με τις διευθύνσεις IP των συστατικών της πλατφόρμας, όπως του Εκτελεστή Ροών Εργασίας, την Υπηρεσία Παρακολούθησης και τον Αξιολογητή, που βρίσκονται εντός του εικονικοποιημένου περιβάλλοντος, (γ) παραμέτρους διαμόρφωσης για τα ASC, (δ) το ID του SLA στο οποίο ανήκει η ροή εργασίας και (ε) τη χρονική στιγμή κατά την οποία θα πρέπει η εφαρμογή να είναι έτοιμη προς εκτέλεση. Με την παραλαβή των ανωτέρω πληροφοριών ο Διαχειριστής Ροών Εργασίας ενημερώνει την ροή εργασίας με τους συγκεκριμένους πόρους που θα χρησιμοποιηθούν, κάνοντάς την πλέον συγκεκριμένη, και δημιουργεί έναν επίμονο πόρο WSRF που αποθηκεύει όλη την πληροφορία.

3. *Φάση εκτέλεσης.*

   Προ-εκτελεστικό στάδιο. Πριν από την πραγματική εκτέλεση της εφαρμογής προηγείται το προεκτελεστικό στάδιο. Το στάδιο αυτό ξεκινάει βάσει του χρόνου που θα πρέπει να είναι διαθέσιμη η εφαρμογή. Κατά τη διάρκεια αυτού του σταδίου αρχικοποιούνται όλες οι εικονικές μηχανές από τον πάροχο IaaS. Ο Διαχειριστής Ροών Εργασίας καλεί τον Εκτελεστή που είναι υπεύθυνος για την συγκεκριμένη εφαρμογή και βρίσκεται εντός του εικονικοποιημένου περιβάλλοντος, περνώντας

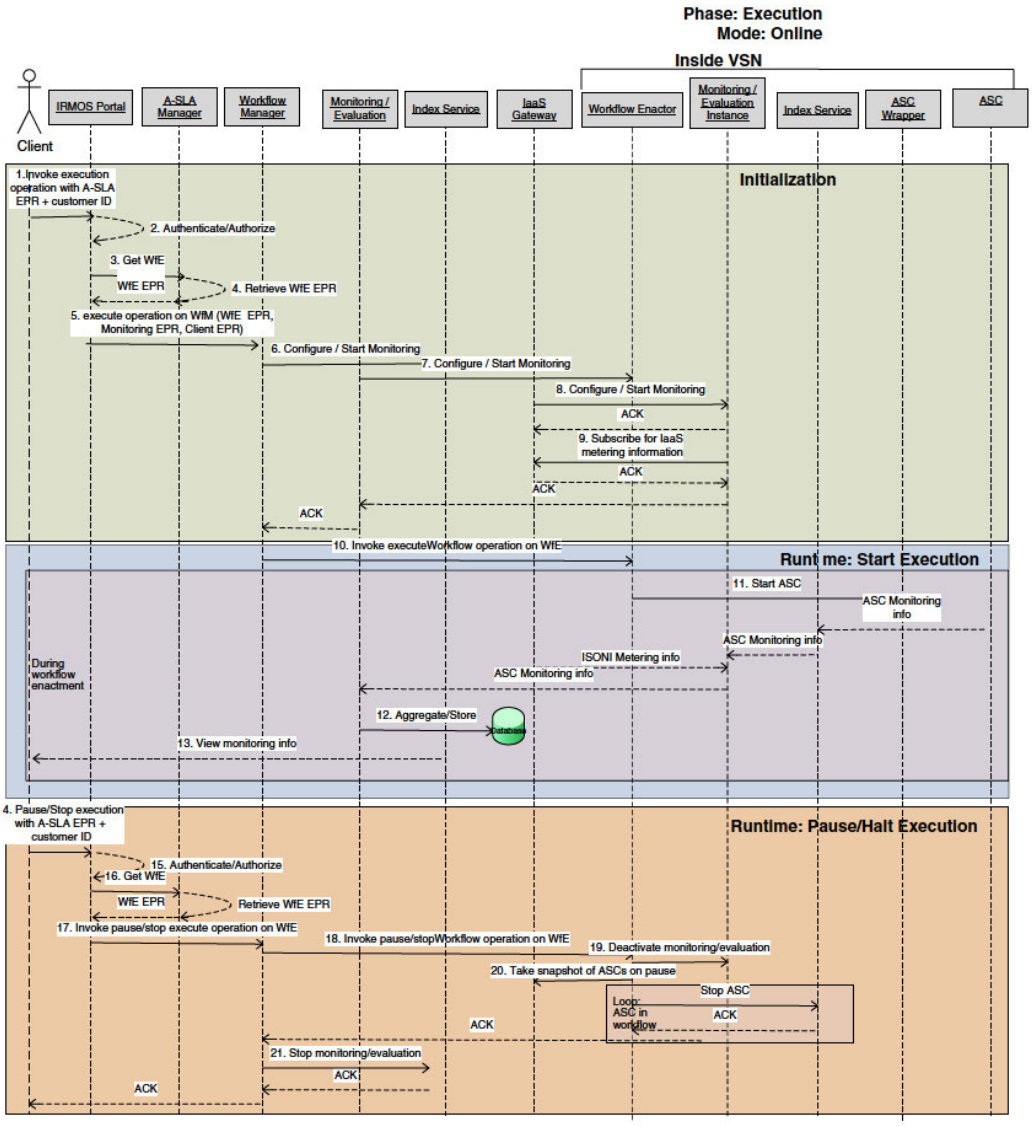


Σχήμα 6.17: *Ακολουθιακό διάγραμμα φάσης διαπραγμάτευσης*

του το κείμενο περιγραφής της ροής εργασίας μαζί με όλες τις πληροφορίες διαμόρφωσης για τα ASC. Ο Εκτελεστής χρησιμοποιεί αυτές τις πληροφορίες για να διαμορφώσει όλα τα ASC, καθιστώντας τα έτοιμα προς εκτέλεση.

Στάδιο εκτέλεσης. Η εκτέλεση της εφαρμογής ξεκινάει όταν ο χρήστης εισέρχεται στην πλατφόρμα μέσω της Υπηρεσίας Πύλης, εισάγοντας τα κατάλληλα στοιχεία για την διαπίστευσή του. Αν τα στοιχεία είναι σωστά, ο χρήστης δύναται να ζητήσει την εκτέλεση της εφαρμογής. Η Υπηρεσία Πύλης μεταφέρει αυτήν την εντολή μαζί με το ID του SLA υπό το οποίο θα εκτελεστεί η ροή εργασίας στον Διαχειριστή Ροών Εργασίας. Στη συνέχεια ειδοποιείται η υπηρεσία Παρακολούθησης, που είναι υπεύθυνη για την παρακολούθηση της εκτέλεσης της εφαρμογής, και της προωθούνται οι παράμετροι που θα πρέπει να ελέγχονται. Επίσης ειδοποιείται και ο Αξιολογητής. Όταν όλες οι υπηρεσίες είναι έτοιμες, ο Διαχειριστής καλεί τον Εκτελεστή Ροών Εργασίας και ζητεί την εκτέλεση της εφαρμογής. Ο εκτελεστής χρησιμοποιεί την περιγραφή της ροής εργασίας για να εκτελέσει κάθε διαφορετική εργασία. Το ακολουθιακό διάγραμμα της φάσης εκτέλεσης φαίνεται στο Σχήμα 6.18.

Προκειμένου να υποστηριχθεί η αλληλεπίδραση με τον τελικό χρήστη δίνεται η δυνατότητα μία ροή εργασίας να διακοπεί ή να τεθεί σε ύπνωση. Σε αυτήν την περίπτωση το αίτημα του χρήστη μεταφέρεται από την υπηρεσία Πύλης μέσω του Διαχειριστή στον Εκτελεστή Ροών Εργασίας. Σε περίπτωση τερματισμού, ο Εκτελεστής είναι υπεύθυνος να εκτελέσει την διαδικασία διακοπής σε κάθε ASC που εκτελείται



Σχήμα 6.18: *Ακολουθιακό διάγραμμα φάσης εκτέλεσης*

εκείνη τη στιγμή και να σταματήσει την περαιτέρω εκτέλεση της ροής, ειδοποιώντας ταυτόχρονα τις άλλες υπηρεσίες, όπως την υπηρεσία παρακολούθησης και τον Αξιολογητή. Σε περίπτωση που ζητηθεί η ύπνωση της ροής τότε ο Εκτελεστής επικοινωνεί με το εικονικό περιβάλλον και ζητεί να τεθούν όλες οι εικονικές μηχανές σε κατάσταση ύπνωσης. Στη συνέχεια λαμβάνεται ένα αντίγραφο κάθε μηχανής (το οποίο περιλαμβάνει την τρέχουσα μνήμη) το οποίο αποθηκεύεται. Από εκεί και πέρα ο πάροχος είναι δυνατόν να χρησιμοποιήσει τους φυσικούς πόρους για να εκτελεστούν άλλες εικονικές μηχανές, χωρίς όμως εγγυήσεις. Στη συνέχεια ο χρήστης μπορεί να ζητήσει να επανέλθει η κανονική λειτουργία της εφαρμογής εντός των χρονικών ορίων για τα οποία ισχύει η δέσμευση των πόρων.

Επίσης δίνεται η δυνατότητα στον χρήστη να αλλάξει τον ορισμό μίας ροής εργασίας που εκτελείται. Για να συμβεί αυτό, ο χρήστης πρέπει πρώτα να διακόψει προσωρινά την εκτέλεση, όπως περιγράφηκε παραπάνω. Αυτό συμβαίνει ώστε να διαφυλαχτεί η λογική ακεραιότητα της ροής εργασίας, καθώς αν συνεχίσουν να εκτελούνται ASC ενώ αλλάζει η περιγραφή της ροής μπορεί να προκύψουν προβλήματα. Στη συνέχεια, μέσω των εργαλείων που του παρέχονται από την πλατφόρμα για τον ορισμό μίας ροής εργασίας είναι δυνατόν να τροποποιήσει μία εργασία στα τμήματα εκείνα που δεν έχουν εκτελεσθεί προς το παρόν. Έτσι δημιουργείται μία νέα περιγραφή VSND. Η καινούρια περιγραφή είναι δυνατόν να έχει τις ίδιες απαιτήσεις πόρων όπως και η αρχική. Σε διαφορετική περίπτωση, αν δηλαδή απαιτούνται νέοι πόροι, τότε

απαιτείται μία καινούρια φάση διαπραγμάτευσης του SLA κατά την οποία ο πάροχος δύναται να απορρίψει την αλλαγή. Αν η αλλαγή γίνει αποδεκτή τότε η ανανεωμένη περιγραφή της ροής εργασίας μεταφέρεται στον Εκτελεστή ο οποίος αναδιαμορφώνει την υπηρεσία επισκόπησης καθώς και τον Αξιολογητή. Από εκεί και πέρα ο χρήστης είναι δυνατόν να ζητήσει την συνέχιση της ανανεωμένης ροής εργασίας.

Ο Εκτελεστής είναι επίσης δυνατόν να λαμβάνει μηνύματα από τον Αξιολογητή. Ο Αξιολογητής λαμβάνει συνεχώς δεδομένα από την υπηρεσία παρακολούθησης η οποία ελέγχει την εκτέλεση των διαφορετικών ASC και συλλέγει δεδομένα σχετικά με τις μετρικές που έχουν τεθεί ως «παρακολουθήσιμες» στο υπογραφέν SLA. Το SLA περιέχει επίσης κανόνες του οποίους χρησιμοποιεί ο Αξιολογητής για να παράγει συμβάντα βάσει των μετρικών που είναι υπό παρακολούθηση. Οι κανόνες μπορούν να περιέχουν απλούς τελεστές σύγκρισης ($=, >, <, \neq$) καθώς και λογικούς τελεστές AND και OR. Επίσης μπορούν να χρησιμοποιηθούν συμβάντα που παράγει ο πάροχος IaaS αναφορικά με την κατάσταση των φυσικών πόρων.

Σε κάθε κανόνα δίνεται ένα ID το οποίο χαρακτηρίζει τον κανόνα και το συμβάν που δημιουργεί. Επίσης ορίζονται οι παράμετροι που θα χρησιμοποιηθούν για να γίνει η αξιολόγηση. Καθώς κάθε Αξιολογητής είναι μοναδικός για κάθε VSN μόνο το ASC ID χρειάζεται για να ορισθεί μοναδικά μία παράμετρος. Αν κατά τη φάση εκτέλεσης η πρόταση που ορίζει τον κανόνα βεβαιωθεί ως αληθής τότε δημιουργείται ένα γεγονός. Παράδειγμα ενός κανόνα φαίνεται στο Σχήμα 6.19.

```
<event id="reconfigure">
<parameter id="droppedframes">
      <name>droppedframes</name>
      <ASC_ID>LB1</ASC_ID>
  </parameter>
  <rule>
   ((droppedframes>15)
   </rule>
</event>
```

Σχήμα 6.19: *Παράδειγμα κανόνα που ορίζει ένα γεγονός*

Η περιγραφή της ροής εργασίας περιέχει και πληροφορίες σχετικές με τα βήματα που θα πρέπει να ακολουθηθούν σε διάφορα γεγονότα. Τα βήματα αυτά μπορούν να περιλαμβάνουν την κλήση άλλων υπηρεσιών ή την επαναδιαμόρφωση υπηρεσιών που εκτελούνται. Ένα παράδειγμα είναι το ακόλουθο: Θεωρούμε μία υπηρεσία η οποία παρέχει βίντεο σε γνωστή μορφή και με συγκεκριμένο ρυθμό ροής πλαισίων (frame rate). Αν διαπιστωθεί ότι ο ρυθμός πέφτει, δηλαδή χάνονται πλαίσια, είναι δυνατόν η υπηρεσία να επαναδιαμορφωθεί ώστε να παράγει video σε άλλη μορφή η οποία θα έχει ανάγκη από λιγότερους πόρους, έως ότου προστεθεί στην ροή εργασίας μία νέα υπηρεσία η οποία θα επιτρέψει στην εφαρμογή να παρέχει video στην επιθυμητή ροή πλαισίων και μορφή. Αυτό δίνει την δυνατότητα μία μικρή και παροδική παραβίαση των ορών υπό τους οποίους εκτελείται η εφαρμογή να μην είναι καταστροφική για την εφαρμογή, αλλά να μπορεί να ανακάμψει εντός ενός μικρού χρονικού περιθωρίου.

Ο Εκτελεστής μπορεί επίσης να αντιδρά στην περίπτωση σφαλμάτων.

Η περιγραφή των σφαλμάτων και των βημάτων που θα πρέπει να ακολουθηθούν περιλαμβάνονται στην περιγραφή της ροής εργασίας. Οι ενέργειες που μπορεί να εκτελέσει ο Εκτελεστής μπορούν να είναι απλές, όπως να επαναεκτελεστεί μία υπηρεσία, να δοθεί σήμα αποτυχίας ολόκληρης της ροής εργασίας ή πιο σύνθετες. Αυτές περιγράφονται από τον χρήστη που έχει αναπτύξει την εφαρμογή και περιλαμβάνει ενέργειες όπως κλήση υπηρεσιών και επαναδιαμόρφωση αυτών. Είναι σημαντικό να τονιστεί εδώ η διαφορά ανάμεσα σε γεγονότα και σφάλματα. Γεγονότα είναι γνωστές καταστάσεις του συστήματος, τις οποίες ο δημιουργός της εφαρμογής γνωρίζει και έχει περιγράψει, όπως είναι το παράδειγμα της αλλαγής της μορφής ενός video που δόθηκε πρωτύτερα. Σφάλματα είναι καταστάσεις που δεν μπορεί να γνωρίζει από πριν ο δημιουργός, όπως παραδείγματος χάριν η αδυναμία κλήσης μίας υπηρεσίας. Σε τέτοιες περιπτώσεις μπορούν να εκτελεστούν γενικές ενέργειες, όπως να αποθηκευτεί η κατάσταση όλων των εικονικών μηχανών. Ο δημιουργός της εφαρμογής μπορεί να ορίσει τις ενέργειες αυτές στην περιγραφή της ροής εργασίας. Η επιλογή της επαναπροσπάθειας παρέχεται καθώς πολλά σποραδικά σφάλματα είναι δυνατόν να ξεπεραστούν μέσω αυτής της μεθόδου, ενώ είναι εύκολη και «φθηνή» στην υλοποίηση.

Κατά την διάρκεια εκτέλεσης της ροής εργασίας ο Εκτελεστής Ροών Εργασίας μπορεί να χρειαστεί να λάβει αποφάσεις βασιζόμενος σε μεταβλητές που κατέχουν τα ASC, όπως στην περίπτωση ενός βρόχου while. Η μεταβλητές αυτές αποθηκεύονται σε μία υπηρεσία δεδομένων (Index Service) η οποία βρίσκεται στην ίδια εικονική μηχανή με κάθε ASC.

Η υπηρεσία Index είναι μια ευέλικτη αποθήκη δεδομένων η οποία παρέχει πρόσβαση στα δεδομένα μέσω τυποποιημένων ερωτημάτων WSRF και διεπαφών subscription/notification. Έτσι, ο Εκτελεστής είναι δυνατόν να θέτει ερωτήματα στην υπηρεσία Index και να λαμβάνει την τιμή μεταβλητών που εμπεριέχονται στην περιγραφή της ροής δεδομένων.

Επίσης ο Εκτελεστής προωθεί πληροφορίες σχετικές με την κατάσταση της ροής εργασίας στον Διαχειριστή, ο οποίος με την σειρά του παρέχει πληροφόρηση στον χρήστη μέσω της υπηρεσίας Πύλης.

7 Ποσοτική και Ποιοτική Αξιολόγηση του Προτεινόμενου ΣΔΡΕ

Στον παρόν κεφάλαιο γίνεται πειραματική αξιολόγηση του προτεινόμενου συστήματος διαχείρισης ροών εργασίας μέσω μεμονωμένων πειραμάτων καθώς και μέσω μίας εφαρμογής μεταπαραγωγής βίντεο (video post-production).

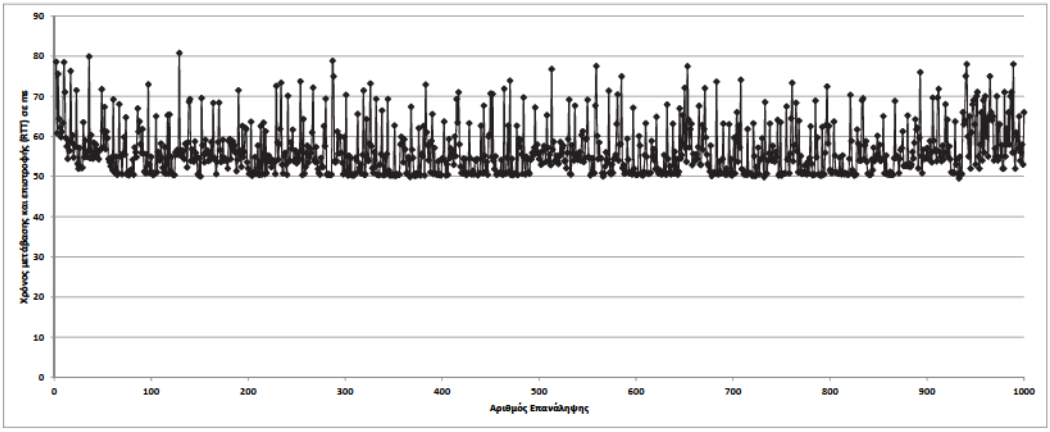
7.1 Αξιολόγηση Λήψης Γεγονότων.

Το πρώτο πείραμα είχε σκοπό την αξιολόγηση του συστήματος ως προς την ικανότητα του να αναγνωρίζει καταστάσεις και να δρα αναλόγως. Για το λόγο αυτό ένα ASC παρήγαγε ένα γεγονός το οποίο αφού αξιολογήθηκε από τον Αξιολογητή μεταφέρθηκε στον Εκτελεστή ο οποίος με την σειρά του ενημέρωνε το ASC για την λήψη του μηνύματος. Σε κάθε περίπτωση γινόταν μέτρηση του χρόνου που χρειάστηκε το μήνυμα.

Για το πείραμα χρησιμοποιήθηκαν 4 υπολογιστές P4 - 3 GHz με 1 GB μνήμης RAM οι οποίοι έτρεχαν ως λειτουργικό το Fedora Core 9 με

πυρήνα 2.6.27 και στους οποίους ήταν εγκατεστημένες οι υπηρεσίες της πλατφόρμας, ενώ 4 υπολογιστές Athlon 62 2 * 2,4 GHz με 4 GB μνήμης RAM και 6 υπολογιστές Core2 Quad Q9400 - 4 * 2,66GHz με 8 GB μνήμης RAM χρησιμοποιήθηκαν για να φιλοξενήσουν τα ASC.

Στην πρώτη περίπτωση ο Εκτελεστής Ροών εργασίας, ο Αξιολογητής και το ASC ήταν μέρη του ίδιου εικονικού δικτύου υπηρεσιών (VSN). Το πείραμα εκτελέστηκε χίλιες φορές και τα αποτελέσματα φαίνονται στο Σχήμα 7.1. Η ανάλυση των αποτελεσμάτων φαίνεται στον πίνακα 7.1.



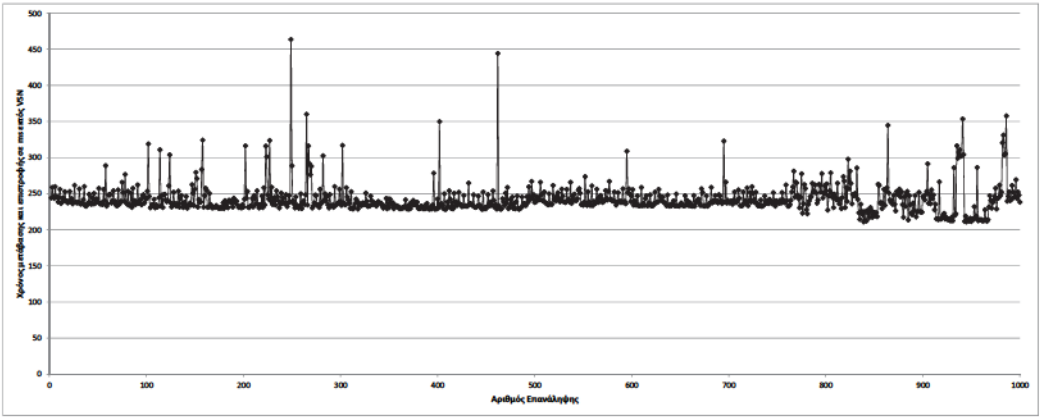
Σχήμα 7.1: *Χρόνος μετάβασης και επιστροφής μηνύματος όταν τα μέρη βρίσκονται εντός του ίδιου VSN.*

Πίνακας 7.1: *Στατιστικά πειράματος χρόνου μετάβασης και επιστροφής μηνύματος όταν τα μέρη βρίσκονται εντός του ίδιου VSN*

Μέση Τιμή	56,25711099
Ελάχιστη Τιμή	49,568508
Μέγιστη Τιμή	80,772051
Τυπική Απόκλιση	6,226931415
Διάστημα Εμπιστοσύνης ($a = 0,05$)	[55, 87116887, 56, 6430531]
90ο εκατοστημόριο	65,4802962

Στη συνέχεια εκτελέστηκε το ίδιο πείραμα με την διαφορά ότι δεν υπήρχε

εικονικό δίκτυο υπηρεσιών και τα διαφορετικά μέρη (Εκτελεστής, Αξιολογητής και ASC) ήταν διασκορπισμένα γεωγραφικά. Όπως είναι προφανές, σε αυτή την περίπτωση δεν υπήρχε εγγύηση της ποιότητας της υπηρεσίας. Τα αποτελέσματα φαίνονται στο Σχήμα 7.2 και η στατιστική ανάλυση στον πίνακα 7.2.



