ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ένταξη Σημασιολογίας Δικτύων Διασύνδεσης Υψηλής Επίδοσης σε Εικονικές Μηχανές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Ελισάβετ Γ. Κοζύρη

Επιβλέπων: Νεκτάριος Γ. Κοζύρης
Αν. Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2010

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ένταξη Σημασιολογίας Δικτύων Διασύνδεσης Υψηλής Επίδοσης σε Εικονικές Μηχανές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Ελισάβετ Γ. Κοζύρη

Επιβλέπων: Νεκτάριος Γ. Κοζύρης
Αν. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 6 Ιουλίου 2010.

.....	.....	.....
Νεκτάριος Κοζύρης	Κωστής Σαγώνας	Νικόλαος Παπασπύρου
Αν. Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2010

.....
Ελισάβετ Γ. Κοζύρη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

Περίληψη

Οι συστοιχίες εικονικών μηχανών αποτελούν την εναλλακτική λύση των συστοιχιών πραγματικών υπολογιστών, καθώς επιτρέπουν την εκτέλεση πολυνηματικών επιστημονικών εφαρμογών, αλλά συγχρόνως παρέχουν τη δυνατότητα αποδοτικότερου, δικαιότερου και οικονομικότερου διαμοιρασμού του υλισμικού ανάμεσα στις εφαρμογές που μπορεί να εκτελούνται ταυτόχρονα σε ένα κέντρο δεδομένων. Το μειονέκτημα που παραμένει στις συστοιχίες εικονικών μηχανών είναι η επιβάρυνση στον χρόνο εκτέλεσης των εφαρμογών που χρησιμοποιούν συσκευές Εισόδου/Εξόδου, καθώς προστίθενται επίπεδα εικονικοποίησης μεταξύ της εφαρμογής και του υλισμικού. Το αντικείμενο της διπλωματικής ανήκει στο ευρύτερο επιστημονικό πεδίο της εκτέλεσης HPC (High Performance Computing) εφαρμογών σε συστοιχίες εικονικών μηχανών και εστιάζει στην αποδοτικότερη μεταφορά δεδομένων μεταξύ των εικονικών μηχανών και των συσκευών Ε/Ε. Πιο συγκεκριμένα, η παρούσα εργασία πραγματεύεται τη σχεδίαση και υλοποίηση ενός μηχανισμού με τον οποίο τα δίκτυα διασύνδεσης υψηλής επίδοσης εντάσσονται σε εικονικά περιβάλλοντα.

Το εικονικό περιβάλλον στο οποίο βασίζεται αυτός ο μηχανισμός δημιουργείται από τον ελεγκτή εικονικής μηχανής Xen. Σ'αυτό το περιβάλλον προσαρμόστηκε το δίκτυο διασύνδεσης που υλοποιήθηκε κατά τη διάρκεια της διπλωματικής εργασίας "Σχεδίαση και Υλοποίηση μηχανισμού απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10GbE", στο Εργαστήριο Υπολογιστικών Συστημάτων. Το πρωτόκολλο του δικτύου είναι το SLURPoE (Simple Lightweight RDMA Protocol over Ethernet), το οποίο εικονικοποιήθηκε στην παρούσα διπλωματική έτσι, ώστε να ενταχθεί στο εικονικό περιβάλλον του Xen.

Η εικονικοποίηση του πρωτοκόλλου SLURPoE υλοποιήθηκε μέσω της δημιουργίας ενός διαχωρισμένου οδηγού (split driver), όπου το κάτω μισό (backend) τοποθετείται στη πρωτεύουσα εικονική μηχανή και το πάνω μισό (frontend) σε μια φιλοξενούμενη εικονική μηχανή. Διατηρώντας την αρχιτεκτονική του δικτύου και διαχωρίζοντας τις λειτουργίες του SLURPoE στο πάνω και στο κάτω μισό του οδηγού, διατηρούνται τα βασικά πλεονεκτήματα του πρωτοκόλλου, όπως η απευθείας μεταφορά δεδομένων από τον χώρο χρήστη στην

κάρτα δικτύου και αντιστρόφως, χωρίς την παρεμβολή του λειτουργικού συστήματος και του ελεγκτή εικονικής μηχανής.

Με χρήση ενός απλού μετρο-προγράμματος (micro-benchmark) αξιολογήθηκε μερικώς η απόδοση του πρωτοκόλλου μέσα σε εικονικό περιβάλλον. Το πρωτόκολλο εμφανίζει μια σταθερή επιβάρυνση 12μs όσον αφορά στο χρόνο απόκρισης, ενώ για μεγάλα μηνύματα (>512KB) εμφανίζει αποτελέσματα σχεδόν όμοια με τα αντίστοιχα σε μη εικονικό περιβάλλον.