Σχήμα 7.2: *Χρόνος μετάβασης και επιστροφής μηνύματος χωρίς τη δημιουργία VSN.*

Πίνακας 7.2: *Στατιστικά πειράματος χρόνου μετάβασης και επιστροφής μηνύματος χωρίς την δημιουργία VSN*

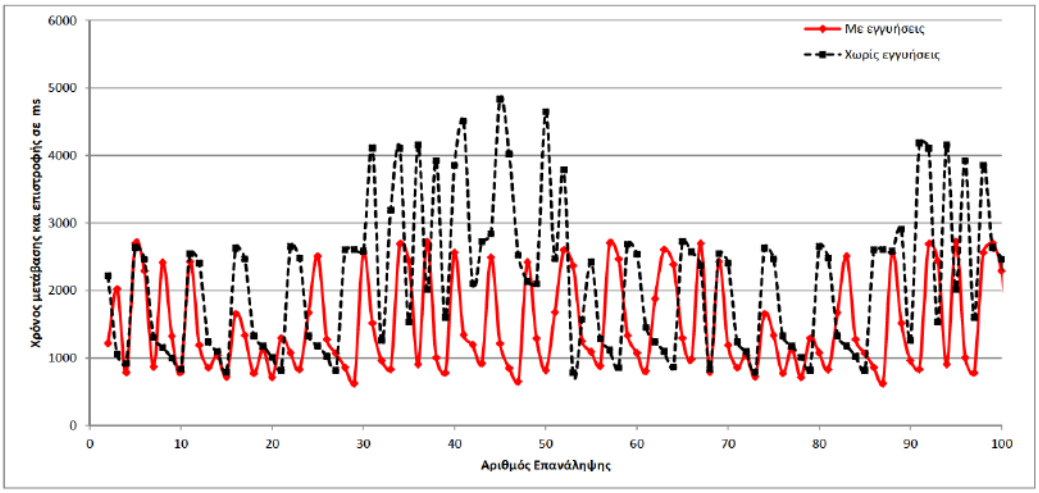
Μέση Τιμή	242,5430876
Ελάχιστη Τιμή	211,406836
Μέγιστη Τιμή	463,699074
Τυπική Απόκλιση	20,53237998
Διάστημα Εμπιστοσύνης ($a = 0,05$)	[241, 3675477, 243, 9308398]
90ο εκατοστημόριο	258,04335

Τα αποτελέσματα του πειράματος είναι αναμενόμενα. Στην πρώτη περίπτωση ο παραγωγός των γεγονότων (το ASC) και οι απαραίτητες υπηρεσίες της πλατφόρμας (Εκτελεστής, Αξιολογητής και Υπηρεσία Παρακολούθησης) βρίσκονται γεωγραφικώς κοντά και μάλιστα το μεταξύ τους δίκτυο επικοινωνίας παρέχει εγγυήσεις ποιότητας υπηρεσίας. Έτσι, όχι μόνο

είναι χαμηλοί οι χρόνοι αλλά και δεν ξεπερνούν το όριο των 80ms. Αντιθέτως, στην περίπτωση που δεν δημιουργείται VSN, όχι μόνο δεν είναι σίγουρο ότι τα διαφορετικά μέρη θα βρίσκονται σε μικρή απόσταση, αλλά και το μεταξύ τους δίκτυο δεν είναι εγγυημένο. Έτσι παρατηρούνται υψηλοί χρόνοι και μάλιστα με αρκετές αιχμές που απέχουν πολύ από την μέση τιμή. Έτσι γίνεται προφανές ότι αν δεν ακολουθείτο η προταθείσα πολυεπίπεδη αρχιτεκτονική δεν θα ήταν δυνατόν να παρασχεθούν εγγυήσεις ποιότητας υπηρεσίας ως προς την δυνατότητα της πλατφόρμας να αναγνωρίζει καταστάσεις και να τις αντιμετωπίζει.

7.2 Αξιολόγηση Εκτέλεσης Τετριμμένης Ροής Εργασίας με Χρήση VSN.

Σκοπός του δευτέρου πειράματος ήταν η αξιολόγηση της προταθείσης αρχιτεκτονικής και υλοποίησης σε συνθήκες φόρτου. Έτσι το πείραμα περιελάμβανε τον Εκτελεστή να καλεί μία τετριμμένη εφαρμογή η οποία απλά επιβεβαίωνε την λήψη της εντολής εκτέλεσης στον Εκτελεστή. Ταυτόχρονα σε μία τυχαία χρονική στιγμή ζητήθηκε η εκτέλεση άλλων εφαρμογών που κατανάλωναν υπολογιστική ισχύ στο ίδιο υπολογιστή που φιλοξενούσε την εικονική μηχανή. Κάθε φορά γινόταν μέτρηση του χρόνου μετάβασης και επιστροφής. Στο πρώτο σκέλος του πειράματος ο Εκτελεστής και η εφαρμογή ήταν μέρη του ίδιου VSN ενώ στο δεύτερο σκέλος του πειράματος δεν υπήρχε η δημιουργία VSN και έτσι δεν υπήρχε εγγύηση ως προς την υπολογιστική ισχύ ή το διαθέσιμο δίκτυο. Τα αποτελέσματα φαίνονται στο Σχήμα 7.3 ενώ στατιστική ανάλυση αυτών φαίνεται στον πίνακα 7.3.



Σχήμα 7.3: *Εκτέλεση εφαρμογής με και χωρίς την δημιουργία VSN.*

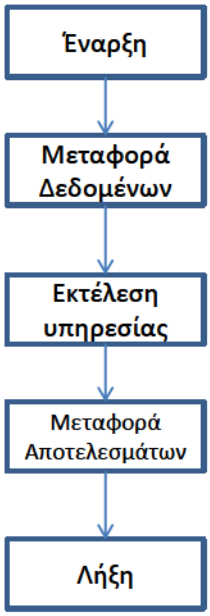
Πίνακας 7.3: *Στατιστικά πειράματος εκτέλεσης εφαρμογής με και χωρίς την δημιουργία VSN*

	Με δημιουργία VSN	**Χωρίς τη δημιουργία VSN**
Μέση Τιμή	1489,95	2149,71
Ελάχιστη Τιμή	625,07	783,13
Μέγιστη Τιμή	2721,39	4831,85
Τυπική Απόκλιση	709,73	1086,25
Διάστημα Εμπιστοσύνης (α=0,05)	[1350, 15, 1629, 76]	[1935, 74, 2363, 68]

Από τα αποτελέσματα είναι προφανές ότι όταν δεν έχουμε άλλο φόρτο στο μηχάνημα που εκτελεί την εικονική μηχανή τα αποτελέσματα είναι παραπλήσια. Εντούτοις, όταν προσομοιώνουμε ένα πραγματικό σύστημα οι δύο αρχιτεκτονικές συμπεριφέρονται διαφορετικά. Η προταθείσα λύση δεν επηρεάζεται από τον φόρτο του μηχανήματος οικοδεσπότη εξαιτίας των εγγυήσεων που μας προσφέρει το λειτουργικό σύστημα και τις οποίες μπορεί να εκμεταλλευτεί η πλατφόρμα μας μέσω του ορισμού ενός VSN. Από την άλλη πλευρά όταν δεν υπάρχουν εγγυήσεις φαίνεται ότι ο χρόνος εκτέλεσης εμφανίζει μεγάλες διακυμάνσεις οι οποίες θα μπορούσαν να έχουν αρνητική επίδραση σε μία εφαρμογή που έχει χαρακτηριστικά πραγματικού χρόνου.

7.3 Αξιολόγηση Εκτέλεσης Ροών Εργασίας σε Σύγκριση με μια Απλή Εφαρμογή για Διαφορετικές Ροές Εργασίας.

Στο επόμενο πείραμα αξιολογήθηκε το Σύστημα Διαχείρισης Ροών Εργασίας συγκρινόμενο με ένα απλό πρόγραμμα το οποίο εκτελεί τις ίδιες λειτουργίες (μεταφορά δεδομένων, κλήση υπηρεσιών) στατικά, χωρίς δηλαδή να περιέχει κάποια λογική, να δέχεται μηνύματα ή να μπορεί να αλλάξει τη ροή εργασίας δυναμικά. Έτσι το απλό αυτό πρόγραμμα μπορούμε να το θεωρήσουμε ως βάση και να μετρήσουμε την επιπλέον φόρτο που εισάγει το

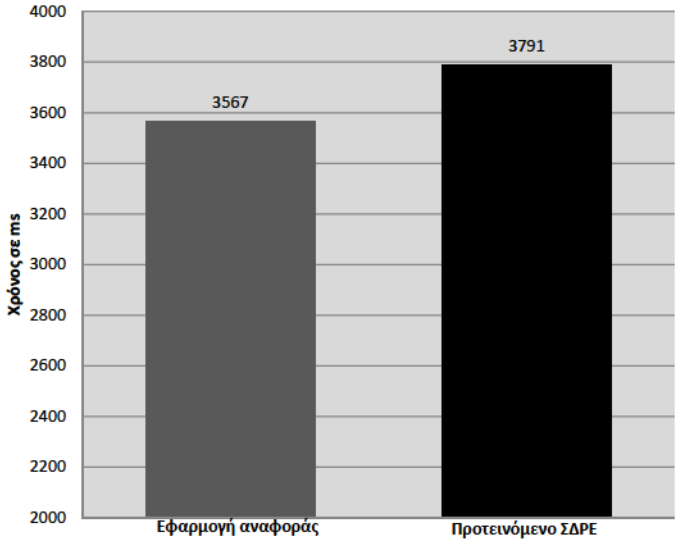


Σχήμα 7.4: *Απλή πειραματική ροή εργασίας*

Η πρώτη ροή εργασίας που χρησιμοποιήθηκε περιελάμβανε τη

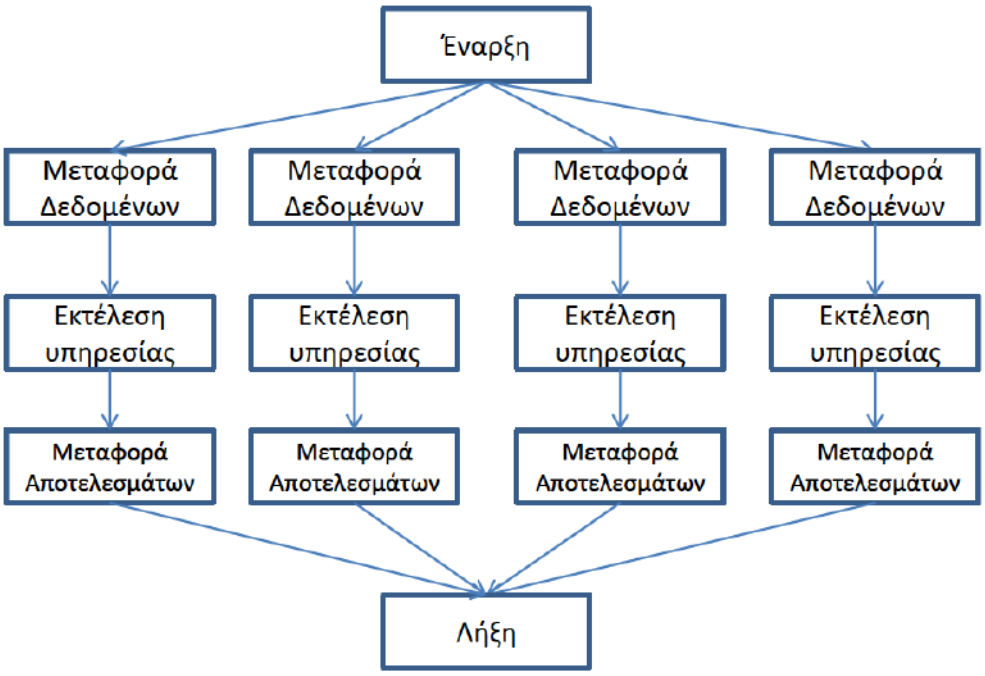
μεταφορά δεδομένων σε έναν κόμβο, την εκτέλεση μίας υπηρεσίας και τη μεταφορά των αποτελεσμάτων σε έναν άλλο κόμβο. Η ροή εργασίας φαίνεται σχηματικά στο Σχήμα 7.4. Για την μεταφορά των δεδομένων χρησιμοποιήθηκε το GridFTP.

Όπως φαίνεται από τα αποτελέσματα (Σχήμα 7.5), το προταθέν σύστημα διαχείρισης ροών εργασίας είναι περίπου 6% πιο αργό από την απλή εφαρμογή.



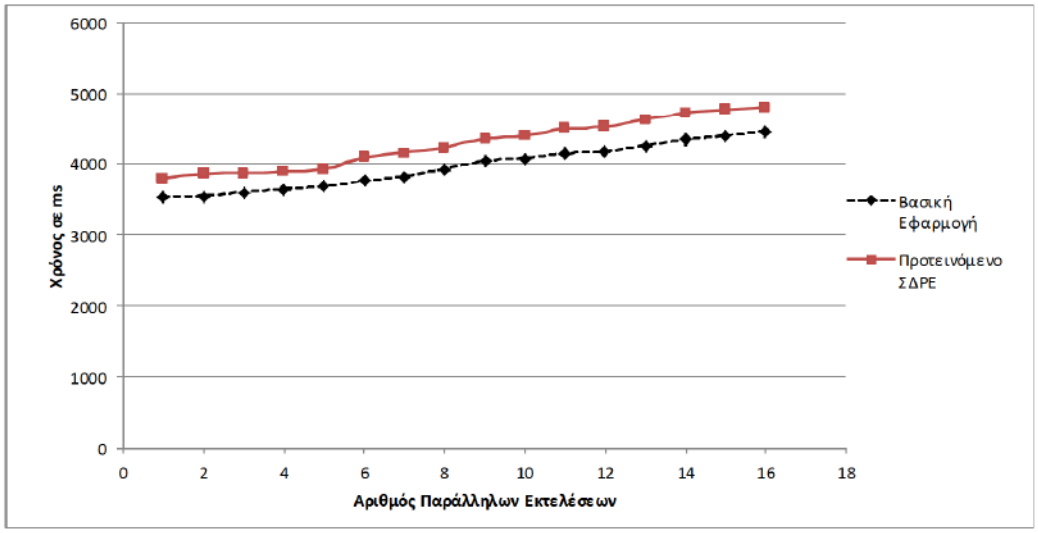
Σχήμα 7.5: *Χρόνος εκτέλεσης απλής ροής εργασίας*

Στο επόμενο πείραμα αξιολογήθηκε το προτεινόμενο ΣΔΡΕ σε σύγκριση με την βασική εφαρμογή κατά την εκτέλεση μίας πιο πολύπλοκης ροής εργασίας. Η ροή εργασίας περιελάμβανε την μεταφορά δεδομένων σε πολλαπλούς κόμβους, την εκτέλεση μίας υπηρεσίας σε κάθε κόμβο και τη μεταφορά των αποτελεσμάτων σε έναν κεντρικό κόμβο, όπως φαίνεται στο Σχήμα 7.6.



Σχήμα 7.6: *Ροή εργασίας με παράλληλες εκτελέσεις υπηρεσιών*

Το πείραμα εκτελέστηκε για διαφορετικό αριθμό παράλληλων εκτελέσεων έως και τις 16 ταυτόχρονες εκτελέσεις. Στο πείραμα αυτό δεν χρησιμοποιήθηκαν εικονικές μηχανές ώστε να μην υπεισέρχονται στις μετρήσεις σφάλματα που μπορεί να οφείλονται σε εξωγενείς, ως προς το ΣΔΡΕ, παράγοντες όπως παραδείγματος χάριν το κόστος της εικονικοποίησης. Τα αποτελέσματα του πειράματος φαίνονται στο Σχήμα 7.7. Ο προταθέν μηχανισμός είναι κατά 7,97% πιο αργός. Μπορούμε λοιπόν να συμπεράνουμε ότι το προταθέν ΣΔΡΕ παρουσιάζει παρόμοια συμπεριφορά με την βασική μηχανή μεταφοράς δεδομένων και κλήσης υπηρεσιών και ότι ο επιπρόσθετος φόρτος είναι χαμηλός.



Σχήμα 7.7: *Χρόνος εκτέλεσης ροής εργασίας με παράλληλες εκτελέσεις υπηρεσιών*

7.4 Αξιολόγηση μέσω Εφαρμογής Μεταπαραγωγής Βίντεο.

Το προτεινόμενο ΣΔΡΕ αξιολογήθηκε με χρήση τριών εφαρμογών που παρουσιάζουν απαιτήσεις πραγματικού χρόνου. Πιο συγκεκριμένα ο έλεγχος του ΣΔΡΕ έγινε μέσω μίας εφαρμογής eLearning, μίας εφαρμογής εικονικής πραγματικότητας και μίας εφαρμογής μεταπαραγωγής βίντεο (video post-production). Στο παρόν κεφάλαιο εστιάζουμε στην εφαρμογή μεταπαραγωγής βίντεο, καθώς αυτή παρουσιάζει τις μεγαλύτερες απαιτήσεις ως προς το ΣΔΡΕ.

7.4.1 Περιγραφή εφαρμογής μεταπαραγωγής βίντεο

Η μεταπαραγωγή βίντεο αποτελείται από πολλά βήματα. Συνήθως ξεκινάει με την εμφάνιση των αρνητικών, αν η πηγή είναι αναλογική κάμερα,

ή με την καταχώρηση αρχείων εικόνων, αν η πηγή είναι ψηφιακή κάμερα. Το τελικό προϊόν είναι το βίντεο που θα χρησιμοποιηθεί στους κινηματογράφους ή σε δίσκους DVD και Bluray.

Τα στιγμιότυπα που θα χρησιμοποιηθούν πρέπει να περάσουν από έλεγχο πριν συμπεριληφθούν στο τελικό προϊόν. Ο σκοπός του καθημερινού ελέγχου (Dailies review) είναι να εντοπιστούν σφάλματα ως προς την ποιότητα ή το περιεχόμενο εγκαίρως ώστε να υπάρχει αρκετός χρόνος για βελτιώσεις. Συνήθως από υλικό δύο ωρών, μόνο το ένα δέκατο χρησιμοποιείται στο τελικό προϊόν. Η κλασσική προσέγγιση στο πρόβλημα είναι να αποσταλούν τα στιγμιότυπα μίας μέρας σε συγκεκριμένα «σημαντικά» άτομα για να ελεγχθούν. Υπάρχει περίπτωση οι υπάρχουσες λήψεις να μην θεωρηθούν ικανοποιητικές από τους υπευθύνους, με αποτέλεσμα να πρέπει να γυριστεί ξανά μία σκηνή.

Η προτεινόμενη εφαρμογή επιτρέπει τον έλεγχο μίας ροής βίντεο και την εφαρμογή διορθώσεων σε αυτή ως προς το χρώμα. Η εφαρμογή είναι διαδραστική και επιτρέπει σε ομάδες τριών ως πέντε ανθρώπων να συμμετάσχουν σε μία κοινή συνεδρία χωρίς να χρειάζεται να βρίσκονται στην ίδια τοποθεσία. Κάθε ένας από τους συμμετέχοντες/θεατές χρειάζεται μόνο ένα μέτριο σταθμό εργασίας και σύνδεση στο Διαδίκτυο. Σε κάθε συνδεδεμένο τερματικό δίνεται η δυνατότητα απλού ελέγχου της ροής του βίντεο (έναρξη/διακοπή/παύση). Οι θεατές μπορούν να αποχωρήσουν από την συνεδρία ενώ νέοι θεατές μπορούν να συμμετάσχουν χωρίς να υπάρχει επίπτωση στους υπολοίπους. Όταν φύγει και ο τελευταίος θεατής, η ροή του βίντεο συνεχίζεται όπως ακριβώς συμβαίνει και με την μετάδοση του τηλεοπτικού σήματος. Πρέπει να σημειωθεί εδώ ότι το κανάλι επικοινωνίας μεταξύ των τερματικών και της εφαρμογής βρίσκεται εκτός των αρμοδιοτήτων

του παρόχου και έτσι δεν μπορούν να δοθούν εγγυήσεις ποιότητας υπηρεσίας. Παρ'ολα αυτά η εφαρμογή θα πρέπει να χρησιμοποιεί το υπάρχον εύρος ζώνης όσο καλύτερα γίνεται. Για το λόγο αυτό θα πρέπει να αναγνωρίζει ότι το διαθέσιμο εύρος ζώνης προς ένα τερματικό δεν είναι επαρκές και να παραλείπει καρέ μόνο προς το συγκεκριμένο τερματικό. Οι θεατές μπορούν να χρησιμοποιήσουν άλλα κανάλια επικοινωνίας, όπως τηλεφωνικές γραμμές, για την μεταξύ τους επικοινωνία.

Όταν οι θεατές το κρίνουν απαραίτητο μπορούν να γίνουν αλλαγές στις παραμέτρους του βίντεο. Οι αλλαγές αυτές θα πρέπει να γίνονται άμεσα και η ροή του βίντεο θα πρέπει να συνεχίζει σύμφωνα με τις νέες τιμές. Για το λόγο αυτό η επεξεργασία των εικόνων του βίντεο θα πρέπει να γίνεται σε πραγματικό χρόνο. Είναι αποδεκτό να χάνεται ένας μικρός αριθμός πλαισίων (frames) σποραδικά αλλά γενικώς η εφαρμογή θα πρέπει να παρέχει συνεχές βίντεο. Έτσι, η υποδομή πρέπει να παρέχει αρκετή υπολογιστική δύναμη και εύρος ζώνης ώστε η απόδοση της εφαρμογής να είναι σταθερή.

7.4.2 Αρχιτεκτονική Εφαρμογής Μεταπαραγωγής Βίντεο

Η εφαρμογή αποτελείται από τα εξής κύρια μέρη:

- *Διακομιστής Αποθήκευσης (Storage server).* Το πρωτότυπο υλικό θα πρέπει να είναι αποθηκευμένο σε έναν διακομιστή ο οποίος είναι αρκετά γρήγορος ώστε να παρέχει τα πλαίσια (frames) με ικανοποιητικό ρυθμό. Επίσης θα πρέπει να παρέχεται χαμηλή διακύμανση καθυστέρησης (jitter). Αυτό οφείλεται στο γεγονός ότι τα αρχεία είναι αρκετά μεγάλα και έτσι μειώνεται η δυνατότητα ενδιάμεσης αποθήκευσης (buffering)

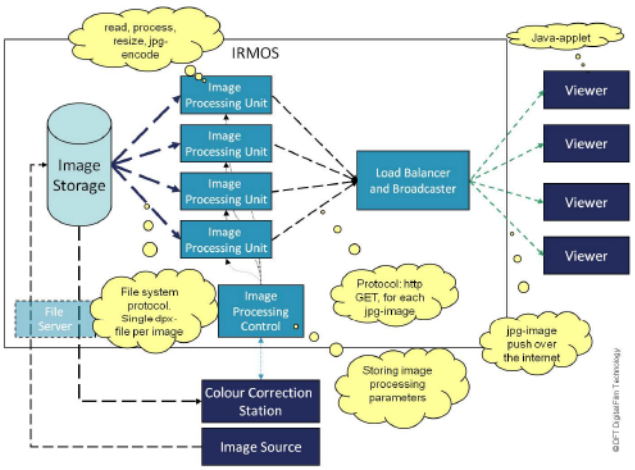
στην εφαρμογή.

- *Image Processing Control - IPC.* Το IPC είναι ένα κεντρικό συστατικό του συστήματος που είναι υπεύθυνο για τη διατήρηση των παραμέτρων επεξεργασίας της εικόνας που χρησιμοποιούν τα IPU. Το IPC θα πρέπει να είναι προσπελάσιμο από τον εξωτερικό σταθμό επεξεργασίας χρώματος καθώς ο υπεύθυνος για τις χρωματικές αλλαγές θα πρέπει να μεταφέρει στο IPC τις παραμέτρους επεξεργασίας της εικόνας.

- *Image Processing Unit - IPU.* Το IPU είναι υπεύθυνο να επεξεργάζεται κάθε πλαίσιο σύμφωνα με τις παραμέτρους που λαμβάνει από το IPC.

- *Load Balancer and Broadcaster - LBB.* Το LBB έχει διττό ρόλο. Από την μία μεριά συλλέγει και σειριοποιεί τις εικόνες που παράγουν τα πολλαπλά IPC και τις αναμεταδίδει στους συνδεδεμένους πελάτες. Από την άλλη μεριά, λαμβάνει υπ'οψη του το διαθέσιμο εύρος ζώνης κάθε συνδεδεμένου τερματικού ώστε η αναμετάδοση να γίνεται απροβλημάτιστα.

Το κάθε ένα από τα κύρια μέρη έχει μοντελοποιηθεί και περιγραφεί ως ASC. Επίσης κατά την φάση δημοσίευσης δημιουργήθηκε η αφηρημένη περιγραφή της ροής εργασίας, μέρος της οποίας φαίνεται στο Σχήμα 7.9. Η γενική απαίτηση της εφαρμογής ήταν να μπορεί να παράγει σταθερά 24 πλαίσια προς του συνδεδεμένους πελάτες. Μετά από ανάλυση έγινε σαφές ότι για να επιτευχθεί αυτός ο ρυθμός πλαισίων ήταν απαραίτητο να υπάρχουν 8 Image Processing Unit. Έτσι το δημιουργηθέν VSN αποτελούνταν από 10 εικονικές μηχανές που εκτελούσαν τα μέρη της εφαρμογής (1 IPC, 8 IPU



Σχήμα 7.8: *Αρχιτεκτονική εφαρμογής μεταπαραγωγής βίντεο*

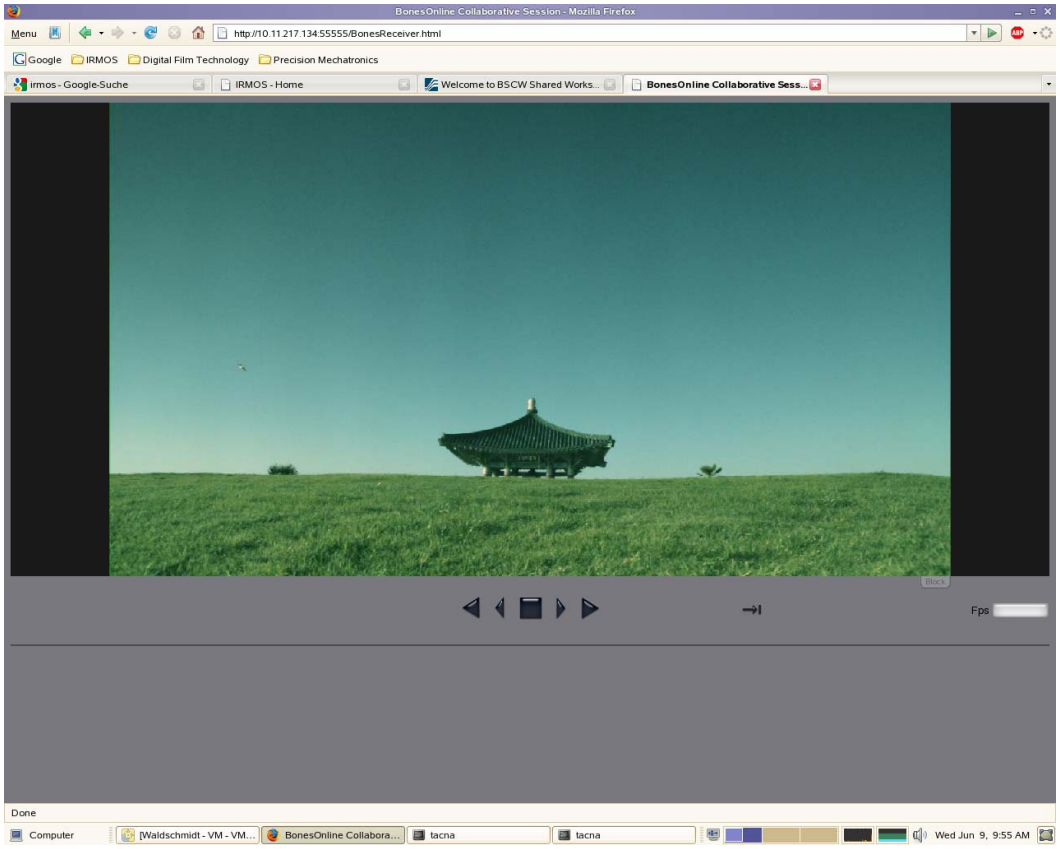
και 1 LBB) καθώς και μία VMU η οποία εκτελούσε τα απαραίτητα μέρη της πλατφόρμας, δηλαδή τον Εκτελεστή Ροών Εργασίας, τον Αξιολογητή και την Υπηρεσία Παρακολούθησης.

```
<process name="Post Production">
  <sequence>
    <invoke>
      <name>"Start IPC" </name>
      <partnerLink>"DFT.DFP.ipc"</partnerLink>
      <legacy>"true"</legacy>
      <operation>"/home/start"</operation>
      <error>
        <action>"retry"</action>
      </error>
    </invoke>
    <while>
      <condition> "ipcstate != ready" </condition>
      <componentID>"DFT.DFP.ipc </componentID>
      <empty/>
    </while>
     <flow>
       <invoke>
         <name>"Start IPU1" </name>
         <partnerLink>"DFT.DFP.ipu1"</partnerLink>
         <legacy>"true"</legacy>
         <operation>"/home/start"</operation>
         <error>
           <action>"retry"</action>
         </error>
       </invoke>
       <invoke>
        <name>"Start IPU2" </name>
                          ...
       </invoke>
                  ...
     </flow>
     <invoke>
       <name>"Start LB" </name>
       <partnerLink>"DFT.DFP.lb"</partnerLink>
       <legacy>"true"</legacy>
       <operation>"/home/start"</operation>
       <error>
         <action>"retry"</action>
       </error>
     </invoke>
```

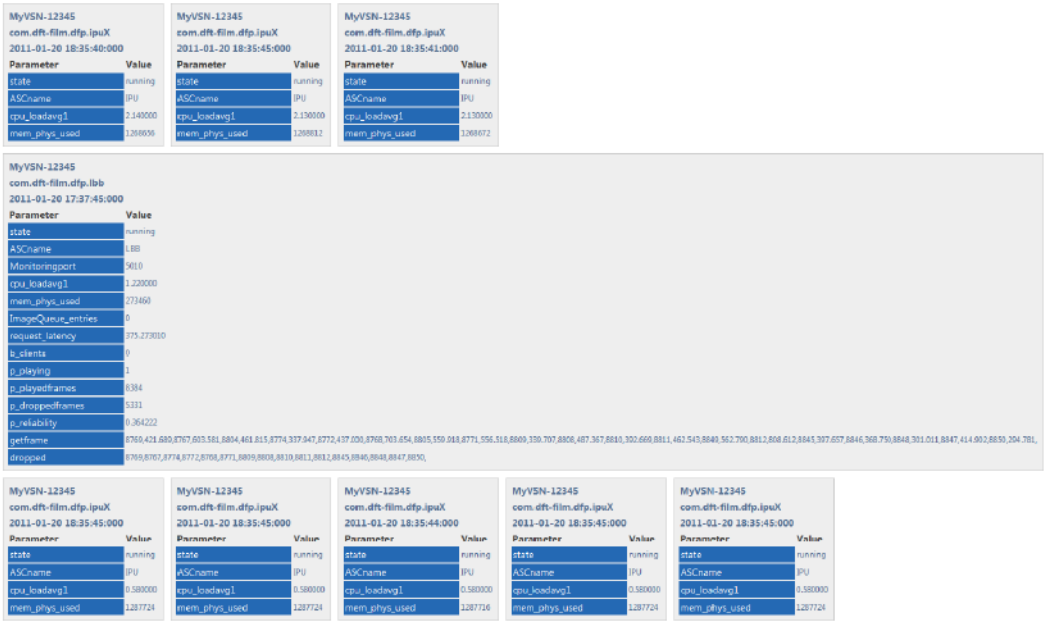
Σχήμα 7.9: *Αφηρημένη ροή εργασίας της εφαρμογής μεταπαραγωγής βίντεο*

Στα σχήματα 7.10, 7.11 και 7.12 παρουσιάζονται κάποιες εικόνες από την εκτέλεση της εφαρμογής μεταπαραγωγής video καθώς και από την πλατφόρμα.

Σχήμα 7.10: *Σταθμός ελέγχου της εφαρμογής μεταπαραγωγής βίντεο*



Σχήμα 7.11: *Η οθόνη του πελάτη της εφαρμογής*



Σχήμα 7.12: *Η υπηρεσία παρακολούθησης της πλατφόρμας*

7.4.3 Έλεγχος Ανοχής σε Σφάλματα

Για να ελεγχθεί η ανοχή του συστήματος σε σφάλματα εκτελέστηκε το πείραμα που περιγράφεται κατωτέρω. Κατά την εκτέλεση της εφαρμογής, απενεργοποιήθηκε ένα φυσικό μηχάνημα στο οποίο έτρεχε ένα IPU. Αυτό είχε ως αποτέλεσμα να αρχίσει η εφαρμογή να χάνει πλαίσια, γεγονός το οποίο αναγνωρίστηκε από τον Αξιολογητή ο οποίος ενημέρωσε τον Εκτελεστή της ροής. Η περιγραφή της ροής εργασίας όριζε ότι σε μία τέτοια κατάσταση θα πρέπει η εφαρμογή να αρχίσει να χρησιμοποιεί έναν άλλο αλγόριθμο συμπίεσης του βίντεο. Αυτό έγινε ώστε να μπορέσει η εφαρμογή να λειτουργήσει με ρυθμό 24 πλαισίων ανά δευτερόλεπτο ακόμα και αν η ποιότητα της εικόνας ήταν χειρότερη. Παράλληλα η δυσλειτουργία ενός μηχανήματος έγινε αντιληπτή και από τον πάροχο IaaS ο οποίος ξεκίνησε μία νέα εικονική μηχανή IPU σε ένα άλλο φυσικό μηχάνημα. Το γεγονός αυτό μεταφέρθηκε στον Εκτελεστή ροών, ο οποίος όφειλε να αρχικοποιήσει το IPU με τις κατάλληλες παραμέτρους. Μόλις το ASC ήταν λειτουργικό η πλατφόρμα όφειλε να το αναγνωρίσει και να ειδοποιήσει την εφαρμογή ότι θα πρέπει να επανέλθει σε φυσιολογική λειτουργία. Ο συνολικός χρόνος που χρειάστηκε για να όλη τη διαδικασία μπορεί να ορισθεί ως:

$$T_{total} = max((T_{evaluate}+T_{reconfigure}), (T_{ackfailure}+T_{bootup}+T_{notify}+T_{configurevmu}+ T_{startasc} + T_{reconfigure}))$$

Κατά την εκτέλεση του πειράματος το T_{total} μετρήθηκε κατά μέσο όρο ίσο με 149,1 sec το οποίο μπορεί οφείλεται στους χρόνους που φαίνονται στον πίνακα 7.4.

Όπως ήταν αναμενόμενο, φάνηκε πως οι μεγαλύτεροι χρόνοι είναι

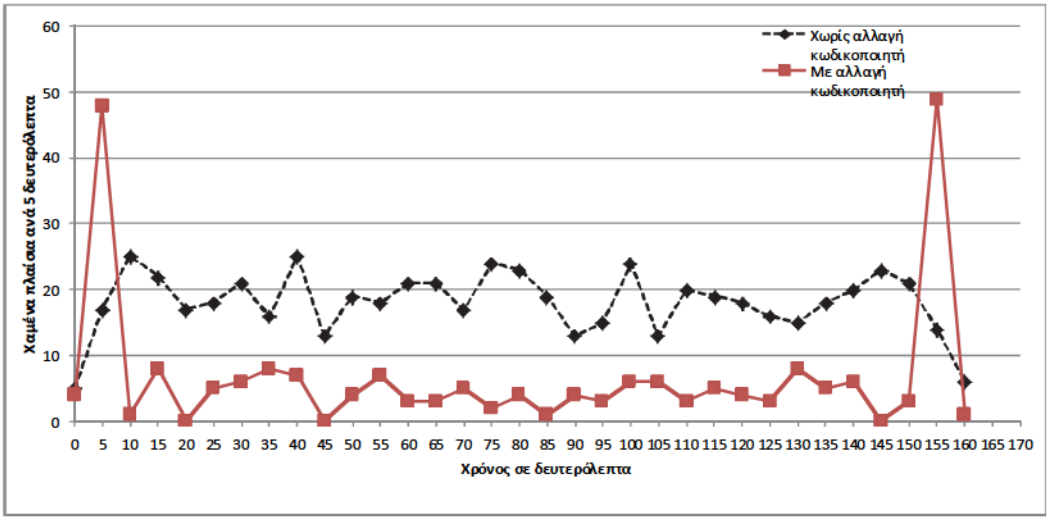
Πίνακας 7.4: *Μέσοι χρόνοι των διαφορετικών διαδικασιών κατά την επαναφορά του συστήματος μετά από σφάλμα.*

$T_{evaluate}$	47 ms
$T_{reconfigure}$	1,9 s
$T_{ackfailure}$	1,2 s
T_{bootup}	79,8 s
T_{notify}	41 ms
$T_{configurevmu}$	1,9 s
$T_{startasc}$	64,3 s
$T_{reconfigure}$	1,9 s

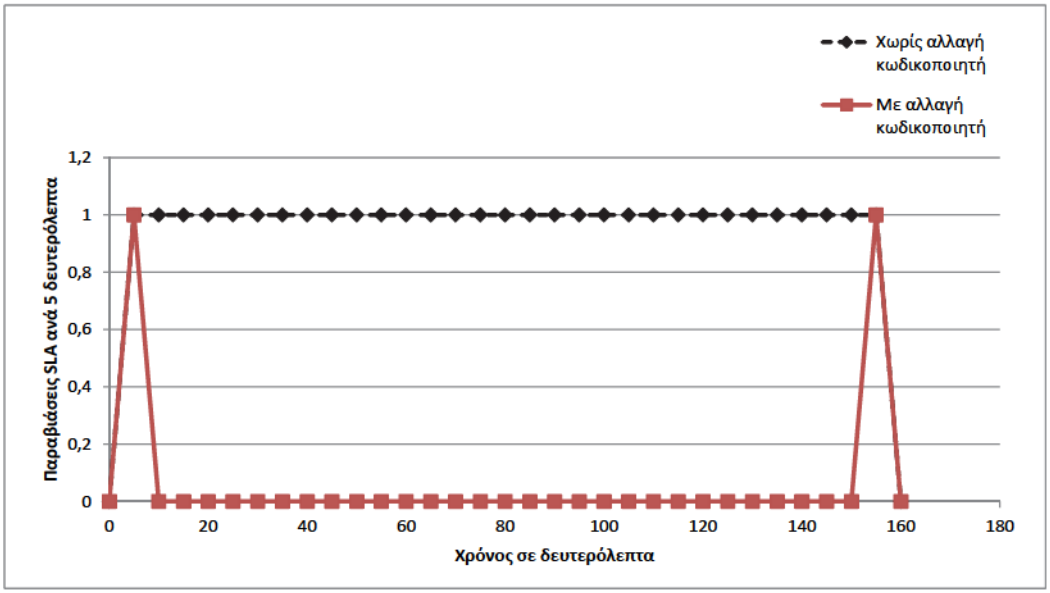
αυτοί που χρειάζονται α) για να ξεκινήσει μία εικονική μηχανή (79,8 sec) και β) για να ξεκινήσει μία υπηρεσία της εφαρμογής (64,3 sec). Έτσι είναι προφανές ότι έχουμε σημαντικά οφέλη από το γεγονός ότι ο εκτελεστής ροών εργασίας έχει την δυνατότητα να ρυθμίζει την εφαρμογή ώστε να χρησιμοποιεί άλλον κωδικοποιητή, έως ότου είναι διαθέσιμη μία νέα εικονική μηχανή για να αναπληρώσει αυτήν που έχει παρουσιάσει σφάλμα.

Για να ποσοτικοποιήσουμε τα οφέλη της αρχιτεκτονικής σχετικά με την δυνατότητα αντίδρασης σε γεγονότα εκτελέσαμε δύο διαφορετικές εκδοχές της ροής εργασίας. Και στις δύο περιπτώσεις ένα IPU τέθηκε εκτός λειτουργίας. Στην πρώτη περίπτωση όταν το σύστημα αντιλαμβανόταν το σφάλμα δεν εκτελούνταν αλλαγή στον κωδικοποιητή, ενώ στη δεύτερη, σε περίπτωση σφάλματος, γινόταν αλλαγή σε χαμηλότερης ποιότητας κωδικοποίηση μέχρις ότου γίνει διαθέσιμο ένα καινούριο IPU. Όπως έχει ήδη αναφερθεί, η χρήση χαμηλότερης ποιότητας κωδικοποίησης επιτρέπει στο σύστημα να λειτουργεί με ρυθμό 24 πλαισίων ανά δευτερόλεπτο. Σκοπός του πειράματος ήταν να μετρήσουμε τα χαμένα πλαίσια λόγω της απουσίας του IPU και τις παραβιάσεις του SLA που τα χαμένα πλαίσια δημιουργούσαν στις δύο περιπτώσεις. Και στις δύο περιπτώσεις το SLA όριζε ότι η εφαρμογή θα πρέπει να μην χάνει πάνω

από 10 πλαίσια ανά διάστημα 5 δευτερολέπτων. Τα αποτελέσματα φαίνονται στα Σχήματα 7.13 και 7.14 αντίστοιχα.



Σχήμα 7.13: *Σύγκριση αριθμού χαμένων πλαισίων με αλλαγή και χωρίς κωδικοποιητή*



Σχήμα 7.14: *Σύγκριση αριθμού παραβιάσεων του SLA με αλλαγή και χωρίς κωδικοποιητή*

Είναι προφανές ότι όταν προς αναπλήρωση μίας εικονικής μηχανής που παρουσίασε σφάλμα, η μόνη διορθωτική ενέργεια από τη μεριά του συστήματος είναι να ξεκινήσει μία καινούρια, η απόδοση της εφαρμογής πέφτει με αποτέλεσμα να χάνονται πολλά πλαίσια. Αυτό έχει ως επακόλουθο να παραβιάζεται διαρκώς το υπογραφέν SLA.

Έτσι, είναι προφανές ότι η δυνατότητα που δίνει ο μηχανισμός για πολλαπλές διορθωτικές κινήσεις, όπως είναι η αλλαγή του τρόπου λειτουργίας της εφαρμογής, κάνει το σύστημα πιο ευέλικτο και ανεκτικό σε σφάλματα. Επίσης οι χρόνοι αντίδρασης του μηχανισμού επιτρέπουν την γρήγορη αντίληψη του σφάλματος και την έγκαιρη λήψη των κατάλληλων μέτρων.

7.5 Αξιολόγηση μέσω Εφαρμογής Αναγνώρισης Συμπεριφοράς

Το προτεινόμενο ΣΔΡΕ αξιολογήθηκε και μέσω μίας εφαρμογής αναγνώρισης συμπεριφοράς από ακολουθίες βίντεο. Στην παρούσα ενότητα παρουσιάζεται η εφαρμογή και τα αποτελέσματα που προέκυψαν.

7.5.1 Περιγραφή και Αρχιτεκτονική της Εφαρμογής

Σκοπός της εφαρμογής είναι η αναγνώριση ροών εργασίας σε πραγματικό χρόνο σε βιομηχανικά περιβάλλοντα μέσω ανάλυσης ακολουθιών βίντεο. Έτσι, ένας βιομηχανικός χώρος παραγωγής αυτοκινήτων ελέγχεται συνεχώς μέσω καμερών. Το ληφθέν βίντεο αναλύεται ώστε να γίνει δυνατή η αναγνώριση της κίνησης των εργατών και κατ' επέκταση η αναγνώριση της ροής εργασίας. Έτσι, είναι δυνατόν να διασφαλιστεί η ποιότητα της διαδικασίας παραγωγής και η ασφάλεια των εργατών καθώς επίσης να εντοπιστούν λάθη ή να εξαχθούν πληροφορίες που θα βελτιώσουν την απόδοση της γραμμής παραγωγής.

Η εφαρμογή αποτελείται από τέσσερα διαφορετικά μέρη τα οποία έχουν αναπτυχθεί ως υπηρεσίες και μπορούν να δρουν ανεξάρτητα. Συγκεκριμένα η εφαρμογή αποτελείται από τις υπηρεσίες:

- Εξαγωγής χαρακτηριστικών (Feature Extraction - FE)
- Κατάτμησης ακολουθιών (Time Segmentation - TS)
- Ταξινόμησης εργασιών (Task Classification -TC)

- Αναγνώρισης ροής εργασίας μέσω γενετικού αλγορίθμου (Workflow Recognition)

Το γεγονός ότι η εφαρμογή έχει αναπτυχθεί μέσω μίας υπηρεσιοστρεφούς αρχιτεκτονικής την κάνει ιδιαιτέρως ευέλικτη. Παραδείγματος χάριν επιτρέπει την εύκολη εναλλαγή του αλγορίθμου εξαγωγής χαρακτηριστικών ή την εναλλαγή μεταξύ χρήσης νευρωνικού δικτύου Hopfield με την χρήση γενετικού αλγόριθμου για την αναγνώριση της ροής εργασίας. Παράλληλα επιτρέπει την ταυτόγχρονη χρήση πολλαπλών στιγμιοτύπων των υπηρεσιών προσφέροντας έτσι σημαντικές ικανότητες ελαστικότητας (elasticity).

Για την εξαγωγή των χαρακτηριστικών χρησιμοποιήθηκε η μέθοδος των ροπών Zernike. Οι ροπές Zernike τάξης p ορίζονται ως:

$$A_{pq} = \frac{p+1}{\pi} \int_0^1 \int_{-\pi}^{\pi} R_{pq}(r) e^{-jq\theta} f(r,\theta)\, r\, \mathrm{d}r\, \mathrm{d}\theta \tag{7.1}$$

όπου $r = \sqrt{x^2+y^2}, \theta = \tan^{-1}(y/x)$ και $-1 < x, y < 1$ και:

$$R_{pq}(r) = \sum_{s=0}^{\frac{p-q}{2}} (-1)^s \frac{(p-s)!}{s!\,(\frac{p+q}{2}-s)!\,(\frac{p-q}{2}-s)!}\, r^{p-2s} \tag{7.2}$$

όπου $p - q =$ άρτιος και $0 \leq q \leq p$.

Η ροπή μπορεί να θεωρηθεί ως μέτρο της πληροφορίας που εμπεριέχεται. Έτσι, ροπές μεγαλύτερης τάξης περιέχουν περισσότερες λεπτομέρειες και αναπαριστούν καλύτερα μία εικόνα. Από την άλλη μεριά οι ροπές μεγαλύτερης τάξης είναι δυσκολότερες να υπολογιστούν και άρα χρειάζονται μεγαλύτερη υπολογιστική ισχύ και χρόνο. Παραδείγματος χάριν,

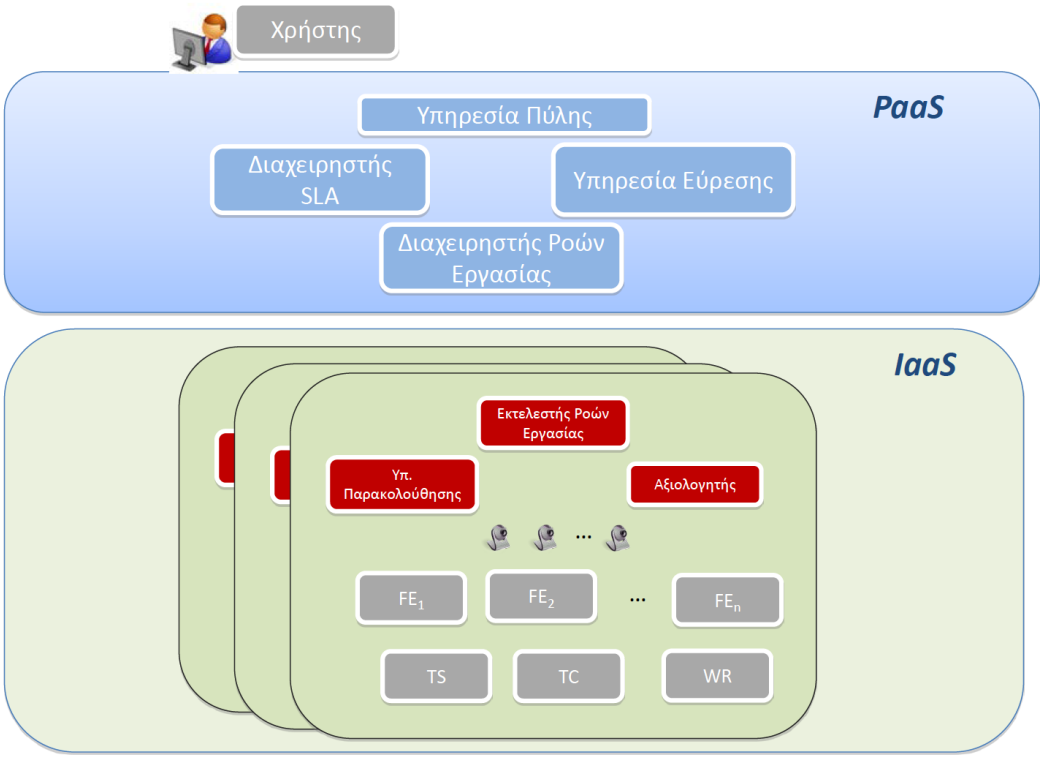
για ροπές μέχρι και 5ης τάξης θα πρέπει να υπολογιστούν δώδεκα ροπές (μέτρο και φάση) με αποτέλεσμα το διάνυσμα χαρακτηριστικών κάθε frame να είναι διάστασης 24. Το μέγεθος αυτό είναι εφικτό να υπολογιστεί σε πραγματικό χρόνο καθώς οι κάμερες λειτουργούν με ρυθμό 25 fps. Από την άλλη μεριά αν χρησιμοποιηθούν ροπές 20ής τάξης το διάνυσμα χαρακτηριστικών αποκτά μέγεθος 242. Σε αυτή την περίπτωση η εξαγωγή των χαρακτηριστικών είναι πιο αργή και ο ρυθμός πέφτει στο 1 fps.

Το γεγονός αυτό μαζί με την ανάγκη ύπαρξης πολλαπλών καμερών, ώστε να μπορεί να καλυφθεί επαρκώς ένα βιομηχανικό περιβάλλον το οποίο εμφανίζει εμπόδια και επικαλύψεις, οδηγεί στην ανάγκη πολλαπλών ταυτόχρονων υπηρεσιών εξαγωγής χαρακτηριστικών, κάθε μία από τις οποίες υπολογίζει τις ροπές μίας τάξης.

Έτσι η τελική εφαρμογή μπορεί να περιγραφεί ως μία ροή εργασίας την οποία καλείται να εκτελέσει το ΣΔΡΕ. Παράλληλα, ανάλογα με τις απαιτήσεις για την απόδοση του συστήματος μπορούν να χρησιμοποιηθούν περισσότερες ή λιγότερες υπηρεσίες με το ανάλογο κόστος σε πόρους. Η αρχιτεκτονική της εφαρμογής φαίνεται στο σχήμα 7.15.

7.5.2 Αποτελέσματα

Για το πείραμα χρησιμοποιήθηκαν 4 υπολογιστές P4 - 3 GHz με 1 GB μνήμης RAM οι οποίοι έτρεχαν ως λειτουργικό το Fedora Core 9 με πυρήνα 2.6.27 και στους οποίους ήταν εγκατεστημένες οι υπηρεσίες της πλατφόρμας ενώ 4 υπολογιστές Athlon 62 2 * 2,4 GHz με 4 GB μνήμης RAM και 6 υπολογιστές Core2 Quad Q9400 - 4 * 2,66GHz με 8 GB μνήμης



Σχήμα 7.15: *Αρχιτεκτονική εφαρμογής αναγνώρισης συμπεριφοράς*

RAM χρησιμοποιήθηκαν για να φιλοξενήσουν τα ASC με τις υπηρεσίες της εφαρμογής αναγνώρισης συμπεριφοράς σε βιομηχανικά περιβάλλοντα.