Λέξεις Κλειδιά

Εικονική μηχανή, Δίκτυο διασύνδεσης υψηλής επίδοσης, Επικοινωνία επιπέδου χώρου χρήστη, Ελεγκτής εικονικής μηχανής, Διαχωρισμένος οδηγός, Xen, SLURPoE, HPC

Abstract

Today, with the advent of Virtualization techniques, Virtual Machine (VM) environments are becoming a great trend, providing flexibility, dedicated execution and isolation to a vast number of services. These infrastructures, built on clusters of multicores, offer huge processing power, ideal for mass deployment of compute-intensive applications. However, bridging the gap between I/O retrieval techniques in Virtualized environments and application demands seems to be a major challenge. The objective of this study is the design and implementation of a mechanism integrating already existing High-performance Interconnection semantics into Virtualized environments.

The Virtual environment used in this thesis is based on Xen's virtualization platform. The High-performance interconnection network integrated in this environment was developed in the Computing Systems Laboratory during a previous diploma thesis entitlted: "Design and implementation of an RDMA mechanism over programmable 10GbE Interfaces". Its protocol, SLURPoE (Simple Lightweight RDMA Protocol over Ethernet), is virtualized using software layers withing the Virtual Machine Monitor (VMM), implemented as a split driver model.

These specific layers host a back-end driver that communicates with the native driver and the device, while guest VM kernels host a front-end driver, exposing a generic device API to guest user– or kernel–space. The protocol's basic operations are assigned to each half, depending on their function while maintaining the architecture of the network and the protocol's main advantage such as the direct data transfer from VM user–space to the hardware and vice–versa, without the intervention of the VMM (zero–copy).

Using a simple micro-benchmark the performance of the Virtualized protocol was partialy evaluated. Preliminary results show that the Virtualized SLURPoE has a 12μs constant overhead latency-wise, but can achieve near-native performance when exchanging large messages (> 512KB).

Keywords

Virtual Machine, High performance network interconnect, User level communication, Virtual machine monitor, Split driver, Xen, SLURPoE, HPC

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου υπό την επίβλεψη του Αναπληρωτή Καθηγητή Νεκτάριου Κοζύρη.

Θα ήθελα να ευχαριστήσω θερμά τον κύριο Κοζύρη που μου έδωσε την ευκαιρία να έρθω σε επαφή με τον επιστημονικό τομέα των υπολογιστικών συστημάτων και που σε όλη τη διάρκεια των σπουδών μου στάθηκε σαν "φάρος" βοηθώντας με να ανακαλύψω αυτό που πραγματικά με ενδιέφερε.

Ευχαριστώ ολόψυχα τον υποψήφιο διδάκτορα Αναστάσιο Νάνο, του οποίου ήταν έμπνευση η παρούσα διπλωματική. Τον ευχαριστώ για την εμπιστοσύνη που μου έδειξε, για το αντικείμενο διπλωματικής που εμπνεύστηκε, μέσω της οποίας ικανοποιήθηκε η επιθυμία μου να φτιάξω "κάτι που να παίζει", για τις γνώσεις που απλόχερα μου χάρισε, για την υπομονή του και για τον χρόνο που αφιέρωσε προκειμένου να ολοκληρωθεί η εργασία.

Ευχαριστώ, επίσης, τους καθηγητές μου που με όπλισαν με όλες τις απαραίτητες γνώσεις έτσι ώστε να ανταποκριθώ στις απαιτήσεις της διπλωματικής εργασίας. Τους ευχαριστώ γιατί μου φανέρωσαν ένα πρότυπο ανθρώπου που θα ήθελα πολύ να ακολουθήσω.

Ευχαριστώ ολόψυχα την οικογένειά μου και το φιλικό μου περιβάλλον που όλα αυτά τα χρόνια δείχνουν κατανόηση, στηρίζουν τις επιλογές μου και με βοηθούν να παραμείνω μια ισορροπημένη και ολοκληρωμένη προσωπικότητα. Ευχαριστώ τους γονείς μου που δημιούργησαν ένα ιδανικό περιβάλλον έτσι ώστε να αφοσιωθώ στη σχολή μου και στη συνέχεια στη διπλωματική μου, την αδελφή μου που πάντα μου υπενθυμίζει τη χαρά της ζωής και τον θείο μου Ιωάννη Κοζύρη που με έμαθε να αναζητώ την πραγματική γνώση. Τέλος, ευχαριστώ των Θεόδωρο Γκούντουβα για την ηθική υποστήριξη σε σημεία που πίστευα ότι η διπλωματική αυτή ήταν πέρα από τις δυνατότητές μου.