Για την αξιολόγηση της υπηρεσιοστρεφούς προσέγγισης η εφαρμογή συγκρίθηκε με μία πανομοιότυπη εφαρμογή με τη διαφορά όμως ότι όλες οι υπηρεσίες προσφέρονται κεντρικοποιημένα από μία εφαρμογή. Για τα πειράματα χρησιμοποιήθηκαν ροπές Zernike 5^{ης} και 20^{ής} τάξης που οδηγούν αντίστοιχα σε διανύσματα διάστασης 24 και 242. Τα αποτελέσματα φαίνονται στο πίνακα 7.5.

Πίνακας 7.5: *Σύγκριση κεντρικοποιημένης και υπηρεσιοστρεφούς προσέγγισης της εφαρμογής*

	Ροπές μέχρι 5ης τάξης	Ροπές μέχρι 20ης τάξης
Κεντρικοποιημένη	25 fps	1 fps
Υπηρεσιοστρεφής	25 fps	25 fps

Είναι εμφανές ότι η κεντρικοποιημένη προσέγγιση λειτουργεί ικανοποιητικά για ροπές χαμηλής τάξης, αλλά αποτυγχάνει πλήρως για ροπές 20^{ής} τάξης. Αυτό οφείλεται στο γεγονός ότι δεν είναι δυνατόν να παρασχεθούν περισσότεροι πόροι για την επεξεργασία. Αντίθετα στην υπηρεσιοστρεφή προσέγγιση και χρησιμοποιώντας τις εγγυήσεις που παρέχει η πλατφόρμα, η εφαρμογή λειτουργεί το ίδιο ικανοποιητικά τόσο σε χαμηλές όσο και σε υψηλές ροπές.

Στη συνέχεια υπολογίστηκε η ανάκληση και η ακρίβεια ώστε να ποσοτικοποιηθεί η αποτελεσματικότητα της εφαρμογής στην αναγνώριση ροών εργασίας. Τα αποτελέσματα φαίνονται στον πίνακα 7.6.

Φαίνεται ότι η υπηρεσιοστρεφής προσέγγιση παρουσιάζει αυξημένες επιδόσεις της τάξης του 14% σε σχέση με την κεντρικοποιημένη. Έτσι, είναι ασφαλές να υποστηριχθεί ότι η προτεινόμενη λύση παρέχει τη δυνατότητα στην

Πίνακας 7.6: *Ποσοστά επιτυχούς αναγνώρισης στην μέγιστη δυνατή λειτουργία των δύο προσεγγίσεων*

	Ροπές < $5^{ης}$ τάξης (κεντρικοποιημένη)	Ροπές < $20^{ής}$ τάξης (υπηρεσιοστρεφής)
Ανάκληση	83, 2%	97, 3%
Ακρίβεια	82, 1%	96, 8%

εφαρμογή αναγνώρισης ροών εργασίας να επιτύχει υψηλά ποσοστά ακρίβειας χωρίς να είναι απαραίτητη η χρήση εξειδικευμένου υλισμικού. Παράλληλα, μέσω των δυνατοτήτων που παρέχει η εικονικοποίηση είναι δυνατόν να ελέγχεται η χρήση των πόρων και να προσαρμόζεται στις δεδομένες ανάγκες κατά τη διάρκεια της παραγωγικής διαδικασίας.

7.6 Σύνοψη του Κεφαλαίου

Σε αυτή την ενότητα παρουσιάστηκε το προταθέν Σύστημα Διαχείρισης Ροών Εργασίας και επιβεβαιώθηκε η λειτουργία του. Ο μηχανισμός είναι πολυεπίπεδος και βρίσκεται τόσο στο επίπεδο της πλατφόρμας όσο και μέσα στο εικονικοποιημένο περιβάλλον. Λαμβάνοντας υπ' όψη ότι οι σύγχρονες εφαρμογές τείνουν να υιοθετήσουν την υπηρεσιοστρεφή προσέγγιση και άρα να αποτελούνται από πολλά ανεξάρτητα δομικά μέρη που συνεργάζονται για να παρέχουν μία υπηρεσία, πρωταρχικός σκοπός του προταθέντος ΣΔΡΕ είναι να διαχειρίζεται τα μέρη αυτά, να επιβλέπει την εκτέλεση της εφαρμογής και να αντιδρά σε γεγονότα λαμβάνοντας διαρκώς υπ' όψη τις απαιτήσεις πραγματικού χρόνου της εφαρμογής. Για το λόγο αυτό, οι απαιτήσεις ποιότητας υπηρεσίας είναι στον πυρήνα του προταθέντος ΣΔΡΕ. Παράλληλα, με την πολυεπίπεδη δομή που προτείνεται, ελαχιστοποιούνται οι αλληλεπιδράσεις του ΣΔΡΕ με τις υπηρεσίες της πλατφόρμας, δίνοντας την δυνατότητα για την έγκαιρη αναγνώριση συμβάντων και τη λήψη των ανάλογων δράσεων. Επίσης, λόγω της ύπαρξης ενός μέρους του ΣΔΡΕ στο εικονικοποιημένο περιβάλλον, δίνεται η δυνατότητα να θεωρηθεί μέρος της εφαρμογής και να μετρηθεί η απόδοσή του καθώς και ο φόρτος που το ΣΔΡΕ επιφέρει στην εκτέλεση της εφαρμογής.

Η επιβεβαίωση της ορθής λειτουργίας του ΣΔΡΕ έγινε μέσω μίας σειράς πειραμάτων τα οποία έδειξαν ότι ο μηχανισμός έχει σημαντικά οφέλη κατά την εκτέλεση μίας εφαρμογής με απαιτήσεις πραγματικού χρόνου. Ο μηχανισμός είναι αρκετά «ελαφρύς» σε σχέση με μία απλή εφαρμογή κλήσης υπηρεσιών, ενώ δύναται να αναγνωρίσει έγκαιρα καταστάσεις και να ενεργήσει ανάλογα.

8 Επέκταση του Μηχανισμού Επίβλεψης

8.1 Εισαγωγή στους Μηχανισμούς Επίβλεψης

Κατά την ανάπτυξη και χρήση του Συστήματος Διαχείρισης Ροών Εργασίας έγινε σαφές ότι σημαντικό ρόλο στην λειτουργία του συστήματος είχε ο μηχανισμός επίβλεψης (monitoring) της πλατφόρμας.

Παραδοσιακά, κάθε κατανεμημένο σύστημα χρειάζεται να περιλαμβάνει μηχανισμούς επίβλεψης ούτως ώστε να μπορεί να ελέγχει διαρκώς την κατάστασή του και να ενημερώνεται για την απόδοσή του. Αυτό γίνεται ιδιαιτέρως επιτακτικό στα Νέφη όπου υπάρχουν σε ισχύ Συμβόλαια Επιπέδου Υπηρεσίας (SLA) και στα οποία πληροφορίες χρεώσεων και παραβιάσεων των SLA είναι σημαντικές τόσο για τον πάροχο όσο και για τον πελάτη (Keller & Ludwig, 2003).

Επιπλέον, μια υποδομή Νέφους πρέπει να είναι σε θέση να διαχειριστεί αποτελεσματικά και διαφανώς την ανοχή σε σφάλματα και τον χρονοπρογραμματισμό των πόρων, να μπορεί να εκτελέσει τη μετανάστευση υπηρεσιών και δεδομένων και να μπορεί να προβλέψει και να χειριστεί

πολλαπλούς φόρτους εργασίας προκειμένου να είναι σε θέση να εξασφαλίσει συγκεκριμένη ποιότητα υπηρεσιών (QoS) προς τους τελικούς χρήστες. Άλλες χρήσεις των πληροφοριών επίβλεψης αποτελούν η καταγραφή της συμπεριφοράς των χρηστών, η καταγραφή πληροφοριών που χρειάζονται για νομικούς λόγους (auditing), η απάντηση σε ad hoc ερωτήματα, κ.α.

Λαμβάνοντας υπ' όψη το γεγονός ότι τα Νέφη είναι μεγάλα σε έκταση, καθώς και το γεγονός ότι διαφέρουν από τα Πλέγματα από τα οποία έχουν προέλθει (Foster, Zhao, Raicu, & Lu, 2008), γίνεται προφανές ότι πρέπει να ακολουθηθεί μία προσέγγιση που στοχεύει να ικανοποιήσει τις ιδιαιτερότητες των Νεφών. Πέρα από το να παρέχει απλές πληροφορίες, ένα σύγχρονο σύστημα επιτήρησης θα πρέπει να παρέχει δυνατότητες συνάθροισης των μετρικών χαμηλού επιπέδου, έτσι ώστε να δημιουργούνται γεγονότα υψηλού επιπέδου. Επίσης, θα πρέπει να δύναται να λαμβάνει αποφάσεις ούτως ώστε να κάνει ευκολότερη τη διαχείριση ενός Νέφους. Κάτι τέτοιο βρίσκεται σε πλήρη συνάφεια με τις τελευταίες τάσεις στις επιχειρήσεις όπου οι οργανωτικοί κανόνες και τα σύστημα επεξεργασίας σύνθετων γεγονότων παρέχουν ακριβή πληροφόρηση στους διαχειριστές των συστημάτων και τους πελάτες τους (K.-U. Schmidt, Anicic, & Stühmer, 2008).

Λόγω του γεγονότος αυτού κρίθηκε σκόπιμο να υλοποιηθεί ένας νέος μηχανισμός επίβλεψης,ο οποίος θα ήταν πιο αποδοτικός και θα παρείχε περισσότερες δυνατότητες. Στο κεφάλαιο αυτό παρουσιάζεται αυτός ο μηχανισμός επίβλεψης.

8.2 Απαιτήσεις Συστημάτων Επίβλεψης

Η βασική λειτουργία ενός συστήματος ελέγχου είναι να συλλεχθούν οι πληροφορίες σχετικά με την κατάσταση των πόρων ενδιαφέροντος και να διανεμηθούν στους μηχανισμούς που λαμβάνουν τις αποφάσεις. Η φύση του οικοσυστήματος των Νεφών επιβάλλει περαιτέρω απαιτήσεις που πρέπει να ικανοποιούνται ώστε να είναι ένα σύστημα επίβλεψης αποτελεσματικό. Ένα μη λεπτομερές σύνολο απαιτήσεων για τον μηχανισμό επίβλεψης είναι το ακόλουθο:

- *Κλιμάκωση.* Να μπορεί να κλιμακώνεται σε μεγάλα μεγέθη, τυπικά των περιβαλλόντων Νεφών.

- *Μη-επεμβατικότητα.* Να έχει τη λιγότερο δυνατή επεμβατικότητα στο σύστημα που ελέγχει και να επιφέρει την ελάχιστη δυνατή επιβάρυνση.

- *Ευρωστία.* Να μπορεί να αντεπεξέλθει σε σφάλματα.

- *Επεκτασιμότητα.* Να μπορεί να δεχθεί νέους παρόχους πληροφορίας χωρίς να χρειάζεται να γίνει επανέναρξη του μηχανισμού.

- *Δυνατότητες ομοσπονδίας (federation).* Να μπορεί να υποστηρίξει σενάρια ομοσπονδίας, κατά τα οποία δύο Νέφη συνενώνονται για να παρέχουν κοινές υπηρεσίες.

- *Ασφάλεια.* Να εφαρμόζει κατάλληλες πρακτικές ώστε τα δεδομένα παρακολούθησης να είναι προσπελάσιμα μόνο από εξουσιοδοτημένους πελάτες.

Η πιο απλοϊκή σκέψη είναι τα δεδομένα παρατήρησης χαμηλού επιπέδου να μεταδίδονται ως έχουν στους τελικούς καταναλωτές. Μία τέτοια λύση παρουσιάζει προβλήματα κλιμάκωσης σε περιβάλλοντα Νεφών ιδιαίτερα σε περιπτώσεις όπου τα δεδομένα θα πρέπει να διασχίσουν μεγάλες γεωγραφικές αποστάσεις. Στον αντίποδα μπορούν να χρησιμοποιηθούν τεχνικές συνάθροισης ή διήθησης ώστε να επιτευχθεί μείωση του όγκου των δεδομένων, καθώς και τεχνικές οι οποίες επιτυγχάνουν την επίβλεψη συναρτήσεων δεδομένων (Sharfman, Schuster, & Keren, 2007), (Giatrakos, Deligiannakis, Garofalakis, Sharfman, & Schuster, 2012).

8.3 Βιβλιογραφική Ανασκόπηση των Μηχανισμών Επίβλεψης

Σε αυτήν την παράγραφο παρουσιάζονται συνοπτικά κάποιες προτάσεις μηχανισμών επίβλεψης, ώστε να δομηθεί μία εικόνα των τεχνικών που έχουν χρησιμοποιηθεί σε παρόμοιους τομείς. Για τον λόγο αυτό γίνεται διαχωρισμός σε τρεις κατηγορίες και συγκεκριμένα: γενικές λύσεις, λύσεις από τον τομέα των Πλεγμάτων και λύσεις από τον τομέα των Νεφών.

8.3.1 Γενικές Λύσεις Επίβλεψης

Το Nagios (Barth, 2008) είναι μία λύση ανοιχτού κώδικα η οποία χρησιμοποιείται για την παρακολούθηση υπηρεσιών και υπολογιστών. Ο χρήστης έχει την δυνατότητα να ορίσει παραμέτρους, όπως τον ρυθμό μετάδοσης των δεδομένων και τον αριθμό προσπαθειών σε περιπτώσεις

σφάλματος. Κάθε υπηρεσία ορίζεται πως αντιστοιχεί σε συγκεκριμένο τερματικό. Επίσης, δίνεται η δυνατότητα να ορισθούν εξαρτήσεις μεταξύ υπηρεσιών.

Το Ganglia (Massie, Chun, & Culler, 2004) είναι ένα κατανεμημένο σύστημα παρακολούθησης που στοχεύει στα υπολογιστικά συστήματα υψηλής απόδοσης, όπως είναι οι συστάδες υπολογιστών clusters και τα Πλέγματα. Στο Ganglia κάθε κόμβος επιβλέπει τον εαυτό του και προωθεί τις πληροφορίες σε όλους τους άλλους κόμβους ή σε ένα υποσύνολο αυτών, αν ακολουθείται μία ιεραρχική δομή. Το σύστημα χρησιμοποιεί XML για την περιγραφή των δεδομένων, XDR για την μεταφορά των δεδομένων και το Round Robin Database tool (RRDtool) για την αποθήκευσή τους.

Το MonALISA (Legrand κ. συν., 2009) είναι μία πλατφόρμα παρακολούθησης που βασίζεται στο Jini (Waldo, 1999) και το οποίο είναι μία «συλλογή αυτόνομων και αυτο-περιγραφικών υποσυστημάτων που βασίζονται σε τεχνολογίες πρακτόρων και τα οποία λειτουργούν ως δυναμικές υπηρεσίες». Οι υπηρεσίες του MonALISA μπορούν να ανακαλύπτουν η μία την άλλη καθώς και να ανακαλύπτονται από πελάτες με χρήση των υπηρεσιών Lookup Discovery. Το σύστημα περιλαμβάνει έναν εξυπηρετητή ανά τοποθεσία. Κάθε σταθμός εξυπηρετεί πολλαπλές υπηρεσίες και είναι υπεύθυνος για την χρονοδρομολόγησή τους καθώς και για να λαμβάνει από κάθε κόμβο τα στοιχεία επίβλεψής του. Η αρχιτεκτονική του MonALISA είναι ελαστική, αλλά εγείρονται ερωτήματα ως προς την κλιμάκωσή του, λόγω χρήσης του Jini.

8.3.2 Λύσεις Επίβλεψης Περιβαλλόντων Πλέγματος

Το Globus Toolkit (Foster & Kesselman, 1997) είναι ένα από τα πιο γνωστά πλαίσια για την δημιουργία υπολογιστικών πλεγμάτων. Το Globus Toolkit Monitoring and Discovery System (MDS4) παρέχει μηχανισμούς για την επίβλεψη και εύρεση πόρων και υπηρεσιών σε Πλέγματα. Το MDS χρησιμοποιεί τις προδιαγραφές WSRF και WS-N για να περιγράψει τις πηγές πληροφόρησης και να διαχειριστεί τις συνδρομές. Η υπηρεσία Index του MDS έχει την δυνατότητα να συλλέξει πληροφορίες επίβλεψης από οποιαδήποτε πηγή αρκεί αυτή να χρησιμοποιεί την κατάλληλη μορφή για την περιγραφή των πληροφοριών μέσω XML. Στη συνέχεια έχει την δυνατότητα να δημοσιεύσει τις πληροφορίες αυτές σαν διαδικτυακούς πόρους. Το MDS δεν περιέχει το ίδιο πηγές πληροφοριών αλλά εξαρτάται από εξωτερικές πηγές όπως το Ganglia και το Nagios.

Το GridICE (Andreozzi κ. συν., 2005) αναπτύχθηκε στα πλαίσια του έργου DataTag (Martin-Flatin & Primet, 2005) για να διευκολύνει τους διαχειριστές συστημάτων Πλέγματος. Το GridICE ακολουθεί μία κεντρικοποιημένη προσέγγιση όπου ένας κεντρικός εξυπηρετητής ζητάει περιοδικά δεδομένα επιτήρησης από πολλαπλούς κόμβους. Ο κεντρικός εξυπηρετητής είναι βασισμένος στο Nagios και έχει επεκταθεί με διάφορες προσθήκες, όπως ένα ειδικό plugin που μπορεί να ζητάει δεδομένα από υπηρεσίες Globus μέσω του MDS. Οι πληροφορίες αποθηκεύονται σε μία βάση δεδομένων και μπορούν να χρησιμοποιηθούν για να εξαχθούν διάφορα στατιστικά μεγέθη. Η κεντρικοποιημένη προσέγγιση του συστήματος είναι ανεπαρκής για μεγάλης κλίμακας συστήματα και οφείλουν να γίνουν

επεκτάσεις για να μπορούν να τα υποστηρίξουν. Μία τέτοια επέκταση θα ήταν η χρήση πολλαπλών κεντρικών εξυπηρετητών.

Ένα γενικό σύστημα επίβλεψης περιγράφεται στο (Tierney κ. συν., 2002). Η βασική ιδέα είναι ότι οι παραγωγοί και καταναλωτές πληροφοριών επιτήρησης μπορούν να ανακαλύψουν ο ένας τον άλλο μέσω μία υπηρεσίας καταλόγου. Στη συνέχεια οι πληροφορίες στέλνονται απεύθείας από τους παραγωγούς στους καταναλωτές. Η πρόταση υποστηρίζει τόσο ένα μοντέλο δημοσίευσης/εγγραφής (publish/subscribe όσο και ένα μοντέλο ερώτησης/απάντησης (query/response). Αν υπάρχει ανάγκη για συνάθροιση πληροφοριών τότε μία υπηρεσία θα πρέπει να είναι καταναλωτής βασικών γεγονότων και παραγωγός σύνθετων γεγονότων.

8.3.3 Λύσεις Επίβλεψης Περιβαλλόντων Νέφους

Η Amazon παρέχει στους πελάτες της το CloudWatch (*Amazon CloudWatch*, α.χ.) ως μηχανισμό επίβλεψης. Το CloudWatch μπορεί να χρησιμοποιηθεί είτε μέσω μιας διεπαφής διαδικτύου είτε μέσω μίας προγραμματιστικής διεπαφής. Το CloudWatch μπορεί να παρέχει υπηρεσίες παρακολούθησης και για το Elastic Block Storage (EBS) αλλά όχι για την υπηρεσία αποθήκευσης S3. Για το EBS παρέχονται οχτώ προκαθορισμένες μετρικές παρακολούθησης με συχνότητα πέντε λεπτών και πιό συγκεκριμένα παρέχονται τα VolumeReadBytes, VolumeWriteBytes, VolumeReadOps, VolumeWriteOps, VolumeTotalReadTime, VolumeTotalWriteTime, VolumeIdleTime και VolumeQueueLength.

Μία άλλη πρόταση για επίβλεψη υπηρεσιών της Amazon είναι το

CloudStatus (*CloudStatus*, α.χ.). Το CloudStatus είναι μία πλατφόρμα ανοιχτού λογισμικού και βασίζεται στο Hyperic HQ. Εκτός από υπηρεσίες της Amazon έχει την δυνατότητα να επιβλέπει και υπηρεσίες της Google App Engine.

Το Windows Azure (*Windows Azure*, α.χ.) παρέχει και αυτό το δικό του μηχανισμό για επίβλεψη που ονομάζεται Azure Diagnostic Manager. Ο μηχανισμός επιτρέπει τη συλλογή πληροφοριών επίβλεψης από πολλαπλές πηγές και μπορεί να χρησιμοποιηθεί για την ανάλυση της επίδοσης της υπηρεσίας. Ο χρήστης έχει επίσης την δυνατότητα να ορίζει δικές του μετρικές απόδοσης οι οποίες μπορούν να χρησιμοποιηθούν για να βρεθούν σημεία που προκαλούν συμφόρηση στην απόδοση της υπηρεσίας.

Δύο συστήματα που έχουν σχεδιαστεί συγκεκριμένα για περιβάλλοντα νεφών είναι το Compuware Gomez (*Compuware Gomez*, α.χ.) και το LogicMonitor (*LogicMonitor*, α.χ.). Και τα δύο είναι πολύ εύκολα στη χρήση αλλά έχουν ως στόχο τον διαχειριστή της υποδομής και δεν εξυπηρετούν τις ανάγκες άλλων μηχανισμών της πλατφόρμας.

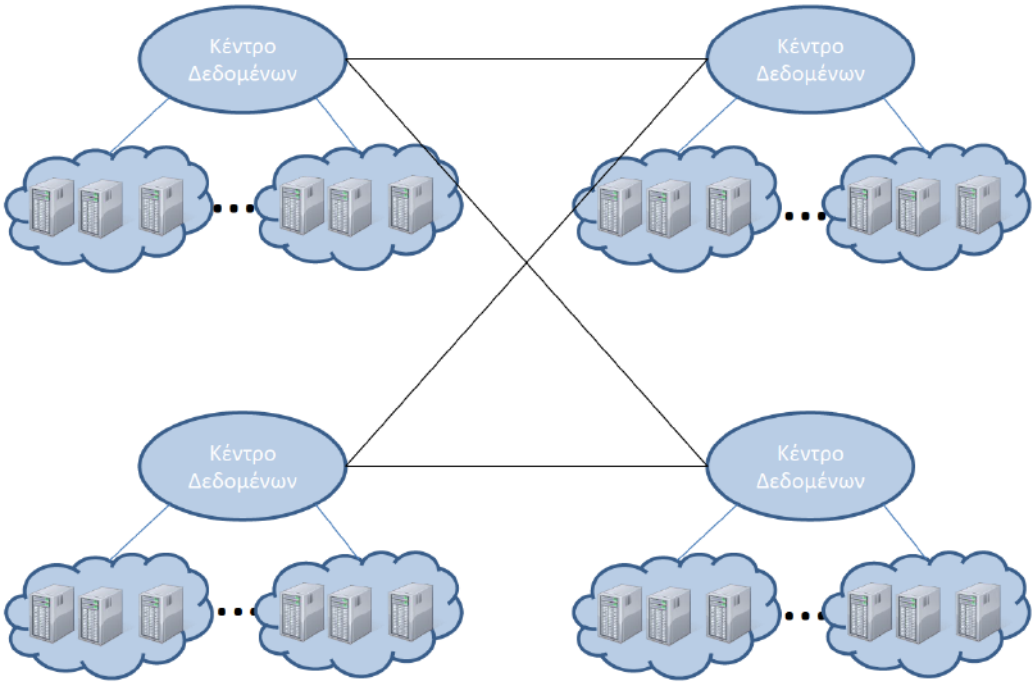
Το Lattice (Clayman κ. συν., 2010) είναι ένα σύστημα ανοιχτού λογισμικού που έχει ως κύριο χαρακτηριστικό την δημιουργία ενός βέλτιστου δικτύου μεταφοράς δεδομένων επίβλεψης με σκοπό τη χρήση όσο το δυνατόν λιγότερων δικτυακών πόρων και χρησιμοποιήθηκε διεξοδικά στα πλαίσια του ευρωπαϊκού προγράμματος Reservoir (Clayman, Toffetti, Galis, & Chapman, 2012).

8.4 Σχεδίαση και Υλοποίηση του Μηχανισμού Επίβλεψης

Βασικό στοιχείο του προς υλοποίηση μηχανισμού επίβλεψης ήταν η δυνατότητα του να χρησιμοποιηθεί σε διαφορετικές πλατφόρμες. Για την επιβεβαίωσή του χρησιμοποιήθηκε ως αντικαταστάτης του μηχανισμού επίβλεψης που χρησιμοποιήθηκε στην πλατφόρμα IRMOS. Παράλληλα έγινε ενσωμάτωσή του στην πλατφόρμα του ευρωπαϊκού έργου VISION Cloud (Kolodner κ. συν., 2011).

8.4.1 Σύντομη Παρουσίαση της Πλατφόρμας VISION Cloud

Βασικός στόχος του VISION Cloud είναι να παρέχει υποστήριξη σε εφαρμογές που κάνουν χρήση μεγάλου όγκου δεδομένων data-intesive προσφέροντας εξελιγμένες υπηρεσίες αποθήκευσης. Η φυσική υποδομή που λαμβάνεται υπ'οψη βασίζεται σε υλισμικό καθημερινής χρήσης, ακολουθώντας την τάση που κυριαρχεί στους τομείς των εφαρμογών που βασίζονται σε μεγάλο όγκο δεδομένων καθώς και στον τομέα των Bid Data analytics (Ghemawat, Gobioff, & Leung, 2003), (Kouzes, Anderson, Elbert, Gorton, & Gracio, 2009). Έτσι, η φυσική υποδομή (Σχήμα 8.1) αποτελείται από ένα δίκτυο κέντρων δεδομένων που ενώνονται με μία μισθωμένη γραμμή. Κάθε κέντρο δεδομένων αποτελείται από ένα ή περισσότερα cluster που έχουν υπολογιστικούς, αποθηκευτικούς και δικτυακούς πόρους. Κάθε κόμβος της υποδομής μπορεί να έχει 12 με 24 σκληρούς δίσκους (π.χ. δίσκους των 2TB). Η αρχιτεκτονική του συστήματος θα πρέπει να υποστηρίζει δεκάδες κέντρα



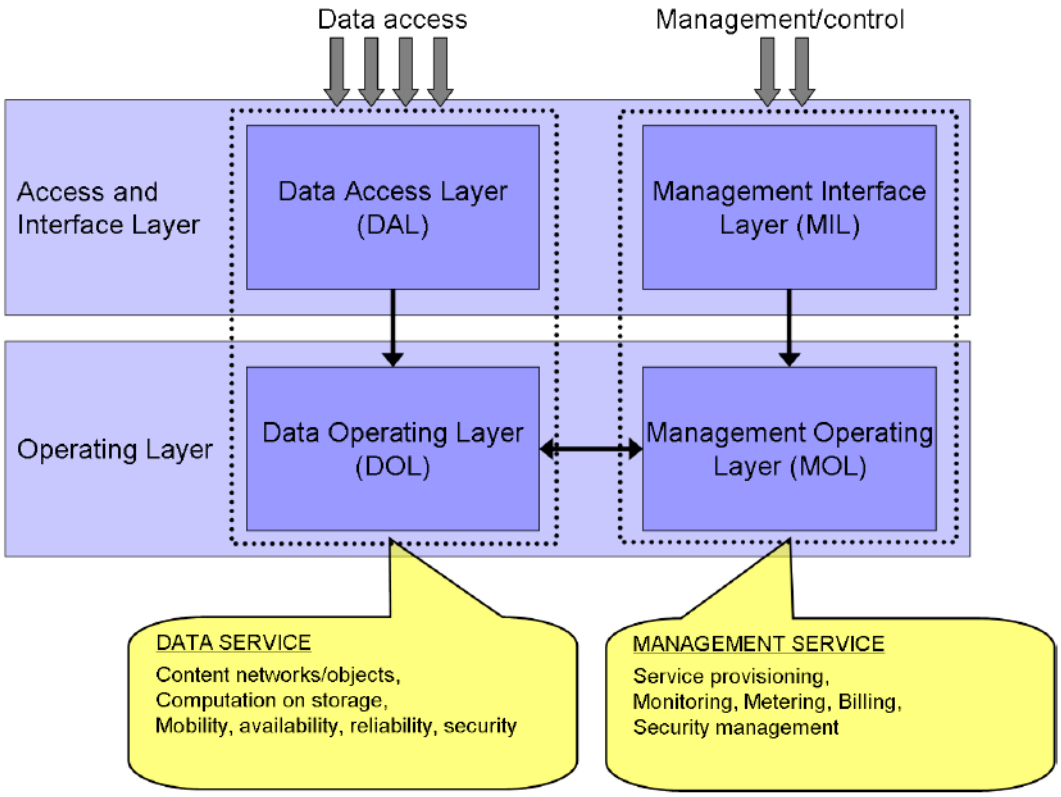
Σχήμα 8.1: *Μοντέλο της φυσική υποδομής*

δεδομένων κάθε ένα από τα οποία μπορεί να περιέχει εκατοντάδες cluster αποθήκευσης, όπου κάθε cluster μπορεί να έχει μερικές εκατοντάδες κόμβους.

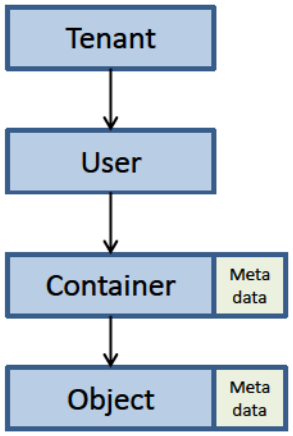
Η αρχιτεκτονική του VISION Cloud μπορεί να χωριστεί σε δύο επίπεδα, ένα το οποίο ασχολείται με ενέργειες πάνω στα δεδομένα, όπως δημιουργία και ανάκτηση αντικειμένων δεδομένων και ένα που ασχολείται με τις ενέργειες διαχείρισης της πλατφόρμας, όπως την διαπραγμάτευση SLA ή την λήψη αποφάσεων για την τοποθέτηση νέων container δεδομένων (Kolodner κ. συν., 2011). Ο διαχωρισμός αυτός οφείλεται στην διαφορά που έχουν τα Νέφη αποθήκευσης από τα υπολογιστικά Νέφη. Στα πρώτα οι υπηρεσίες διαχείρισης του Νέφους χρησιμοποιούνται για να παρέχουν υπολογιστικούς πόρους σε εξωτερικής οντότητες, όπως ορίζει ο πάροχος της υπηρεσίας. Στα αποθηκευτικά Νέφη, από τη άλλη μεριά, διαφορετικοί καταναλωτές μπορεί

να αλληλεπιδρούν με το Νέφος για να ανακτήσουν τα δεδομένα (Gogouvitis, Kousiouris, Vafiadis, Kolodner, & Kyriazis, 2012). Η υπηρεσία δεδομένων και η υπηρεσία διαχείρισης έχουν σχεδιαστεί να είναι διακριτές και ανεξάρτητες ώστε να πραγματοποιούν αυτόν τον διαχωρισμό. Η εξωτερική διεπαφή του συστήματος βασίζεται στο Cloud Data Management Interface (CDMI) (*Cloud Data Management Interface (CDMI)*, α.χ.) όπως έχει προταθεί από την Storage Networking Industry Association (SNIA), ενώ έχουν γίνει επεκτάσεις όπου υπήρχε ανάγκη. Η χρήση μίας πρότυπης διεπαφής εξασφαλίζει ότι η πλατφόρμα θα μπορεί να προσπελασθεί με ένα γνωστό τρόπο βοηθώντας έτσι την διαλειτουργικότητα και επιτρέποντας να λάβουν χώρα σενάρια ομοσπονδίας.

Το μοντέλο που ακολουθείται από την πλατφόρμα για τα δεδομένα επεκτείνει το μοντέλο που προτείνεται από το CDMI. Το βασικό συστατικό είναι το αντικείμενο δεδομένων (data object) το οποίο περιέχει δεδομένα οποιοδήποτε μεγέθους και τύπου. Τα αντικείμενα δεδομένων «ομαδοποιούνται» σε container τα οποία είναι η μονάδα τοποθέτησης, δηλαδή όλα τα αντικείμενα ενός container θα βρίσκονται στο ίδιο cluster. Τα αντικείμενα και οι containers έχουν μετα-δεδομένα τα οποία περιγράφουν τα δεδομένα. Τα μετα-δεδομένα μπορεί να είναι είτε ορισμένα από τον χρήστη είτε από το σύστημα. Ένας «ένοικος» (tenant) είναι η οντότητα η οποία εγγράφεται στις υπηρεσίες της πλατφόρμας και τιμολογείται ανάλογα. Κάθε ένοικος έχει χρήστες οι οποίοι χρησιμοποιούν τις υπηρεσίες της πλατφόρμας και στους οποίους ανήκουν τα αντικείμενα.



Σχήμα 8.2: *Τα διαφορετικά επίπεδα της πλατφόρμας*



Σχήμα 8.3: *Μοντέλο δεδομένων του VISION Cloud*

8.4.2 Αρχιτεκτονική του Μηχανισμού Επίβλεψης

Σε κάθε κόμβο του συστήματος τρέχει ένα στιγμιότυπο του μηχανισμού επίβλεψης. Το κάθε στιγμιότυπο είναι υπεύθυνο να συλλέγει δεδομένα από τον τοπικό κόμβο και να τα επεξεργάζεται πριν τα αποστείλει σε άλλα μέρη της πλατφόρμας και κυρίως σε συστατικά που είναι υπεύθυνα για την λήψη αποφάσεων όπως αυτά που είναι υπεύθυνα να έχουν συνεχώς μία ενημερωμένη εικόνα για την κατάσταση της πλατφόρμας (Voulodimos κ. συν., 2011).

Το κάθε στιγμιότυπου του μηχανισμού παρακολούθησης αποτελείται από διάφορα μέρη. Πίο συγκεκριμένα αποτελείται από τα εξής:

- *Συλλέκτης.* Η βιβλιοθήκη αυτή χρησιμοποιείται από κάθε παραγωγό πληροφοριών για να στέλνει γεγονότα στο τοπικό στιγμιότυπο του μηχανισμού παρακολούθησης. Είναι εύκολη στη χρήση και σκοπός της είναι να κρύβει την εσωτερική δικτύωση του μηχανισμού.

- *Κυρίως στιγμιότυπο.* Το κυρίως στιγμιότυπο του μηχανισμού παρακολούθησης είναι υπεύθυνο να συντονίζει τα υπόλοιπα μέρη του συστήματος. Είναι το κύριο μέρος του συστήματος καθώς είναι αυτό που λαμβάνει τα γεγονότα και τα αποστέλλει στους καταναλωτές.

- *Μηχανή κανόνων (rule engine).* Κάθε γεγονός που λαμβάνεται επεξεργάζεται από μία μηχανή κανόνων η οποία είναι υπεύθυνη να επιλέξει τις επόμενες ενέργειες, αν δηλαδή θα εκτελεστεί συνάθροιση, αν το γεγονός θα αποσταλεί σε άλλο κόμβο κτλ.

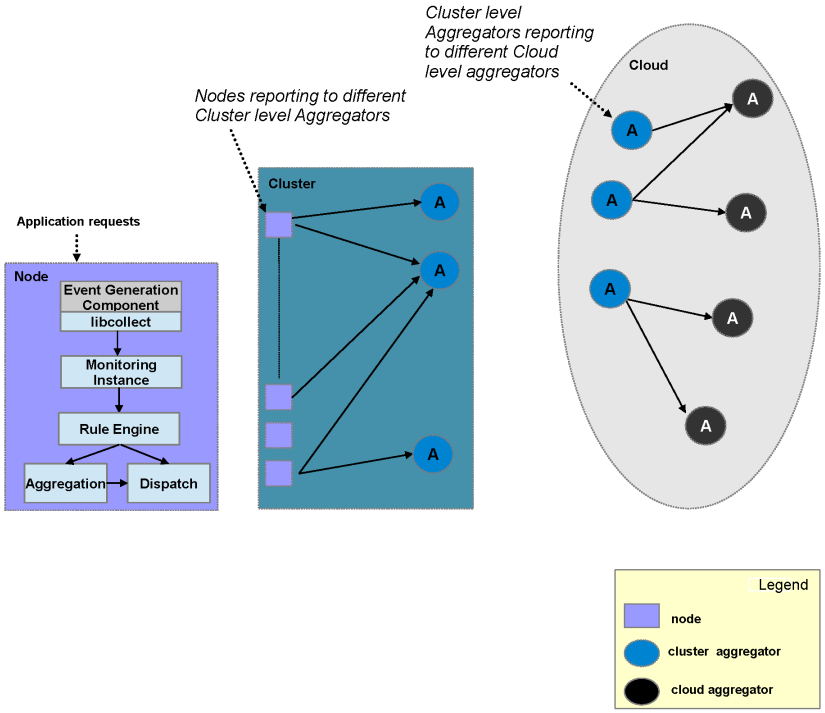
- *Συναθροιστής.* Το μέρος αυτό είναι υπεύθυνο για να εκτελεί τις διαφορετικές συναρτήσεις συνάθροισης όταν αυτό είναι απαραίτητο.

Το αποτέλεσμα του συναθροιστή είναι γεγονότα τα οποία αποτελούν συνάθροιση άλλων γεγονότων.

- *Αποστολέας*. Το μέρος αυτό είναι υπεύθυνο να αποστέλλει γεγονότα σε καταναλωτές με την χρήση μίας βιβλιοθήκης ασύγχρονης επικοινωνίας υψηλής απόδοσης.

Τα διαφορετικά μέρη έχουν αναπτυχθεί με την γλώσσα Java, ενώ η δικτύωση μεταξύ των παραγωγών και των καταναλωτών επιτυγχάνεται με χρήση της βιβλιοθήκης 0MQ (*Zeromq*, α.χ.). Η 0MQ υποστηρίζει πολλαπλά πρότυπα επικοινωνίας και μεταφοράς όπως in-process, inter-process, TCP και multicast, αναλόγως από την τοποθεσία του λήπτη του μηνύματος. Επιτρέπει επίσης τον έλεγχο πολλών πολιτικών επικοινωνίας και παραμέτρων ποιότητας υπηρεσίας, όπως χρονικά όρια και όρια στάθμης μηνυμάτων σε μία ουρά (high water mark). Μία σημαντική παράμετρος της βιβλιοθήκης είναι ότι δεν χρειάζεται να χρησιμοποιηθούν μεσίτες (brokers) επιτρέποντας έτσι την απευθείας επικοινωνία μεταξύ ομότιμων μελών και μειώνοντας το κόστος συντήρησης του συστήματος. Επιπρόσθετα, η βιβλιοθήκη έχει πολύ χαμηλές απαιτήσεις σε μνήμη.

Ειδικά κομμάτια κώδικα συλλέγουν πληροφορίες σχετικές με τους φυσικούς πόρους του συστήματος και τις αποστέλλουν στον τοπικό αντιπρόσωπο του συστήματος παρακολούθησης. Η εφαρμογή παρέχει υψηλές δυνατότητες διαμόρφωσης και έτσι είναι δυνατόν να χρησιμοποιηθούν και άλλοι μηχανισμοί συλλογής πληροφοριών ως πηγές πληροφόρησης χρησιμοποιώντας τις διεπαφές που παρέχει το σύστημα. Το μόνο προαπαιτούμενο είναι να χρησιμοποιηθεί το ίδιο σχήμα για την περιγραφή



Σχήμα 8.4: *Αρχιτεκτονική μηχανισμού παρακολούθησης*

των γεγονότων.

Οι καταναλωτές πληροφοριών μπορούν να εγγραφούν σε συγκεκριμένα θέματα (topics) και να λαμβάνουν γεγονότα. Κάθε θέμα αντιπροσωπεύει έναν συγκεκριμένο κανόνα παρακολούθησης και συνάθροισης.

8.4.3 Συνάθροιση Πληροφοριών

Η φυσική αρχιτεκτονική του VISION Cloud, αλλά και των περισσότερων Νεφών, μπορεί να χωριστεί σε τέσσερα επίπεδα. Συγκεκριμένα μπορεί να ορισθεί το επίπεδο του κόμβου, του cluster, του κέντρου δεδομένων και τέλος του Νέφους. Οι παραγωγοί και οι καταναλωτές πληροφοριών μπορούν να βρίσκονται σε οποιοδήποτε επίπεδο. Αυτό σημαίνει ότι ένας καταναλωτής μπορεί να λαμβάνει γεγονότα από τον ίδιο κόμβο ή από ένα απομακρυσμένο κόμβο σε άλλο κέντρο δεδομένων.

Είναι προφανές ότι όσο ανεβαίνουμε στην φυσική αρχιτεκτονική (από τον κόμβο στο Νέφος) το κόστος της επικοινωνίας αυξάνει. Επίσης, πολλές υπηρεσίες διαχείρισης της πλατφόρμας χρειάζονται πληροφορίες επίβλεψης διαφορετικών επιπέδων συνάθροισης και με διαφορετικούς χρονισμούς. Παραδείγματος χάριν, υπάρχει η ανάγκη συνάθροισης των ενεργειών ανάγνωσης ενός χρήστη σε όλο το Νέφος, καθώς και η ανάγκη συνάθροισης της ρυθμαπόδοσης σε ένα cluster. Λόγω αυτών των διαφορετικών απαιτήσεων, καθώς και του κόστους επικοινωνίας, ένας από τους σχεδιαστικούς στόχους του μηχανισμού είναι να παρέχει δυνατότητες συνάθροισης όσο το δυνατόν πιο κοντά στην πηγή γίνεται πριν διαδοθούν περαιτέρω. Για αυτό το λόγο το σύστημα διαχωρίζει ανάμεσα σε διαφορετικά επίπεδα συνάθροισης. Πιο

συγκεκριμένα, ένας κανόνας μπορεί να ορισθεί ανά:

- χρήστη
- ένοικο
- κόμβο
- cluster
- κέντρο δεδομένων και
- Νέφος

Επίσης είναι δυνατόν να ορισθεί το χρονικό διάστημα κατά το οποίο θα εκτελείται η συνάθροιση. Ένας κανόνας συνάθροισης στο επίπεδο του χρήστη σημαίνει ότι θα πρέπει να συλλεχθούν πληροφορίες από όλο το Νέφος, καθώς ένας χρήστης μπορεί να εκτελέσει εντολές σε οποιονδήποτε κόμβο του Νέφους. Παρ΄ όλα αυτά μπορεί να εκτελεστεί μερική συνάθροιση στα κατώτερα επίπεδα, ώστε να επιτευχθεί μείωση του κόστους επικοινωνίας. Παραδείγματος χάριν, αν υπάρχει ένα κανόνας ο οποίος ορίζει ότι θα πρέπει να υπολογίζονται οι ενέργειες ανάγνωσης ενός χρήστη κάθε 30 λεπτά, τότε ο κάθε κόμβος μπορεί να συλλέγει αυτά τα γεγονότα, να τα αθροίζει και να τα μεταφέρει τα αποτελέσματα σε υψηλότερα επίπεδα σε συγκεκριμένα χρονικά διαστήματα. Έτσι, τα μηνύματα που περιέχουν πληροφορίες επίβλεψης μειώνονται σημαντικά.

8.4.4 Ανοχή σε Σφάλματα

Σε ένα μεγάλο κατανεμημένο σύστημα, όπως είναι ένα Νέφος αποθήκευσης, είναι αναμενόμενο να υπάρξουν σφάλματα κατά την λειτουργία

του. Έτσι, το σύστημα παρακολούθησης θα πρέπει να είναι σε θέση να αντεπεξέλθει σε αυτά και να βεβαιώνει ότι τα μηνύματα φτάνουν στον προορισμό τους. Φυσικά η ανοχή σε σφάλματα έχει κόστος και υπάρχουν περιπτώσεις όπου το να χαθούν ορισμένα μηνύματα δεν είναι καταστροφικό για το σύστημα. Για αυτόν τον λόγο οι καταναλωτές πληροφοριών μπορούν να ορίσουν το επίπεδο ασφάλειας της παράδοσης για κάθε πληροφορία. Αν επιθυμούν να είναι σίγουρη η παράδοση τότε κάθε πληροφορία επίβλεψης αποθηκεύεται σε μία κατανεμημένη βάση δεδομένων, συγκεκριμένα στην Apache Cassandra (*Apache Cassandra*, α.χ.). Αν υπάρξει σφάλμα σε κάποιον κόμβο τότε τα γεγονότα που έχουν συσσωρευτεί μπορούν να ανακτηθούν από ένα άλλο στιγμιότυπο του μηχανισμού και να προωθηθούν στον καταναλωτή. Όταν ένα γεγονός επιδοθεί στον καταναλωτή σβήνεται από την κατανεμημένη βάση δεδομένων.

8.4.5 Δυνατότητες Επίβλεψης για τον Χρήστη

Εκτός από το να παρέχει δυνατότητες επίβλεψης στα διαφορετικά μέρη της πλατφόρμας, ο μηχανισμός επίβλεψης επεκτάθηκε ώστε να παρέχει παρόμοιες δυνατότητες και στους χρήστες. Για να επιτευχθεί αυτό επεκτάθηκαν οι ουρές ειδοποίησης του CDMI. Για να δημιουργηθεί μία ουρά ειδοποίησης ο χρήστης δημιουργεί μία συνηθισμένη ουρά κατά το CDMI και προσθέτει μεταδεδομένα, τα οποία αποτελούν εντολές προς την πλατφόρμα για το ποια δεδομένα θα πρέπει να περιέχονται. Μερικά από τα γεγονότα που μπορούν να ορισθούν είναι:

- cdmi_create_processing: Παράγεται ένα γεγονός όταν δημιουργείται ένα

αντικείμενο και η λειτουργία δεν έχει ακόμα τερματιστεί.

- cdmi_create_complete: Παράγεται ένα γεγονός όταν ολοκληρωθεί η δημιουργία ενός αντικειμένου.
- cdmi_read: Παράγεται ένα γεγονός όταν αναγνωσθεί ένα αντικείμενο
- cdmi_modify_processing: Παράγεται ένα γεγονός όταν τροποποιείται ένα αντικείμενο αλλά η ενέργεια δεν έχει ολοκληρωθεί.
- cdmi_modify_complete: Παράγεται ένα γεγονός όταν ολοκληρωθεί η τροποποίηση ενός αντικειμένου.
- cdmi_rename: Παράγεται ένα γεγονός όταν μετονομάζεται ένα αντικείμενο.
- cdmi_delete: Παράγεται ένα γεγονός όταν διαγράφεται ένα αντικείμενο.

Ένας χρήστης μπορεί επίσης να επιλέξει να ορίσει έναν κανόνα που θα τροφοδοτεί μία ουρά. Όταν ο χρήστης δώσει μία εντολή δημιουργίας, το σύστημα ελέγχει αν ο κανόνας έχει ορισθεί σωστά και κατόπιν δημιουργεί την αντίστοιχη ουρά. Πρέπει εδώ να τονισθεί ότι παρότι χρησιμοποιούνται timestamps το CDMI ορίζει ότι δεν είναι δεδομένο ότι τα γεγονότα θα παραδοθούν με συγκεκριμένη σειρά και για αυτό το λόγο οι πελάτες θα πρέπει να είναι σε θέση να χειρισθούν γεγονότα εκτός σειράς. Επίσης ο χρήστης μπορεί να ορίσει κανόνες πάνω σε αντικείμενα και container που κατέχει ο ίδιος. Σε διαφορετική περίπτωση η δημιουργία της ουράς αποτυγχάνει.

Ένας εξωτερικός χρήστης μπορεί επομένως να κάνει τα ακόλουθα:

- Να δημιουργήσει μία καινούρια ουρά

- Να διαβάσει μία ουρά
- Να διαγράψει μία ουρά και όλα τα δεδομένα της

Η εξωτερική διεπαφή του μηχανισμού επίβλεψης επιτρέπει σε δύο παρόχους να δημιουργήσουν σενάρια ομοσπονδίας καθώς μπορούν να επιβλέπουν μερικώς τις ενέργειες που λαμβάνουν χώρα στο ομόσπονδο Νέφος. Αυτό μπορεί να συμβεί με αυτοματοποιημένο τρόπο, χωρίς την ανάγκη παρέμβασης από κάποιον χειριστή.

8.4.6 Ορισμός Γεγονότων και Κανόνων

Τα γεγονότα τα οποία παράγονται από το σύστημα έχουν μία απλή μορφή η οποία περιέχει κάποια σταθερά πεδία καθώς και πεδία τα οποία εξαρτώνται από κάθε συγκεκριμένο γεγονός. Τα υποχρεωτικά πεδία είναι:

- *IP*: Αυτό το πεδίο περιέχει την διεύθυνση IP του παραγωγού του γεγονότος.
- *Service*: Το πεδίο αυτό περιέχει το όνομα του παραγωγού του γεγονότος.
- *Timestamp*: Το πεδίο αυτό χρησιμοποιείται από την βιβλιοθήκη για να προστίθεται ένας χρονικός προσδιορισμό σε κάθε μήνυμα.
- *ID*: Κάθε γεγονός πρέπει να περιέχει έναν μοναδικό αριθμό για την ταυτοποίησή του.
- *Type*: Το πεδίο αυτό περιέχει τον τύπο του δημιουργηθέντος γεγονότος.

```
{
"event": {
"IP": "10.0.2.211",
"Service": "object_service",
"timestamp": "2012-06-15 15:32:10.137",
"ID": "111",
"Type": "BytesRead",
"Bytes": 1000,
"Tenant": "ntua"
"User":"john"}
}
```

Σχήμα 8.5: *Παράδειγμα ενός γεγονότος σε μορφή JSON*

Οι κανόνες που μπορούν να δωθούν στον μηχανισμό για να καθορίσουν πώς θα πρέπει να χειριστεί ο μηχανισμός κάθε γεγονός αποτελούνται από τρία μέρη. Στο πρώτο μέρος καθορίζεται το όνομα του topic το οποίο χρησιμοποιείται για να γίνει η εγγραφή στην συγκεκριμένη ουρά και πρέπει να είναι μοναδικό στο Νέφος. Επίσης ορίζονται το επίπεδο που θα γίνεται η συνάθροιση (κόμβος, cluster, κτλ) καθώς και η χρόνος κατά τον οποίον θα εκτελείται. Στο μέρος when ορίζονται οι περιπτώσεις κατά τις οποίες θα εφαρμόζεται ο κανόνας. Για αυτόν τον σκοπό ορίζονται βασικές συναρτήσεις bool πάνω στα πεδία του γεγονότος. Αν για κάποιο γεγονός ικανοποιεί πολλαπλούς κανόνες τότε θα εφαρμοστούν όλοι οι κανόνες. Στη συνέχεια, στο μέρος then ορίζονται οι ενέργειες που θα πρέπει να εκτελεστούν αν ένα γεγονός ικανοποιεί τις συνθήκες ενός κανόνα. Αφού εφαρμοστεί η επιλεγμένη συνάρτηση συνάθροισης το αποτέλεσμα σώζεται σε μία ουρά.