Ελισάβετ Γ. Κοζύρη
Αθήνα, 6η Ιουλίου 2010

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Προσαρμογή των διεπαφών δικτύου υψηλής απόδοσης σε εικονικό περιβάλλον

Ο παραλληλισμός έχει ήδη γίνει τρόπος σκέψης στους προγραμματιστές, αφού δίνει λύση στην εκτέλεση χρονοβόρων και πολύπλοκων προγραμμάτων. Ο παράλληλος προγραμματισμός, όμως, απαιτεί τις κατάλληλες υποδομές έτσι ώστε να αποδειχτεί αποδοτικότερος του σειριακού. Οι δύο πιο δημοφιλείς επιλογές υλισμικού που υποστηρίζουν τα παράλληλα προγράμματα είναι "η συστοιχία υπολογιστών" (Cloud computing) και οι υπερυπολογιστές (Supercomputing). Το πρώτο βασίζεται σε ολοκληρωμένους υπολογιστές, με επεξεργαστή, αποθηκευτικό χώρο, παροχή ενέργειας και διεπαφή δικτύου, που είναι συνδεδεμένοι μέσω ενός συμβατού δικτύου, όπως το Ethernet. Σε αντίθεση με αυτό, ένα υπερυπολογιστής αποτελείται από πολλούς επεξεργαστές που είναι συνδεδεμένοι μεταξύ τους μέσω ενός τοπικού, υψηλής ταχύτητας διαύλου (computer bus). Επειδή, μέχρι στιγμής, το κόστος κατασκευής και αγοράς ενός υπερυπολογιστή είναι πολύ υψηλό, σε σχέση με τους συμβατούς υπολογιστές, το ενδιαφέρον των προγραμματιστών έχει στραφεί στη "συστοιχία υπολογιστών".

Πολλές εφαρμογές που εκτελούνται σε συστοιχίες υπολογιστών απαιτούν ανταλλαγή δεδομένων μεταξύ των παράλληλων νημάτων, κάτι που παίζει σημαντικό ρόλο στον χρόνο εκτέλεσής τους. Η προσπάθεια για βελτιστοποίηση της διαδικασίας ανταλλαγής δεδομένων προώθησε τη δημιουργία και την εξέλιξη των δικτύων διασύνδεσης υψηλής επίδοσης, τα οποία συνδέουν τους υπολογιστές της συστοιχίας. Τα δίκτυα αυτά βασίζονται στο γεγονός ότι τα περισσότερα στρώματα του IP/TCP είναι περιττά γι' αυτή την κατηγορία επικοινωνίας, καθώς έχουν μικρότερη κλίμακα και πολυπλοκότητα από το διαδίκτυο. Χρησιμοποιούν, λοιπόν, απλούστερα πρωτόκολλα και τεχνικές άμεσης

πρόσβασης στο υλισμικό έτσι ώστε να μειώσουν τις καθυστερήσεις της μεταφοράς δεδομένων.

Μια συστοιχία υπολογιστών αποτελείται πάντα από πραγματικούς υπολογιστές; Η απάντηση είναι όχι, αφού τα τελευταία χρόνια υπάρχει έντονο ενδιαφέρον για δημιουργία συστοιχίας που αποτελείται από εικονικές μηχανές, δηλαδή εικονικούς υπολογιστές. Αυτό σημαίνει ότι ένα μεγάλο κέντρο δεδομένων μπορεί να δημιουργεί και να εκτελεί ταυτόχρονα εικονικές μηχανές, οι οποίες είτε θα εκτελούν μεμονωμένες εφαρμογές είτε θα συμμετέχουν στην εκτέλεση πολυνηματικών εφαρμογών. Στη δεύτερη περίπτωση προκύπτουν ζητήματα επίδοσης της μεταξύ τους επικοινωνίας, κάτι που προέρχεται από τις συστοιχίες πραγματικών υπολογιστών. Σε αυτή τη περίπτωση όμως θα πρέπει να ξεπεραστεί και η καθυστέρηση που προσθέτουν τα επίπεδα εικονικοποίησης. Συνεπώς, γίνεται προσπάθεια να ενταχθούν τα δίκτυα διασύνδεσης υψηλής επίδοσης στα εικονικά περιβάλλοντα, έτσι ώστε να βελτιστοποιηθεί η επικοινωνία των εικονικών μηχανών. Η προσπάθεια αυτή είναι αντικείμενο της παρούσας διπλωματικής.

1.2 Αντικείμενο της διπλωματικής

Αντικείμενο της διπλωματικής εργασίας είναι η σχεδίαση και η υλοποίηση της εισαγωγής ενός δικτύου διασύνδεσης υψηλής επίδοσης σε εικονικό περιβάλλον. Το εικονικό περιβάλλον δημιουργείται από τον ελεγκτή εικονικής μηχανής Xen και το δίκτυο διασύνδεσης υψηλής επίδοσης που αναπτύχθηκε από το Εργαστήριο Υπολογιστικών Συστημάτων κατά τη διάρκεια της διπλωματικής με τίτλο "Σχεδίαση και Υλοποίηση μηχανισμού απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10GbE". Το πρωτόκολλο που χρησιμοποιεί το δίκτυο διασύνδεσης ονομάζεται SLURPoE και αναπτύχθηκε κατά τη διάρκεια της ίδιας διπλωματικής.