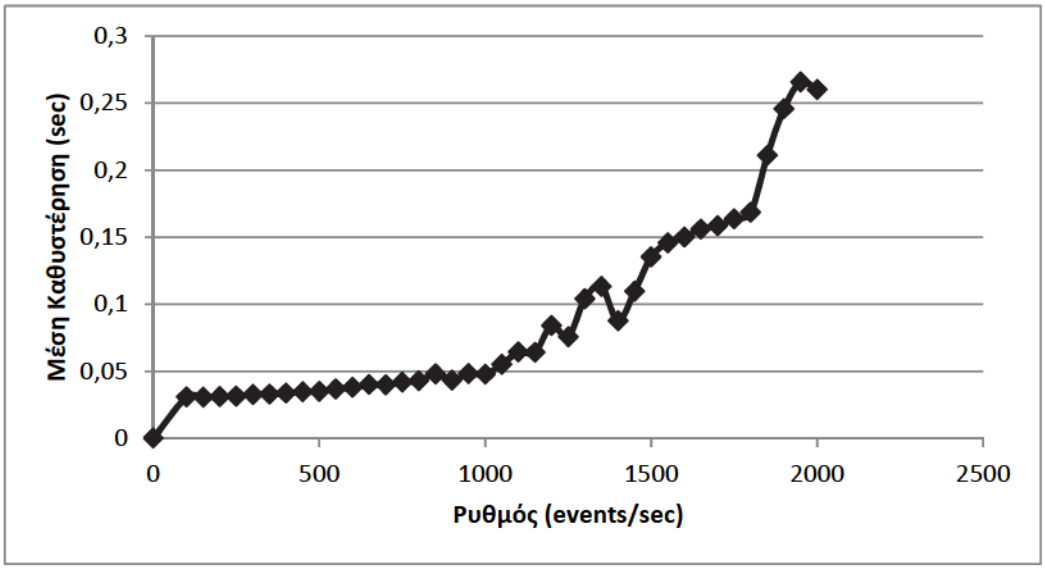
```
-- Report the sum of read bytes from the users of tenant NTUA
every 10 minutes, if the reads have come from the object service
and the #bytes read are larger than 1000.
-- The aggregation result will be queued under topic 'reads_ntua'

topic='reads_ntua'
agg_level='cloud'
agg_period=10

when
  (e.type = 'Bytesread') AND (e.tenant = 'ntua') AND
  (e.producer = 'object_service') AND (e.bytes >= 1000)
then
  SUM e.bytes
```

Σχήμα 8.6: *Παράδειγμα ενός κανόνα*

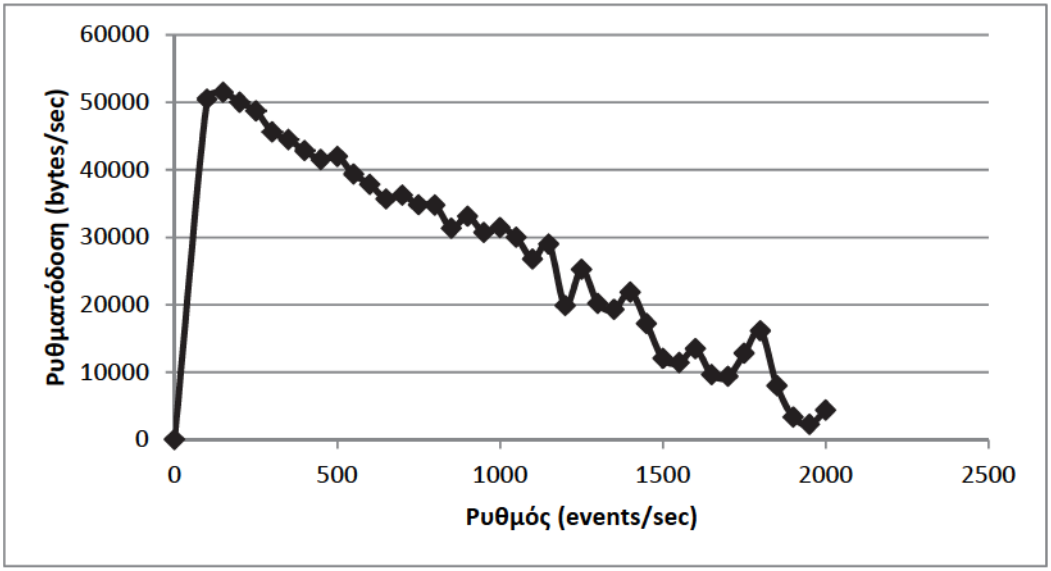
8.5 Ποσοτική Αξιολόγηση

Για να αξιολογήσουμε την απόδοση του μηχανισμού εκτελέσαμε το εξής πείραμα: Ένας παραγωγός γεγονότων παρήγαγε γεγονότα σταθερού μεγέθους 1024 bytes με μεταβαλλόμενο ρυθμό από 100 events το δευτερόλεπτο ως 2000 events το δευτερόλεπτο τα οποία παραδίδονταν σε έναν καταναλωτή στο ίδιο δίκτυο. Για κάθε μήνυμα μετρήθηκε η καθυστέρηση στην λήψη του από τον καταναλωτή καθώς και η ρυθμαπόδοση του συστήματος. Τα αποτελέσματα φαίνονται στο Σχήμα 8.7 και Σχήμα 8.8.

Παρατηρούμε ότι η καθυστέρηση που εισάγει ο μηχανισμός παρακολούθησης αυξάνεται όσο αυξάνει ο ρυθμός παραγωγής μηνυμάτων. Εν τούτοις η καθυστέρηση είναι αρκετά χαμηλή, περίπου 0,05 δευτερόλεπτα για ρυθμό έως 1000 μηνύματα το δευτερόλεπτο και φτάνει ως τα 0,26 δευτερόλεπτα για ρυθμό 2000 μηνύματα ανά δευτερόλεπτο. Αντίστοιχα η ρυθμαπόδοση του συστήματος πέφτει όσο αυξάνει ο ρυθμός παραγωγής μηνυμάτων. Η



Σχήμα 8.7: *Καθυστέρηση στην παράδοση ενός μηνύματος*



Σχήμα 8.8: *Ρυθμαπόδοση του συστήματος*

λειτουργία του συστήματος κρίνεται αρκετά ικανοποιητική, καθώς αναφέρεται στην δυνατότητα ενός μόνο κόμβου. Έτσι ο ρυθμός 1000 μηνυμάτων το δευτερόλεπτο σημαίνει ότι ο κόμβος δέχεται 1000 αιτήσεις από τον χρήστη.

9

Συμπεράσματα και Μελλοντική Εργασία

Οι πρόσφατες τεχνολογικές εξελίξεις στον τομέα της πληροφορικής και δικτύωσης έχουν οδηγήσει στην ταχεία ανάπτυξη του Διαδικτύου. Οι άνθρωποι σε όλο τον κόσμο άρχισαν να το χρησιμοποιούν ως εργαλείο όχι μόνο για ανταλλαγή πληροφοριών, αλλά και για την εκτέλεση διαφόρων εργασιών, από τις ηλεκτρονικές αγορές ως την εξ' αποστάσεως εργασία. Οι επιχειρήσεις αναγκάστηκαν να συμβαδίσουν με την έκρηξη του Διαδικτύου, προκειμένου να αυξήσουν τα κέρδη τους τόσο από τη μείωση του κόστους και όσο και από την εκπλήρωση των δραστικά αυξανόμενων απαιτήσεων των πελατών τους. Η οργανωτική αποτελεσματικότητα και η δυνατότητα ανταπόκρισης αποτελούν κρίσιμους παράγοντες για τον προσδιορισμό της αποδοτικότητας ενός οργανισμού, ιδιαίτερα στο παγκόσμιο επιχειρηματικό περιβάλλον του σήμερα, όπου οι πόροι και οι άνθρωποι τείνουν να είναι γεωγραφικά διάσπαρτοι. Το ίδιο μπορεί να ειπωθεί για την κοινότητα της ηλεκτρονικής επιστήμης (eScience), όπου το κέρδος παίρνει μια διαφορετική μορφή.

Παράλληλα, με την εμφάνιση των κατανεμημένων περιβαλλόντων όπως οι υποδομές Πλέγματος και Νέφους, οι οποίες παρέχουν πολλά

οφέλη, όπως αξιοπιστία, αποδοτικότητα και επεκτασιμότητα, τα συστήματα διαχείρισης ροών εργασίας έχουν λάβει μεγάλη προσοχή. Η διαχείριση ροών εργασίας αποτελεί την επισημοποίηση της επιχειρησιακής διαδικασίας και την αυτόματη εκτέλεσή της. Αυτό παρέχει πολλά πλεονεκτήματα, δεδομένου ότι συστατικά μέρη της εφαρμογής που είναι υπεύθυνα για την εκτέλεση μιας συγκεκριμένης εργασίας μπορούν να αναπτυχθούν αυτόνομα, χωρίς την ανάγκη να ενσωματώνουν οποιαδήποτε επιχειρηματική λογική. Το καθήκον αυτό επαφίεται στο σύστημα διαχείρισης ροών εργασίας. Αυτό σημαίνει ότι τα μέρη αυτά μπορούν να επαναχρησιμοποιηθούν σε διαφορετικές ροές εργασίας, ενώ η ροή εργασίας μπορεί εύκολα να αλλάξει ώστε να παρέχει νέες υπηρεσίες. Για τους λόγους αυτούς, η μοντελοποίηση και η εκτέλεση επιχειρησιακών διεργασιών εμφανίζει αυξημένο ερευνητικό ενδιαφέρον.

Στην παρούσα εργασία μελετήθηκε η δυνατότητα εκτέλεσης εφαρμογών ελαστικού πραγματικού χρόνου σε κατανεμημένα περιβάλλοντα Νέφους. Κύριο συστατικό σε κάθε υπηρεσιοστρεφή υποδομή αποτελεί το ΣΔΡΕ, που αναλαμβάνει να ενορχηστρώσει τις διαφορετικές υπηρεσίες οι οποίες αποτελούν μια κατανεμημένη εφαρμογή. Έτσι μελετήθηκαν οι υπάρχουσες προτάσεις στον τομέα των ΣΔΡΕ και ιδιαιτέρως η γλώσσα BPEL, ούσα η πλέον διαδεδομένη γλώσσα για τον προσδιορισμό και την εκτέλεση ροών εργασίας. Παράλληλα εξετάστηκαν οι δυνατότητες πραγματικού χρόνου που παρέχουν τα ΛΣ-ΓΧ, ώστε να προσδιορισθεί ποιες από αυτές μπορούν να χρησιμοποιηθούν ως βάση για την ανάπτυξη ενός ΣΔΡΕ. Στη συνέχεια προτείνεται η μοντελοποίηση της εφαρμογής ως ροή εργασίας, λαμβάνοντας υπ'οψη της απαιτήσεις της εφαρμογής. Τέλος, προτείνεται μία αρχιτεκτονική που μπορεί να καλύψει τα κενά που παρουσιάζει η διεθνής βιβλιογραφία στον

τομέα των κατανεμημένων ΣΔΡΕ.

Το προτεινόμενο ΣΔΡΕ ακολουθεί μία πολυεπίπεδη αρχιτεκτονική, κατά την οποία ένα μέρος του τοποθετείται εντός του εικονικοποιημένου περιβάλλοντος όπου τρέχουν και οι υπηρεσίες της εφαρμογής. Έτσι το ίδιο το ΣΔΡΕ μπορεί να μοντελοποιηθεί και οι ανάγκες του σε φυσικούς πόρους να αλλάξει ανάλογα με την ροή εργασίας που καλείται να διαχειριστεί. Αυτό οδηγεί αφενός μεν στη δυνατότητα παροχής ισχυρών εγγυήσεων ποιότητας υπηρεσίας, καθώς η λειτουργία του ΣΔΡΕ και της εφαρμογής μπορεί να μοντελοποιηθεί και να αναλυθεί εκ των προτέρων, αφετέρου δε στην καλύτερη διαχείριση των υφιστάμενων φυσικών πόρων, καθώς δεσμεύονται ακριβώς οι πόροι που απαιτούνται για την εκτέλεση μίας εφαρμογής. Επίσης το προταθέν ΣΔΡΕ έχει την δυνατότητα να εκτελεί διορθωτικές ενέργειες σε περιπτώσεις όπου παραβιάζονται ή τείνουν να παραβιαστούν οι περιορισμοί που έχουν ορισθεί για την εφαρμογή.

Το αντικείμενο της διατριβής και η μεθοδολογία επίλυσης του προβλήματος επιτρέπει πολλές μελλοντικές επεκτάσεις. Πρώτο μέλημα είναι να επεκταθεί η μοντελοποίηση της ροής εργασίας, ώστε να επιτρέπει τον αυτόματο έλεγχο της ορθότητάς της. Στο πλαίσιο αυτό έχει ήδη ξεκινήσει μία προσπάθεια για χρήση δικτύων Petri κατά τη φάση μοντελοποίησης. Έτσι, μία ροή εργασίας εκπεφρασμένη με την γλώσσα που προτάθηκε στην παρούσα διατριβή μετατρέπεται σε ένα δίκτυο Petri. Αυτό μας δίνει την δυνατότητα να ελέγχουμε, μέσω του μαθηματικού φορμαλισμού των δικτύων Petri, ποιοτικές και ποσοτικές ιδιότητες της ροής εργασίας, όπως την προσιτότητα, την περατότητα, τη συνέπεια και τη διάρκεια (liveness). Από την ανάλυση αυτή είναι δυνατόν να εντοπισθούν σφάλματα ή να γίνουν δομικές βελτιώσεις στην

προς εκτέλεση ροή εργασίας. Παράλληλα, τα δίκτυα Petri μπορούν εύκολα να παρουσιαστούν μέσω γραφημάτων με αποτέλεσμα να βοηθούν στην κατανόηση μίας επιχειρησιακής διαδικασίας από τον τελικό χρήστη.

Η γλώσσα περιγραφής ροών εργασίας που προτάθηκε προϋποθέτει ότι οι ενέργειες των διεργασιών εκτελούνται από υπηρεσίες Ιστού. Παρότι οι υπηρεσίες Ιστού αποτελούν δομικό λίθο πολλών σύγχρονων επιχειρησιακών διεργασιών, εν τούτοις η απουσία ανθρωπίνων αλληλεπιδράσεων αποτελεί ένα σημαντικό κενό για πολλές πραγματικές διεργασίες. Έτσι, η μοντελοποίηση και η ακόλουθη ενορχήστρωση ανθρωπίνων δραστηριοτήτων ως αναπόσπαστο κομμάτι των ροών εργασίας αποτελεί μελλοντικό ερευνητικό σκοπό. Παραδείγματος χάριν, υπάρχουν πολλές περιπτώσεις όπου ένας ανθρώπινος παράγοντας οφείλει να επιτρέψει ή να απορρίψει την εκτέλεση μίας υπηρεσίας, να αναθέσει την εκτέλεση μίας εργασίας σε συγκεκριμένο χειριστή και γενικότερα να λάβει μία απόφαση που το σύστημα αδυνατεί να πάρει από μόνο του. Παρότι υπάρχουν ήδη προτάσεις στον συγκεκριμένο τομέα (όπως π.χ. η επέκταση BPEL4People), ο συνδυασμός ροών εργασίας που εμπεριέχουν ανθρώπινες δραστηριότητες και συστημάτων πραγματικού χρόνου αποτελεί ανοιχτό ερευνητικό θέμα.

Μία άλλη διάσταση που πρόκειται να διερευνήσουμε είναι ο έλεγχος της ροής εργασίας βάσει των δεδομένων. Κατά την εκτέλεση ροών εργασίας βάσει του ελέγχου (control-driven) οι συνδέσεις μεταξύ των δραστηριοτήτων σε μια ροή εργασίας συνιστούν μεταφορά του ελέγχου από την προηγούμενη εργασία σε εκείνη που ακολουθεί. Αυτό περιλαμβάνει δομές ελέγχου, όπως η *sequence* και οι βρόχοι επανάληψης. Από την άλλη μεριά οι ροές εργασίας βάσει των δεδομένων (data-driven) είναι σχεδιασμένες κυρίως για

την υποστήριξη εφαρμογών που εξαρτώνται από τα δεδομένα. Οι εξαρτήσεις αντιπροσωπεύουν τη ροή των δεδομένων μεταξύ των δραστηριοτήτων της ροής εργασίας από τον παραγωγό των δεδομένων στον καταναλωτή. Είναι προφανές ότι το προταθέν σύστημα βασίζεται κυρίως στην εκτέλεση ροών εργασίας βάσει του ελέγχου. Σκοπός μας είναι να το επεκτείνουμε ώστε να μπορεί να εκτελεί ροές εργασίας που βασίζονται στις αυξημένες δυνατότητες που παρέχουν τα σύγχρονα Νέφη Αποθήκευσης και κυρίως στην χρήση *μεταδεδομένων* (metadata) για την ενεργοποίηση διαφορετικών ροών εργασίας.

Βιβλιογραφία

Aalst, W. M. P. V. D., Hofstede, A. H. M. T., Kiepuszewski, B., & Barros, A. P. (2003). Workflow Patterns. , *14*(1), 5-51–.

[Αναφέρεται στη Σελ. 29]

Abeni, L., & Buttazzo, G. (1998). Integrating Multimedia Applications in Hard Real-Time Systems. Στο *Proceedings of the IEEE Real-Time Systems Symposium* (σελ. 4–). Washington, DC, USA: IEEE Computer Society. Ανακτήθηκε από `http://dl.acm.org/citation.cfm?id=827270.829047`

[Αναφέρεται στη Σελ. 72]

Allcock, W., Bresnahan, J., Kettimuthu, R., & Link, M. (2005). The Globus Striped GridFTP Framework and Server. Στο *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference* (σελ. 54).

[Αναφέρεται στη Σελ. 94]

Amazon CloudWatch. (α.χ.). Ανακτήθηκε από `https://aws.amazon.com/cloudwatch/`

[Αναφέρεται στη Σελ. 145]

Andreozzi, S., Bortoli, N. D., Fantinel, S., Ghiselli, A., Rubini, G. L., Tortone, G., & Vistoli, M. C. (2005). GridICE: a monitoring service for Grid systems. *Future Generation Computer Systems*, *21*(4), 559 - 571. doi: 10.1016/j.future .2004.10.005

[Αναφέρεται στη Σελ. 144]

Apache Cassandra. (α.χ.). Ανακτήθηκε από `https://cassandra.apache .org/`

[Αναφέρεται στη Σελ. 156]

ASG. (α.χ.). *Adaptive Services Grid.* Ανακτήθηκε από `http://asg-platform .org/cgi-bin/twiki/view/Public`

[Αναφέρεται στη Σελ. 30]

Ayers, Y., & Barabanov, V. (1997). Introducing Real-Time Linux. *Linux J.*, *1997*.

[Αναφέρεται στη Σελ. 56]

Barth, W. (2008). *Nagios: System and Network Monitoring*. No Starch Press.

[Αναφέρεται στη Σελ. 142]

Battre, D., Wieder, P., & Ziegler, W. (2008). *WS-Agreement Specification Version 1.0.* Ανακτήθηκε από `http://www.ogf.org/documents/GFD.167.pdf`

[Αναφέρεται στη Σελ. 40]

Beco, S., Cantalupo, B., Giammarino, L., Matskanis, N., & Surridge, M. (2005). OWL-WS: A Workflow Ontology for Dynamic Grid Service Composition. Στο *Proceedings of 1st ieee international conference on e-science and grid computing* (σελ. 148-155–). IEEE Computer Society.

[Αναφέρεται στη Σελ. 32]

Beco, S., Cantalupo, B., & Terracina, A. (2006). The Role of Workflow in Next Generation Business Oriented Grids: Two Different Approaches Leading to a

Unified Vision. Στο *Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)* (σελ. 38–). e-science.

[Αναφέρεται στη Σελ. 32]

Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., ... Wolski, R. (2001). The GrADS Project: Software Support for High-Level Grid Application Development. *International Journal of High Performance Computing Applications(JHPCA)*, *15*(4), 327-344.

[Αναφέρεται στη Σελ. 22]

Bishop, R. (2007). *Mechatronic System Control, Logic, and Data Acquisition.* Taylor & Francis.

[Αναφέρεται στη Σελ. 12]

Blanquer, J., Bruno, J., Gabber, E., Mcshea, M., Ozden, B., Silberschatz, A., & Singh, A. (1999). Resource Management for QoS in Eclipse/BSD. Στο *Proceedings of the FreeBSD'99 Conference.*

[Αναφέρεται στη Σελ. 59]

Bowers, S., Ludascher, B., Ngu, A. H. H., & Critchlow, T. (2006). Enabling ScientificWorkflow Reuse through Structured Composition of Dataflow and Control-Flow. Στο *Proceedings of the 22nd International Conference on Data Engineering Workshops* (σελ. 70–). IEEE Computer Society.

[Αναφέρεται στη Σελ. 31]

Brandic, I., Pllana, S., & Benkner, S. (2008). Specification, planning, and execution of QoS-aware Grid workflows within the Amadeus environment. *Concurr. Comput. : Pract. Exper.*, *20*, 331–345.

[Αναφέρεται στη Σελ. 22]

Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan,

S., ... Sycara, K. (2004). *OWL-S: Semantic Markup for Web Services* (22November 2004 έκδ.). Website. Ανακτήθηκε από `http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/`

[Αναφέρεται στη Σελ. 32]

Buttazzo, G., Lipari, G., Abeni, L., & Caccamo, M. (2005). *Soft Real-Time Systems: Predictability vs. Efficiency (Series in Computer Science)*. Plenum Publishing Co.

[Αναφέρεται στη Σελ. 10], [Αναφέρεται στη Σελ. 55]

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, *25*, 599–616.

[Αναφέρεται στη Σελ. 5]

Checconi, F., Cucinotta, T., & Faggioli, D. (2009). Hierarchical Multiprocessor CPU Reservations for the Linux Kernel. Στο *5th OSPERT Workshop.*

[Αναφέρεται στη Σελ. 58], [Αναφέρεται στη Σελ. 68], [Αναφέρεται στη Σελ. 79]

Chou, T., Chang, S., Lu, Y., Wang, Y., Ouyang, M., Shih, C., ... Liu, J. (2009). EMWF for Flexible Automation and Assistive Devices. Στο *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE* (σελ. 243 -252).

[Αναφέρεται στη Σελ. 26]

Claris, C., George, N. R., & Khaled, H. (2007). Efficient Implementation of Best-Fit Scheduling for Advance Reservations and QoS in Grid. Στο *1st IEEE/IFIP Intl. Workshop on End-to-end Virtualization and Grid Management (EVGM)* (σελ. –). Ανακτήθηκε από `http://citeseerx.ist`

.psu.edu/viewdoc/summary?doi=10.1.1.88.6261

[Αναφέρεται στη Σελ. 40], [Αναφέρεται στη Σελ. 42]

Clayman, S., Galis, A., Chapman, C., Toffetti, G., Rodero-Merino, L., Vaquero, L. M., ... Rochwerger, B. (2010). Monitoring Service Clouds in the Future Internet. Στο *Towards the Future Internet - Emerging Trends from European Research*. IOS Press.

[Αναφέρεται στη Σελ. 146]

Clayman, S., Toffetti, G., Galis, A., & Chapman, C. (2012). Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice. Στο M. Villari, I. Brandic, & F. Tusa (Επιμ.), (σελ. 242-265). IGI Global.

[Αναφέρεται στη Σελ. 146]

Cloud Data Management Interface (CDMI). (α.χ.). Ανακτήθηκε από http://snia.org/sites/default/files/CDMI_SNIA_Architecture_v1.0.1.pdf

[Αναφέρεται στη Σελ. 149]

CloudStatus. (α.χ.). Ανακτήθηκε από http://www.cloudstatus.com/

[Αναφέρεται στη Σελ. 146]

Coles, S. J., Frey, J. G., Hursthouse, M. B., Light, M. E., Milsted, A. J., Carr, L. A., ... Day, M. (2006). An e-Science environment for service crystallography - from submission to dissemination. *Journal of Chemical Information and Modeling*, *46*(3), 1006-1016.

[Αναφέρεται στη Σελ. 32]

Compuware Gomez. (α.χ.). Ανακτήθηκε από https://www.compuware.com/application-performance-management/gomez-apm-products.html

[Αναφέρεται στη Σελ. 146]

Cucinotta, T. (2008). Access Control for Adaptive Reservations on Multi-User Systems. Στο *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS 08. IEEE* (σελ. 387 -396). doi: 10.1109/RTAS.2008.16

[Αναφέρεται στη Σελ. 69]

Cucinotta, T., Mancina, A., Anastasi, G., Lipari, G., Mangeruca, L., Checcozzo, R., & Rusina, F. (2009). A Real-Time Service-Oriented Architecture for Industrial Automation. *Industrial Informatics, IEEE Transactions on*, *5*(3), 267 -277. doi: 10.1109/TII.2009.2027013

[Αναφέρεται στη Σελ. 61]

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., ... Livny, M. (2004). Pegasus: Mapping Scientific Workflows onto the Grid. Στο M. D. Dikaiakos (Επιμ.), *Lecture Notes in Computer Science* (τ. 3165, σελ. 131-140–). Springer Berlin / Heidelberg. Ανακτήθηκε από `http://dx.doi.org/10.1007/978-3-540-28642-4_2`

[Αναφέρεται στη Σελ. 31]

Dierks, T., & Rescorla, E. (2008). *RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2.* Ανακτήθηκε από `http://tools.ietf.org/html/rfc5246`

[Αναφέρεται στη Σελ. 94]

DIP. (α.χ.). *Data, Information, and Process Integration with Semantic Web Services.* Ανακτήθηκε από `http://dip.semanticweb.org/`

[Αναφέρεται στη Σελ. 30]

Dozio, L., & Mantegazza, P. (2003). Real time distributed control systems using RTAI. Στο *Object-Oriented Real-Time Distributed Computing, 2003. Sixth*

IEEE International Symposium on (σελ. 11 - 18). doi: 10.1109/ISORC.2003.1199229

[Αναφέρεται στη Σελ. 56]

Durkee, D. (2010). Why cloud computing will never be free. *Commun. ACM*, *53*(5), 62–69. Ανακτήθηκε από `http://doi.acm.org/10.1145/1735223.1735242` doi: 10.1145/1735223.1735242

[Αναφέρεται στη Σελ. 12]

Eide, E., Stack, T., Regehr, J., & Lepreau, J. (2004). Dynamic CPU Management for Real-Time, Middleware-Based Systems. Στο *In Proc. of 10th IEEE Real-Time and Embedded Technology and Applications Symposium* (σελ. 286–295).

[Αναφέρεται στη Σελ. 61]

Eker, J., Janneck, J. W., Lee, E. A., Jie, L., Xiaojun, L., Ludvig, J., ... Yuhong, X. (2003). Taming heterogeneity - the Ptolemy approach. *Proceedings of the IEEE*, *91*(1), 127-144.

[Αναφέρεται στη Σελ. 23]

Erl, T. (2005). *Service-oriented Architecture: Concepts, Technology, and Design.* Upper Saddle River: Prentice Hall.

[Αναφέρεται στη Σελ. 5]

Faggioli, D., Mancina, A., Checconi, F., & Lipari, G. (2008). Design and implementation of a POSIX compliant sporadic server. Στο *Proceedings of the 10th Real-Time Linux Workshop (RTLW).*

[Αναφέρεται στη Σελ. 58]

Fahringer, T., Jugravu, A., Pllana, S., Prodan, R., Jr, C. S., & Truong, H. L. (2005). ASKALON: a tool set for cluster and Grid computing. *Concurrency and*

Computation: Practice and Experience, *17*, 143-169.

[Αναφέρεται στη Σελ. 22]

Fahringer, T., Prodan, R., Duan, R., Hofer, J., Nadeem, F., Nerieri, F., ... Wieczorek, M. (2007). ASKALON: A Development and Grid Computing Environment for Scientific Workflows. Στο I. J. Taylor, E. Deelman, D. B. Gannon, & M. Shields (Επιμ.), *Workflows for e-Science* (σελ. 450-471–). Springer London. Ανακτήθηκε από `http://dx.doi.org/10.1007/978-1-84628-757-2_27`

[Αναφέρεται στη Σελ. 31]

Farooq, U., Majumdar, S., & Parsons, E. W. (2005). *Efficiently Scheduling Advance Reservations in Grids.* Ottawa, Canada.

[Αναφέρεται στη Σελ. 38], [Αναφέρεται στη Σελ. 41], [Αναφέρεται στη Σελ. 42]

Fay-Wolfe, V., DiPippo, L., Cooper, G., Johnson, R., Kortmann, P., & Thuraisingham, B. (2000). Real-time CORBA. *Parallel and Distributed Systems, IEEE Transactions on*, *11*(10), 1073 - 1089. doi: 10.1109/71.888646

[Αναφέρεται στη Σελ. 60]

Fernandez-Baca, D. (1989). Allocating Modules to Processors in a Distributed System. , *15*(11), 1427-1436–.

[Αναφέρεται στη Σελ. 34]

Foster, I., & Kesselman, C. (1997). Globus: a Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications*, *11*(2), 115-128. doi: 10.1177/109434209701100205

[Αναφέρεται στη Σελ. 144]

Foster, I., & Kesselman, C. (Επιμ.). (1999). *The Grid: Blueprint For A New Computing Infrastructure*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[Αναφέρεται στη Σελ. 5]

Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud Computing and Grid Computing 360-Degree Compared. Στο *Grid Computing Environments Workshop, 2008. GCE '08* (σελ. 1 -10). doi: 10.1109/GCE.2008.4738445

[Αναφέρεται στη Σελ. 140]

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google file system. *SIGOPS Operating Systems Review*, *37*(5), 29–43. doi: 10.1145/1165389.945450

[Αναφέρεται στη Σελ. 147]

Giatrakos, N., Deligiannakis, A., Garofalakis, M., Sharfman, I., & Schuster, A. (2012). Prediction-based geometric monitoring over distributed data streams. Στο *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (σελ. 265–276). New York, NY, USA: ACM. doi: 10.1145/2213836.2213867

[Αναφέρεται στη Σελ. 142]

Globus. (α.χ.). *Globus Project.* Ανακτήθηκε από `http://www.globus.org`

[Αναφέρεται στη Σελ. 22], [Αναφέρεται στη Σελ. 31]

Gogouvitis, S. V., Kousiouris, G., Vafiadis, G., Kolodner, E. K., & Kyriazis, D. (2012). OPTIMIS and VISION Cloud: How to Manage Data in Clouds. Στο *Euro-Par 2011: Parallel Processing Workshops* (τ. 7155, σελ. 35–44). Springer.

[Αναφέρεται στη Σελ. 149]

Gopalan, K. (2001). *Real-Time Support in General Purpose Operating Systems.*

[Αναφέρεται στη Σελ. 57]

Gopalan, K., & Kang, K.-d. (2007). Coordinated Allocation and Scheduling of Multiple Resources in Real-time Operating Systems. *New York*. Ανακτήθηκε από `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.7012`

[Αναφέρεται στη Σελ. 59]

Hayes, B. (2008). Cloud Computing. *Commun. ACM*, *51*(7), 9-11.

[Αναφέρεται στη Σελ. 5]

Heinzl, S., Seiler, D., Juhnke, E., Stadelmann, T., Ewerth, R., Grauer, M., & Freisleben, B. (2009). A scalable service oriented architecture for multimedia analysis, synthesis and consumption. *Int. J. Web Grid Services*, *5*, 219–260.

[Αναφέρεται στη Σελ. 24]

IEEE Standard for Information Technology- Portable Operating System Interface (POSIX) Base Specifications, Issue 7. (2008). *IEEE Std 1003.1-2008 (Revision of IEEE Std 1003.1-2004)*, c1 -3826. doi: 10.1109/IEEESTD.2008.4694976

[Αναφέρεται στη Σελ. 52]

IEEE Standard for Information Technology- Standardized Application Environment Profile (AEP)-POSIX Realtime and Embedded Application Support. (2004). *IEEE Std 1003.13-2003 (Revision of IEEE Std 1003.13-1998)*, i -164. doi: 10.1109/IEEESTD.2004.94801

[Αναφέρεται στη Σελ. 55], [Αναφέρεται στη Σελ. 72]

Joeris, G., & Herzog, O. (1999). Towards Flexible and High-Level Modeling and Enacting of Processes. Στο *Proceedings of the 11th International Conference*

on Advanced Information Systems Engineering (σελ. 88–102). London, UK: Springer-Verlag.

[Αναφέρεται στη Σελ. 21]

Jones, M. B., & Regehr, J. (1999). CPU reservations and time constraints: implementation experience on windows NT. Στο *Proceedings of the 3rd conference on USENIX Windows NT Symposium - Volume 3* (σελ. 10–10). Berkeley, CA, USA: USENIX Association. Ανακτήθηκε από `http://dl.acm.org/citation.cfm?id=1268427.1268437`

[Αναφέρεται στη Σελ. 57]

Kavantzas, N., κ. συν. (2005). *Web Services Choreography Description Language, W3C Candidate Recommendation*. 9.

[Αναφέρεται στη Σελ. 29]

Keller, A., & Ludwig, H. (2003). The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *J. Netw. Syst. Manage.*, *11*(1), 57–81. doi: 10.1023/A:1022445108617

[Αναφέρεται στη Σελ. 139]

Kirschnick, J., Calero, J. M. A., Wilcock, L., & Edwards, N. (2010). Toward an architecture for the automated provisioning of cloud services. *Comm. Mag.*, *48*, 124–131.

[Αναφέρεται στη Σελ. 24]

Kolodner, E., Tal, S., Kyriazis, D., Naor, D., Allalouf, M., Bonelli, L., … Wolfsthal, Y. (2011). A Cloud Environment for Data-intensive Storage Services. Στο *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on* (σελ. 357 -366). doi: 10.1109/CloudCom.2011.55

[Αναφέρεται στη Σελ. 147], [Αναφέρεται στη Σελ. 148]

Korkhov, V., Vasyunin, D., Wibisono, A., Belloum, A. S. Z., Inda, M. A., Roos, M., ... Hertzberger, L. O. (2007). VLAM-G: Interactive data driven workflow engine for Grid-enabled resources. *Sci. Program.*, *15*, 173–188.

[Αναφέρεται στη Σελ. 16], [Αναφέρεται στη Σελ. 24]

Kouzes, R., Anderson, G., Elbert, S., Gorton, I., & Gracio, D. (2009). The Changing Paradigm of Data-Intensive Computing. *Computer*, *42*(1), 26 -34. doi: 10.1109/MC.2009.26

[Αναφέρεται στη Σελ. 147]

Kyriazis, D., Tserpes, K., Menychtas, A., Litke, A., & Varvarigou, T. (2008). An innovative workflow mapping mechanism for Grids in the frame of Quality of Service. *Future Gener. Comput. Syst.*, *24*, 498–511.

[Αναφέρεται στη Σελ. 21]

Lakshmanan, K., & Rajkumar, R. (2008). Distributed Resource Kernels: OS Support for End-To-End Resource Isolation. Στο *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08. IEEE* (σελ. 195 -204). doi: 10.1109/RTAS.2008.37

[Αναφέρεται στη Σελ. 62], [Αναφέρεται στη Σελ. 66]

Legrand, I., Newman, H., Voicu, R., Cirstoiu, C., Grigoras, C., Dobre, C., ... Stratan, C. (2009). MonALISA: An agent based, dynamic service system to monitor, control and optimize distributed systems. *Computer Physics Communications*, *180*(12), 2472 - 2498. doi: 10.1016/j.cpc.2009.08.003

[Αναφέρεται στη Σελ. 143]

Lin, C., Lu, S., Fei, X., Chebotko, A., Pai, D., Lai, Z., ... Hua, J. (2009). A Reference Architecture for Scientific Workflow Management Systems and the

VIEW SOA Solution. *IEEE Trans. Serv. Comput.*, *2*(1), 79-92.

[Αναφέρεται στη Σελ. 16]

Linthicum, D. S. (2009). *Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide* (1st έκδ.). Addison-Wesley Professional.

[Αναφέρεται στη Σελ. 7]

Lloyd, S., Gavaghan, D., Simpson, A., Mascord, M., Seneurine, C., Williams, G., ... Fabritiis, G. d. (2007). Integrative Biology - the challenges of developing a collaborative research environment for heart and cancer modelling. , *23*(3), 457-465–.

[Αναφέρεται στη Σελ. 31]

LogicMonitor. (α.χ.). Ανακτήθηκε από `http://www.logicmonitor.com/`

[Αναφέρεται στη Σελ. 146]

Ludascher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., ... Zhao, Y. (2006). Scientific workflow management and the Kepler system: Research Articles. *Concurrency and Computation: Practice and Experience*, *18*(10), 1039-1065.

[Αναφέρεται στη Σελ. 23]

Market and Technical Requirements Analysis, IRMOS Project. (2010). Ανακτήθηκε από `http://www.irmosproject.eu/Deliverables/Download.aspx?ID=15`

[Αναφέρεται στη Σελ. 15]

Martin-Flatin, J.-P., & Primet, P. V.-B. (2005). Guest Editorial: High-speed networks and services for data-intensive Grids: The DataTAG Project. *Future Generation Computer Systems*, *21*(4), 439–442. doi: 10.1016/j.future.2005.01.001

[Αναφέρεται στη Σελ. 144]

Massie, M. L., Chun, B. N., & Culler, D. E. (2004). The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, *30*(7), 817 - 840. doi: 10.1016/j.parco.2004.04.001

[Αναφέρεται στη Σελ. 143]

McDougall, R., & Mauro, J. (2006). *Solaris Internals: Solaris 10 and OpenSolaris Kernel Architecture*. Prentice Hall.

[Αναφέρεται στη Σελ. 70]

McGough, A. S., Akram, A., Guo, L., Krznaric, M., Dickens, L., Colling, D., … Tsanakas, P. (2007). GRIDCC: A Real-time Grid workflow system with QoS. *Sci. Program.*, *15*, 213–234.

[Αναφέρεται στη Σελ. 23]

McGough, S., Young, L., Afzal, A., Newhouse, S., & Darlington, J. (2004). Workflow Enactment in ICENI. Στο *UK e-Science All Hands Meeting* (σελ. –). Nottingham, UK: IOP Publishing Ltd, Bristol, UK.

[Αναφέρεται στη Σελ. 40]

Mckenney, P. E., Molnar, I., Sarma, D., & Bhattacharya, S. (2006). Extending RCU for Realtime and Embedded Workloads. *OLS2006*. Ανακτήθηκε από `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.145.2137`

[Αναφέρεται στη Σελ. 52]

McMahon, G., & Florian, M. (1975). On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness. , *23*(3), 475-482–. Ανακτήθηκε από `http://or.journal.informs.org/cgi/content/abstract/23/3/475`

[Αναφέρεται στη Σελ. 40]

Mell, P., & Grance, T. (2009). *The NIST Definition of Cloud Computing, Version 15.*

[Αναφέρεται στη Σελ. 5]

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing (Draft) Recommendations of the National Institute of Standards and Technology. *Nist Special Publication*, *145*(6), 7. Ανακτήθηκε από `http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf`

[Αναφέρεται στη Σελ. 6]

Mercer, C. W., Savage, S., & Tokuda, H. (1993). *Processor Capacity Reserves for Multimedia Operating Systems.* Pittsburgh, PA, USA.

[Αναφέρεται στη Σελ. 57], [Αναφέρεται στη Σελ. 65]

Nahrstedt, K., Chu, H.-h., & Narayan, S. (1998). QoS-aware resource management for distributed multimedia applications. *J. High Speed Netw.*, *7*, 229–257. Ανακτήθηκε από `http://dl.acm.org/citation.cfm?id=311274.311277`

[Αναφέρεται στη Σελ. 59]

Netto, M. A., Bubendorfer, K., & Buyya, R. (2007). *SLA-Based Advance Reservations with Flexible and Adaptive Time QoS Parameters.* Vienna, Austria: Springer-Verlag.

[Αναφέρεται στη Σελ. 38], [Αναφέρεται στη Σελ. 42]

Neubauer, F., Hoheisel, A., & Geiler, J. (2006). Workflow-based Grid applications. , *22*(1-2), 6-15–.

[Αναφέρεται στη Σελ. 31]

OASIS. (2006). *OASIS standard v2.0.4, ebXML Business Process Specification Schema Technical Specification v2.0.4.*

[Αναφέρεται στη Σελ. 29]

Oikawa, S., & Rajkumar, R. (1999). Portable RK: a portable resource kernel for guaranteed and enforced timing behavior. Στο *Real-Time Technology and Applications Symposium, 1999. Proceedings of the Fifth IEEE* (σελ. 111 -120). doi: 10.1109/RTTAS.1999.777666

[Αναφέρεται στη Σελ. 57]

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Carver, T., ... Li, P. (2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, *20*(17), 3045-3054.

[Αναφέρεται στη Σελ. 21]

Oinn, T., Greenwood, M., Addis, M., Alpdemir, M. N., Ferris, J., Glover, K., ... Wroe, C. (2006). Taverna: lessons in creating a workflow environment for the life sciences: Research Articles. , *18*(10), 1067-1100–.

[Αναφέρεται στη Σελ. 31]

Oldevik, J. (2006). *MOFScript User Guide.*

[Αναφέρεται στη Σελ. 92]

Palopoli, L., Cucinotta, T., Marzario, L., & Lipari, G. (2009). AQuoSA - adaptive quality of service architecture. *Software: Practice and Experience*, *39*(1), 1–31. Ανακτήθηκε από `http://dx.doi.org/10.1002/spe.883` doi: 10.1002/spe.883

[Αναφέρεται στη Σελ. 58], [Αναφέρεται στη Σελ. 61]

Papazoglou, M., Dells, A., Bouguettaya, A., & Haghjoo, M. (1997). Class library support for workflow environments and applications. *Computers,*

IEEE Transactions on, *46*(6), 673 -686.

[Αναφέρεται στη Σελ. 21]

Raines, G. (2009). Cloud computing and soa. *MITRE technical papers, MITRE Corp., Massachusetts, USA*.

[Αναφέρεται στη Σελ. 9]

Rajkumar, R., Juvva, K., Molano, A., & Oikawa, S. (1997). Resource kernels: a resource-centric approach to real-time and multimedia systems. Στο K. Jeffay, D. D. Kandlur, & T. Roscoe (Επιμ.), (τ. 3310, σελ. 150-164). SPIE. Ανακτήθηκε από `http://link.aip.org/link/?PSI/3310/150/1` doi: 10.1117/12.298417

[Αναφέρεται στη Σελ. 57]

Ranno, F., Shrivastava, S. K., & Wheater, S. M. (1997). *A System for Specifying and Coordinating the Execution of Reliable Distributed Aplications.*

[Αναφέρεται στη Σελ. 21]

Ravindran, B. (2002). Engineering dynamic real-time distributed systems: architecture, system description language, and middleware. *Software Engineering, IEEE Transactions on*, *28*(1), 30 -57. doi: 10.1109/32.979988

[Αναφέρεται στη Σελ. 61]

Ripoll, I., Pisa, P., Abeni, L., Gai, P., Lanusse, A., & Saez, . P. B., S. (2002). *OCERA project deliverable D1.1 RTOS state of the art analysis.*

[Αναφέρεται στη Σελ. 65]

Rodero-Merino, L., Vaquero, L. M., Gil, V., Galán, F., Fontán, J., Montero, R. S., & Llorente, I. M. (2010). From infrastructure delivery to service management in clouds. *Future Gener. Comput. Syst.*, *26*, 1226–1240.

[Αναφέρεται στη Σελ. 25]

Schantz, R. E., Loyall, J. P., Rodrigues, C., & Schmidt, D. C. (2006). Controlling quality-of-service in distributed real-time and embedded systems via adaptive middleware: Experiences with Auto-adaptive and Reconfigurable Systems. *Softw. Pract. Exper.*, *36*(11-12), 1189–1208. Ανακτήθηκε από `http://dx.doi.org/10.1002/spe.v36:11/12` doi: 10.1002/spe.v36:11/12

[Αναφέρεται στη Σελ. 60]

Schantz, R. E., Loyall, J. P., Rodrigues, C., Schmidt, D. C., Krishnamurthy, Y., & Pyarali, I. (2003). Flexible and adaptive QoS control for distributed real-time and embedded middleware. Στο *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware* (σελ. 374–393). New York, NY, USA: Springer-Verlag New York, Inc. Ανακτήθηκε από `http://dl.acm.org/citation.cfm?id=1515915.1515941`

[Αναφέρεται στη Σελ. 60]

Schmidt, D. C., Levine, D. L., & Mungee, S. (1998). The design of the TAO real-time object request broker. *Computer Communications*, *21*(4), 294 - 324. Ανακτήθηκε από `http://www.sciencedirect.com/science/article/pii/S0140366497001655` (<ce:title>Quality of Services in Distributed Systems</ce:title>) doi: 10.1016/S0140-3664(97)00165-5

[Αναφέρεται στη Σελ. 60]

Schmidt, K.-U., Anicic, D., & Stühmer, R. (2008). Event-driven Reactivity: A Survey and Requirements Analysis. Στο *SBPM2008: 3rd international Workshop on Semantic Business Process Management in conjunction with the 5th European Semantic Web Conference (ESWC'08)*. CEUR Workshop Proceedings (CEUR-WS.org, ISSN 1613-0073).

[Αναφέρεται στη Σελ. 140]

Schubert, L. (2010). *The Future Of Cloud Computing, Opportunities for European Cloud Computing Beyond 2010.* Ανακτήθηκε από `http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf`

[Αναφέρεται στη Σελ. 6]

Schwiegelshohn, U., Yahyapour, R., & Wieder, P. (2006). Resource Management for Future Generation Grids. Στο V. Getov, D. Laforenza, & A. Reinefeld (Επιμ.), *Future Generation Grids* (σελ. 99-112–). Springer US. Ανακτήθηκε από `http://dx.doi.org/10.1007/978-0-387-29445-2_6`

[Αναφέρεται στη Σελ. 37]

SEKT. (α.χ.). *Semantic Knowledge Technologies.* Ανακτήθηκε από `http://www.sekt-project.com/`

[Αναφέρεται στη Σελ. 30]

Setz, J. (2007). *Inter-process communication in RTAI and RTLinux* (Seminar Report). Saarland University.

[Αναφέρεται στη Σελ. 63]

Sha, L., Abdelzaher, T., Arzen, K.-E., Cervin, A., Baker, T., Burns, A., ... Mok, A. K. (2004). Real Time Scheduling Theory: A Historical Perspective. *Real-Time Syst.*, *28*, 101–155.

[Αναφέρεται στη Σελ. 10], [Αναφέρεται στη Σελ. 55]

Shankaran, N., Koutsoukos, X. D., Schmidt, D. C., Xue, Y., & Lu, C. (2006). Hierarchical Control of Multiple Resources in Distributed Real-time and Embedded Systems. Στο *Proceedings of the 18th Euromicro Conference on Real-Time Systems* (σελ. 151–160). Washington, DC, USA: IEEE Computer Society. Ανακτήθηκε από `http://dl.acm.org/citation.cfm?id=1153916.1153945` doi: 10.1109/ECRTS.2006.11

[Αναφέρεται στη Σελ. 60]

Sharfman, I., Schuster, A., & Keren, D. (2007). A geometric approach to monitoring threshold functions over distributed data streams. *ACM Transactions on Database Systems*, *32*(4). doi: 10.1145/1292609.1292613

[Αναφέρεται στη Σελ. 142]

Sild, S., Maran, U., Romberg, M., Schuller, B., & Benfenati, E. (2005). OpenMolGRID: Using Automated Workflows in GRID Computing Environment. Στο P. M. A. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld, & M. Bubak (Επιμ.), *Lecture Notes in Computer Science* (τ. 3470, σελ. 464-473–). Springer Berlin / Heidelberg. Ανακτήθηκε από `http://dx.doi.org/10.1007/11508380_48`

[Αναφέρεται στη Σελ. 31]

Singh, G., Kesselman, C., & Deelman, E. (2006). Application-Level Resource Provisioning on the Grid. Στο *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing* (σελ. 83–). IEEE Computer Society.

[Αναφέρεται στη Σελ. 42]

Soonwook, H., & Kesselman, C. (2003). Grid workflow: a flexible failure handling framework for the grid. Στο *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on* (σελ. 126-137–). Ανακτήθηκε από `10.1109/HPDC.2003.1210023`

[Αναφέρεται στη Σελ. 35]

SUPER. (α.χ.). *Semantics Utilized for Process management within and between Enterprises.* Ανακτήθηκε από `http://www.ip-super.org/`

[Αναφέρεται στη Σελ. 30]

Taylor, I., Shields, M., Wang, I., & Harrison, A. (2007). The Triana Workflow Environment: Architecture and Applications. Στο I. J. Taylor, E. Deelman, D. B. Gannon, & M. Shields (Επιμ.), *Workflows for e-Science* (σελ. 320-339–). Springer London. Ανακτήθηκε από `http://dx.doi.org/10.1007/978-1-84628-757-2_20`

[Αναφέρεται στη Σελ. 31]

Tierney, B., Aydt, R., Gunter, D., Smith, W., Swany, M., Taylor, V., & Wolski, R. (2002). *A Grid Monitoring Architecture.*

[Αναφέρεται στη Σελ. 145]

Tokuda, H., Nakajima, T., & Rao, P. (1990). Real-Time Mach: Towards a Predictable Real-Time System. Στο *USENIX MACH Symposium'90* (σελ. 73-82).

[Αναφέρεται στη Σελ. 57]

Topcuoglu, H., Hariri, S., & Min-You, W. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. , *13*(3), 260-274–. Ανακτήθηκε από `10.1109/71.993206`

[Αναφέρεται στη Σελ. 39]

UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. (2009). Ανακτήθηκε από `http://www.omg.org/spec/MARTE/1.0/PDF/`

[Αναφέρεται στη Σελ. 83]

UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification v1.1. (2008). Ανακτήθηκε από `http://www.omg.org/spec/QFTP/1.1/PDF`

[Αναφέρεται στη Σελ. 83]

Valls, M., Alonso, A., Ruiz, J., & Groba, A. (2003). An Architecture of a

Quality of Service Resource Manager Middleware for Flexible Embedded Multimedia Systems. Στο A. Coen-Porisini & A. van der Hoek (Επιμ.), *Software Engineering and Middleware* (τ. 2596, σελ. 36-55). Springer Berlin / Heidelberg. Ανακτήθηκε από `http://dx.doi.org/10.1007/3-540-38093-0_3`

[Αναφέρεται στη Σελ. 59]

Van der Aalst, W. (2011). *Workflow Patterns.* Ανακτήθηκε από `http://www.workflowpatterns.com`

[Αναφέρεται στη Σελ. 29]

Vardhan, V., Sachs, D. G., Yuan, W., Harris, A. F., Adve, S. V., Jones, D. L., ... Nahrstedt, K. (2005). Integrating finegrain application adaptation with global adaption for saving energy. Στο *Proceedings of the 2nd International Workshop on Powe-Aware Real-Time Computing (PARC.*

[Αναφέρεται στη Σελ. 58]

Voulodimos, A., Gogouvitis, S. V., Mavrogeorgi, N., Talyansky, R., Kyriazis, D., Koutsoutos, S., ... Varvarigou, T. (2011). A Unified Management Model for Data Intensive Storage Clouds. Στο *Network Cloud Computing and Applications, International Symposium on* (τ. 0, σελ. 69-72). doi: http://doi.ieeecomputersociety.org/10.1109/NCCA.2011.18

[Αναφέρεται στη Σελ. 151]

Waldo, J. (1999). The Jini architecture for network-centric computing. *Communications of the ACM*, *42*(7), 76–82. doi: 10.1145/306549.306582

[Αναφέρεται στη Σελ. 143]

Weber, M., Partsch, G., Scheller-Huoy, A., Schweitzer, J., & Schneider, G. (1997). Flexible Real-time Meeting Support for Workflow Management Systems. Στο

Proceedings of the 30th Hawaii International Conference on System Sciences: Information Systems Track-Collaboration Systems and Technology - Volume 2 (σελ. 559–). Washington, DC, USA: IEEE Computer Society.

[Αναφέρεται στη Σελ. 26]

Web Service Modeling Language WSML. (2008). ESSI WSML working group. Ανακτήθηκε από `http://www.wsmo.org/wsml`

[Αναφέρεται στη Σελ. 30]

Web Service Modeling Ontology WSMO. (2005). W3C. Ανακτήθηκε από `http://www.w3.org/Submission/WSMO/`

[Αναφέρεται στη Σελ. 30]

Web Services Business Process Execution Language Version 2.0. (2007). OASIS.

[Αναφέρεται στη Σελ. 29]

Web Services Choreography Description Language, Version 1.0, W3C candidate recommendation. (2005). W3C.

[Αναφέρεται στη Σελ. 29]

Weinhardt, C., Anandasivam, A., Blau, B., & Stosser, J. (2009). Business Models in the Service World. *IT Professional*, *11*(2), 28 -33.

[Αναφέρεται στη Σελ. 6]

Wieczorek, M., Siddiqui, M., Villazon, A., Prodan, R., & Fahringer, T. (2006). Applying Advance Reservation to Increase Predictability of Workflow Execution on the Grid. Στο *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing* (σελ. 82–). IEEE Computer Society.

[Αναφέρεται στη Σελ. 39]

Williamson, A. (2010). *Has Amazon EC2 become over subscribed.* `http://alan`

.blog-city.com/has_amazon_ec2_become_over_subscribed.htm.

[Αναφέρεται στη Σελ. 13]

Windows Azure. (α.χ.). Ανακτήθηκε από https://www.windowsazure.com/en-us/

[Αναφέρεται στη Σελ. 146]

WS-SecureConversation 1.3. (2007). Ανακτήθηκε από http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf

[Αναφέρεται στη Σελ. 94]

XML Process Definition Language (XPDL). (2008). Workflow Management Coalition. Ανακτήθηκε από www.wfmc.org/Download-document/WFMC-TC-1025-Oct-10-08-A-Final-XPDL-2.1-Specification.html

[Αναφέρεται στη Σελ. 29]

Yuan, W., & Nahrstedt, K. (2003). Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. *SIGOPS Oper. Syst. Rev.*, *37*, 149–163. Ανακτήθηκε από http://doi.acm.org/10.1145/1165389.945460 doi: http://doi.acm.org/10.1145/1165389.945460

[Αναφέρεται στη Σελ. 57]

Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., & Chang, H. (2004). QoS-aware middleware for Web services composition. *Software Engineering, IEEE Transactions on*, *30*(5), 311 - 327.

[Αναφέρεται στη Σελ. 21]

Zeromq. (α.χ.). Ανακτήθηκε από http://www.zeromq.org/

[Αναφέρεται στη Σελ. 152]

Zhang, R., Lu, C., Abdelzaher, T., & Stankovic, J. (2002). ControlWare:

a middleware architecture for feedback control of software performance. Στο *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on* (σελ. 301 - 310). doi: 10.1109/ICDCS.2002.1022267

[Αναφέρεται στη Σελ. 59]

Zhang, Y., Huang, G. Q., Qu, T., & Ho, O. (2010). Agent-based workflow management for RFID-enabled real-time reconfigurable manufacturing. *Int. J. Comput. Integr. Manuf.*, *23*, 101–112.

[Αναφέρεται στη Σελ. 26]

Zhang, Y., Koelbel, C., & Cooper, K. (2009). Hybrid Re-scheduling Mechanisms for Workflow Applications on Multi-cluster Grid. Στο *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (σελ. 116–123). Washington, DC, USA: IEEE Computer Society.

[Αναφέρεται στη Σελ. 25]

Κατάλογος Δημοσιεύσεων του Συγγραφέα

Επιμέλεια Εκδόσεων Συλλογικών Τόμων

[E1] Data Intensive Storage Services for Cloud Environments, Edited by Dimosthenis Kyriazis, Athanasios Voulodimos, Spyridon V. Gogouvitis and Theodora Varvarigou, IGI Global, 2013, ISBN13: 9781466639348, doi:10.4018/978-1-4666-3934-8

Δημοσιεύσεις σε Διεθνή Επιστημονικά Περιοδικά με Κρίση

[J1] Spyridon V. Gogouvitis, Kleopatra Konstanteli, Stefan Waldschmidt, George Kousiouris, Gregory Katsaros, Andreas Menychtas, Dimosthenis Kyriazis and Theodora Varvarigou, "Workflow management for soft real-time interactive applications in virtualized environments," Future Generation Computer Systems, Volume 28, Issue 1, 2012, Pages 193-209, ISSN 0167-739X.

[J2] Michael C. Jaeger, Alberto Messina, Mirko Lorenz, Spyridon V. Gogouvitis, Dimosthenis Kyriazis, Elliot K. Kolodner, Xiaomeng Su, Enver Bahar, "Content-centric storage - accessing data objects based on metadata and relationships," Informatyka Ekonomiczna, Issue no.3 (25), 2012, ISSN 1507-3858.

[J3] Tommaso Cucinotta, Fabio Checconi, George Kousiouris, Kleopatra Konstanteli, Spyridon V. Gogouvitis, Dimosthenis Kyriazis, Theodora A. Varvarigou, Alessandro Mazzetti, Zlatko Zlatev, Juri Papay, Michael Boniface, Soren Berger, Dominik Lamp, Thomas Voith, Manuel Stein, " Virtualised e-Learning on the IRMOS real-time Cloud," Service Oriented Computing and Applications, Volume 6, Issue 2, Pages 151-166, 2012, ISSN 1863-2386.

[J4] George Kousiouris, Andreas Menychtas, Dimosthenis Kyriazis, Kleopatra Konstanteli, Spyridon Gogouvitis, Gregory Katsaros, Theodora Varvarigou, "Parametric Design and Performance Analysis of a Decoupled Service-Oriented Prediction Framework based on Embedded Numerical Software", IEEE Transactions on Services Computing, 2012.

[J5] George Kousiouris, Andreas Menychtas, Dimosthenis Kyriazis, Spyridon Gogouvitis, Theodora Varvarigou, "Dynamic, behavioral-based estimation of resource provisioning based on high-level application terms in Cloud platforms", Future Generation Computer Systems, ISSN 0167-739X.

[J6] Alysson Bessani, Rudiger Kapitza, Dana Petcu, Paolo Romano, Spyridon V. Gogouvitis, Dimosthenis Kyriazis and Roberto G. Cascella, "A look to

the old-world sky: EU-funded dependability cloud computing research," SIGOPS Oper. Syst. Rev., Volume 46 Issue 2, July 2012, Pages 43-56 , ISSN 0163-5980.

[J7] Gregory Katsaros, George Kousiouris, Spyridon V. Gogouvitis, Dimosthenis Kyriazis, Andreas Menychtas, Theodora Varvarigou "A Self-adaptive hierarchical monitoring mechanism for Clouds," Journal of Systems and Software, Available online 8 December 2011, ISSN 0164-1212, 10.1016/j.jss.2011.11.1043

Κεφάλαια σε Συλλογικούς Τόμους με Κρίση

[B1] Beniamino Di Martino, Salvatore Venticinque, Dimosthenis Kyriazis, and Spyridon V. Gogouvitis. "A Comparison of two Different Approaches to Cloud Monitoring," Inter-cooperative Collective Inteligence: Techniques and Applications, Series in Studies in Computational Intelligence, Vol. 495, 2014.

[B2] Spyridon V. Gogouvitis, Athanasios Voulodimos and Dimosthenis Kyriazis. "Commercial and Distributed Storage Systems," Data Intensive Storage Services for Cloud Environments. IGI Global, 2013. doi:10.4018/978-1-4666-3934-8.ch001

[B3] Nikoletta Mavrogeorgi, Spyridon V. Gogouvitis, Athanasios Voulodimos and Vasilios Alexandrou. "SLA Management in Storage Clouds," Data Intensive Storage Services for Cloud Environments. IGI Global, 2013. doi:10.4018/978-1-4666-3934-8.ch006

[B4] Spyridon V. Gogouvitis, Kleopatra Konstanteli, Dimosthenis Kyriazis, Gregory Katsaros, Tommaso Cucinotta and Michael Boniface. "Workflow Management Systems in Distributed Environments," Achieving Real-Time in Distributed Computing: From Grids to Clouds. IGI Global, 2012 ISBN13: 9781609608279, pp. 115-132. doi:10.4018/978-1-60960-827-9.ch007.

[B5] Tommaso Cucinotta and Spyridon V. Gogouvitis. "Real-Time Attributes in Operating Systems," Achieving Real-Time in Distributed Computing: From Grids to Clouds. IGI Global, 2012, ISBN13: 9781609608279, pp. 275-287. doi:10.4018/978-1-60960-827-9.ch015.

[B6] Elliot K. Kolodner, Alexandra Shulman-Peleg, Dalit Naor, Per Brand, Michel Dao, Albert Eckert, Spyridon V. Gogouvitis, Danny Harnik, Michael C. Jaeger, Dimosthenis P. Kyriazis, Mirko Lorenz, Alberto Messina, Aidan Shribman, Sivan Tal, Athanasios S. Voulodimos, Yaron Wolfsthal. "Data-intensive Storage Services on Clouds: Limitations, Challenges and Enablers," European Research Activities in Cloud Computing, Cambridge Scholars Publishing, 2012

[B7] Stefanos Koutsoutos, Spyridon Gogouvitis, Dimosthenis Kyriazis, Theodora Varvarigou. "Monitoring in Federated & Self-Manageable Clouds," Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice. IGI Global, 2012. doi:10.4018/978-1-4666-1631-8.ch007

Δημοσιεύσεις σε Διεθνή Επιστημονικά Συνέδρια με Κρίση στο Πλήρες Κείμενο

[C1] Manolis Sardis, Spyridon V. Gogouvitis, Thanassis Bouras, Panagiotis Gouvas, Theodora Varvarigou, "Secure Enterprise Interoperability Ontology for Semantic Integration of Business to Business Applications", Eight International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2013

[C2] Nikoletta Mavrogeorgi, Vassilios Alexandrou, Spyridon Gogouvitis, Athanasios Voulodimos, Dimosthenis Kiriazis, Theodora Varvarigou, Elliot K. Kolodner, "Customized SLAs In Cloud Environments", Eight International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2013

[C3] Dimosthenis Kyriazis, Andrew Kapsalis, Konstantinos Kostantos, Spyridon V. Gogouvitis, Theodora Varvarigou, "QoS-Oriented Service Management in Large Scale Federated Clouds", Workshop on Management of Cloud Systems (MoCS 2013), 2013.

[C4] Nikoletta Mavrogeorgi, Vasileios Alexandrou, Spyridon Gogouvitis, Athanasios Voulodimos, Theodora Varvarigou, Alexandra Shulman-Peleg, Elliot K. Kolodner, "Dynamic Rule Based SLA Management In Clouds", IEEE CLOUD 2013

[C5] Michael C. Jaeger, Spyridon Gogouvitis, D. Kiriazis, Alberto Messina, Uwe Hohenstein, Elliot K. Kolodner, Enver Bahar, "Evolution of the CCI",

CLOSER 2013

[C6] Nikoletta Mavrogeorgi, Spyridon Gogouvitis, Athanasios Voulodimos, Vassilios Alexandrou, Dimosthenis Kiriazis, Athanasia Evangelinou, Theodora Varvarigou, Elliot K. Kolodner, "SLA Management in Clouds", CLOSER 2013

[C7] Spyridon V. Gogouvitis, Vassileios Alexandrou, Nikoletta Mavrogeorgi, Stefanos Koutsoutos, Dimosthenis Kyriazis and Theodora Varvarigou, "A Monitoring Mechanism for Storage Clouds", Second International Conference on Cloud and Green Computing (CGC 2012), November 2012.

[C8] Spyridon V. Gogouvitis, Gregory Katsaros, Dimosthenis Kyriazis, Athanasios Voulodimos, Roman Talyansky and Theodora Varvarigou, "Retrieving, Storing, Correlating and Distributing Information for Cloud Management", 9th International Conference on Economics of Grids, Clouds, Systems, and Services, November 2012.

[C9] Michael C. Jaeger, Alberto Messina, Mirko Lorenz, Spyridon V. Gogouvitis, Dimosthenis Kyriazis, Elliot K. Kolodner, Xiaomeng Su, Enver Bahar, "Cloud-Based Content Centric Storage for Large Systems," Federated Conference on Computer Science and Information Systems, Wroclaw, Poland, September 9-12, 2012.

[C10] Spyridon Gogouvitis, Michael C. Jaeger, Hillel Kolodner, Dimosthenis Kyriazis, Fransesco Longo, Mirko Lorenz, Alberto Messina, Maurizio Montagnuolo, Eliot Salant, Fransesco Tusa, "VISION Cloud: A Cloud

Storage Solution Supporting Modern Media Production," IBC2012, RAI, Amsterdam, September, 2012.

[C11] Nikoletta Mavrogeorgi, Spyridon Gogouvitis, Athanasios Voulodimos, Gregory Katsaros, Stefanos Koutsoutos, Dimosthenis Kiriazis, Theodora Varvarigou, Elliot K. Kolodner, "Content Based SLAs in Cloud Computing Environments," 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), Hawaii, USA, June 24-29, 2012.

[C12] Gregory Katsaros, Spyridon Gogouvitis, Nikoletta Mavrogeorgi, Athanasios Voulodimos ,Dimosthenis Kiriazis, Theodora Varvarigou, Roman Talyansky, "A Holistic View of Information Management in Cloud Environments," 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), Hawaii, USA, June 24-29, 2012.

[C13] Athanasios Voulodimos, Spyridon Gogouvitis, Nikoletta Mavrogeorgi, Roman Talyansky, Dimosthenis Kyriazis, Stefanos Koutsoutos, Vasileios Alexandrou,Elliot Kolodner, Per Brand, Theodora Varvarigou, "A Unified Management Model for Data Intensive Storage Clouds," IEEE First International Symposium on Network Cloud Computing and Applications, Toulouse, France, November 21-23, 2011.

[C14] Spyridon V. Gogouvitis, George Kousiouris, George Vafiadis, Hillel Kolodner, Dimosthenis Kyriazis, "OPTIMIS and VISION Cloud: How to manage data in clouds," Cloud Computing: Project and Initiatives - 2011. Collocated with: Euro-Par 2011 International Conference, Bordeaux, France, August 30th 2011.

[C15] Hillel Kolodner, Dalit Naor, Sivan Tal, Stefanos Koutsoutos, Nikoletta Mavrogeorgi, Spyridon Gogouvitis, Dimosthenis Kyriazis, Eliot Salant, "Data-intensive Storage Services on Clouds: Limitations, Challenges and Enablers," eChallenges e-2011 Conference, Florence, Italy, October 27th 2011.

[C16] Tommaso Cucinotta, Spyridon Gogouvitis, Kleopatra Kostanteli, "SLAs in Virtualized Cloud Computing Infrastructures with QoS Assurance," in Proceedings of the International Workshop on eContracting in the Clouds, co-located with the eChallenges 2011 Conference, Florence, Italy, October 27th 2011.

[C17] Andreas Menychtas, Dimosthenis Kyriazis, Spyridon V. Gogouvitis, Karsten Oberle, Thomas Voith, Georgina Gallizo, Soren Berger, Eduardo Oliveros, Mike J. Boniface, "A Cloud Platform for Real-time Interactive Applications," CLOSER 2011, 2011.

[C18] Elliot K. Kolodner, Sivan Tal, Dimosthenis Kyriazis, Dalit Naor, Miriam Allalouf, Lucia Bonelli, Per Brand, Albert Eckert, Erik Elmroth, Spyridon V. Gogouvitis, Danny Harnik, Francisco Hernandez, Michael C. Jaeger, Ewnetu Bayuh Lakew, Jose Manuel Lopez, Mirko Lorenz, Alberto Messina, Alexandra Shulman-Peleg, Roman Talyansky, Athanasios Voulodimos, Yaron Wolfsthal, "A Cloud Environment for Data-intensive Storage Services," to appear in Third IEEE International Conference on Cloud Computing Technology and Science (IEEE CloudCom 2011), 2011.

[C19] Athanasios S. Voulodimos, Dimosthenis P. Kyriazis, Spyridon V. Gogouvitis,

Anastasios D. Doulamis, Dimitrios I. Kosmopoulos, Theodora A. Varvarigou, “QoS-oriented Service Management in Clouds for Large Scale Industrial Activity Recognition," Third International Conference of Soft Computing and Pattern Recognition (SoCPaR 2011), 2011.

[C20] George Kousiouris, Dimosthenis Kyriazis, Spyridon V. Gogouvitis, Andreas Menychtas, Kleopatra Konstanteli and Theodora A. Varvarigou, “Translation of Application-level Terms to Resource-level attributes across the Cloud Stack Layers," Workshop on Management of Cloud Systems (MoCS 2011), in conjunction with the 16th IEEE Symposium on Computers and Communications (ISCC 2011), 2011.

[C21] Spyridon V. Gogouvitis, Kleopatra Konstanteli, Dimosthenis Kyriazis, Theodora Varvarigou, “An Architectural Approach for Event-Based Execution Management in Service Oriented Infrastructures," 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010.

[C22] Spyridon V. Gogouvitis, Kleopatra Konstanteli, George Kousiouris, Gregory Katsaros, Dimosthenis Kyriazis, Theodora Varvarigou, “A Service Oriented Architecture for achieving QoS-aware Workflow Management in Virtualized Environments," 6th International Conference on Network and Service Management (CNSM 2010), Niagra Falls, Canada, October 2010.

[C23] Georgina Gallizo, Roland Kubert, Karsten Oberle, Klaus Satzke, Spyridon V. Gogouvitis, Gregory Katsaros, and Eduardo Oliveros, “A Service Level Agreement Management Framework for Real-time Applications in Cloud

Computing Environments," 2nd International ICST Conference on Cloud Computing (CloudComp 2010) Barcelona, Spain October 26 - 28, 2010

[C24] George Kousiouris, Dimosthenis Kyriazis, Kleopatra Konstanteli, Spyridon Gogouvitis, Gregory Katsaros and Theodora Varvarigou, "A Service-Oriented Framework for GNU Octave-Based Performance Prediction," IEEE Services Computing Conference (SCC) 2010, July 5-10, Miami, USA.

[C25] Michael Boniface, Bassem Nasser, Juri Papay, Stephen C. Phillips, Arturo Servin, Xiaoyu Yang, Zlatko Zlatev, Spyridon V. Gogouvitis, Gregory Katsaros, Kleopatra Konstanteli, George Kousiouris, Andreas Menychtas, Dimosthenis Kyriazis, "Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds," iciw, pp.155-160, 2010 Fifth International Conference on Internet and Web Applications and Services, 2010.

[C26] Gregory Katsaros, George Kousiouris, Spyridon V. Gogouvitis, Dimosthenis Kyriazis, Theodora A. Varvarigou, "A service oriented monitoring framework for soft real-time applications," IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2010), Perth, Australia, December 2010.

[C27] Theodoros Polychniatis, Andreas Menychtas, Spyridon V. Gogouvitis, George Kousiouris, Kleopatra Konstanteli, Dimosthenis Kyriazis, Theodora Varvarigou, "Framework Services for Real-time SOIs", EGEE'08 Conference, Harbiye Askeri Museum, Istanbul - Turkey, 22-26 September,

2008.

[C28] Spyridon V. Gogouvitis, George Kousiouris, Kleopatra Konstanteli, Theodoros Polychniatis, Andreas Menychtas, Dimosthenis Kyriazis, Theodora A. Varvarigou, "Realtime-enabled workflow management in service oriented infrastructures," Proceedings of the 1st ACM Workshop on Analysis and Retrieval of Events/Actions and Workflows in Video Streams, AREA 2008, Vancouver, British Columbia, Canada, October 31, 2008.

Άρθρα υπό Κρίση

[UR1] Stefanos Koutsoutos, Gregory Katsaros, Spyridon Gogouvitis, Theodora Varvarigou and Georgios Kampourakis, "An Architecture for an Efficient and Reprogrammable Cloud Monitoring supporting high level Decision Making Algorithms," The Journal of Systems and Software.

[UR2] P. Brand, S. Dippl, A. Giefer, S. Gogouvitis, E. K. Kolodner, M. Lorenz, A. Messina, M. Montagnuolo, M. Neumann, E. Salant, "A Novel Approach At Producing Media In The Cloud Efficiently And Securely," IBC2013.

Βιογραφικό Σημείωμα

Ο Σπυρίδων Β. Γωγουβίτης γεννήθηκε την 23η Απριλίου 1982 στην Αθήνα. Αποφοίτησε από το Brussels European School I το 2000 με βαθμό διπλώματος 92,84/100. Το 2006 αποφοίτησε από την Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου με βαθμό πτυχίου 7,74 «(Λίαν Καλώς»). Το θέμα της διπλωματικής εργασίας που εκπόνησε ήταν «Μελέτη Θεωρίας Κοινωνικών Δικτύων και Δυνατότητες Εφαρμογής τους στα Ad Hoc δίκτυα» υπό την επίβλεψη του Καθηγητή του Εθνικού Μετσόβιου Πολυτεχνείου κ. Ευστάθιου Δ. Συκά και βαθμολογήθηκε με γενικό χαρακτηρισμό «Άριστα» (10). Μετά την αποφοίτησή του εκπλήρωσε τις στρατιωτικές του υποχρεώσεις ως Αναλυτής - Προγραμματιστής στο Κέντρο Πληροφορικής Υποστήριξης Ελληνικού Στρατού (ΚΕΠΥΕΣ) του ΥΕΘΑ. Τον Οκτώβριο του 2008, έγινε δεκτός για μεταπτυχιακές σπουδές που οδηγούν στην απόκτηση Διδακτορικού Διπλώματος στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου.

Κατά το παρελθόν εργάστηκε στον ιδιωτικό τομέα ως μηχανικός λογισμικού. Κατά τη διάρκεια της εκπόνησης της Διδακτορικής του Διατριβής ο Σ. Γωγουβίτης εργάστηκε σε Ελληνικά και Ευρωπαϊκά Προγράμματα, στα οποία του δόθηκε η δυνατότητα να εμβαθύνει σε ερευνητικά θέματα, άμεσα συνδεδεμένα με την περιοχή του διδακτορικού του. Έλαβε μέρος σε πολλές επιστημονικές συναντήσεις στην Ελλάδα και στο εξωτερικό και συνεργάστηκε με μηχανικούς και ανώτερα στελέχη διαφόρων οργανισμών, εταιριών

υπολογιστικών συστημάτων και ερευνητικών πανεπιστημιακών ομάδων. Σε ένα εκ των Ευρωπαϊκών προγραμμάτων αυτών ήταν συντονιστής και τεχνικός υπεύθυνος πακέτου εργασίας που σχετίζεται με την διαχείριση ενός περιβάλλοντος Νέφους. Έχει εκτελέσει σειρά από παρουσιάσεις και σεμινάρια για ζητήματα διαχείρισης κατανεμημένων συστημάτων (όπως συστήματα Πλέγματος και Νέφους) καθώς και ζητήματα διαχείρισης επιχειρηματικών διαδικασιών και ροών εργασίας. Τα αποτελέσματα της ερευνητικής του εργασίας παρουσιάστηκαν σε διεθνή συνέδρια και δημοσιεύθηκαν στον επιστημονικό Τύπο, σε περιοδικά και βιβλία.

Ο Σ. Γωγουβίτης είναι μέλος του Τεχνικού Επιμελητηρίου της Ελλάδος (ΤΕΕ) από το 2007.