Προκειμένου να επιτευχθεί η προσαρμογή, υλοποιείται ένας νέος διαχωρισμένος οδηγός (split driver), του οποίου το κάτω μισό (back-end) τοποθετείται στη πρωτεύουσα εικονική μηχανή και το πάνω μισό (front-end) τοποθετείται σε μια φιλοξενούμενη εικονική μηχανή. Οι λειτουργίες του πρωτοκόλλου λοιπόν διαχωρίζονται στα δύο μισά του οδηγού, αναθέτοντας τη διεπαφή με το χώρο χρήστη στο πάνω μισό και τη διεπαφή με τη κάρτα δικτύου στο κάτω μισό. Στόχος της διπλωματικής είναι να χρησιμοποιηθούν οι μηχανισμοί επικοινωνίας που διαθέτει το Xen αλλά και τα εργαλεία που παρέχει το Linux για προσπέλαση του υλισμικού, έτσι ώστε να παραμείνουν αμετάβλητες οι διεπαφές προς το χρήστη και τη κάρτα.

Παράλληλα με το σχεδιασμό και την υλοποίηση, αντικείμενο της διπλωματικής είναι και η σύγκριση της επίδοσης του πρωτοκόλλου SLURPoE εκτός

εικονικού περιβάλλοντος με το πρωτόκολλο SLURPoE εντός εικονικού περιβάλλοντος. Με αυτό τον τρόπο εντοπίζονται και σχολιάζονται οι τυχόν αποκλίσεις, οι οποίες οφείλονται στα επιπλέον επίπεδα εικονικοποίησης που εισάγονται.

1.3 Οργάνωση του κειμένου

Το κείμενο οργανώθηκε έτσι ώστε ο αναγνώστης να έρχεται σε επαφή με τον πυρήνα της διπλωματικής σταδιακά. Δηλαδή, γίνεται προσπάθεια να αποκτήσει αρχικά μια γενική γνώση για τα εργαλεία και τις έννοιες που χρησιμοποιούνται, έπειτα να αποκτήσει μια διαίσθηση για το σύστημα που σχεδιάστηκε και τέλος να κατανοήσει τις λεπτομέρειες από τις οποίες αποτελείται η υλοποίηση του συστήματος. Πιο συγκεκριμένα, στο Κεφάλαιο 2 θα δοθεί το θεωρητικό υπόβαθρο που χρειάζεται ο αναγνώστης για να κατανοήσει το υπόλοιπο του κειμένου. Στο Κεφάλαιο 3 παρουσιάζεται ο σχεδιασμός του συστήματος και στο Κεφάλαιο 4 ο τρόπος με τον οποίο αυτό υλοποιήθηκε. Τέλος, στο Κεφάλαιο 5 παρουσιάζονται μετρήσεις που λήφθηκαν προκειμένου να συγκριθεί το πρωτόκολλο SLURPoE στα δύο περιβάλλοντα και στο Κεφάλαιο 6 αναφέρονται τα γενικά συμπεράσματα της διπλωματικής, καθώς και οι μελλοντικές επεκτάσεις της.

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο θα παρατεθούν και θα εξηγηθούν έννοιες στις οποίες βασίστηκε η παρούσα διπλωματική. Οι παρακάτω έννοιες ανήκουν κυρίως σε τρεις κατηγορίες: στις εικονικές μηχανές, στο λειτουργικό σύστημα Linux και στην επικοινωνία σε επίπεδο χρήστη. Θα γίνει προσπάθεια να περιγραφεί το ευρύτερο επιστημονικό πεδίο στο οποίο ανήκουν οι έννοιες αλλά και ο τρόπος με τον οποίο συνδέονται με τη διπλωματική.

2.1 Εικονική μηχανή

Ο όρος "εικονικός" έχει χρησιμοποιηθεί σε πολλά πεδία της επιστήμης και της κοινωνίας και το κάθε ένα από αυτά του έχει προσδώσει μια διαφορετική σημασιολογία. Σύμφωνα με τη φιλοσοφία, "εικονικός" είναι ο "μη πραγματικός". Άλλα πεδία που τον έχουν χρησιμοποιήσει είναι η εκπαίδευση, η οικονομία, η ιατρική, τα μαθηματικά και οι τηλεπικοινωνίες[1]. Το πεδίο όμως στο οποίο συναντάμε τις περισσότερες χρήσεις του είναι η επιστήμη των υπολογιστών. Αυτό μπορεί να εξηγηθεί από το ότι το λογισμικό δίνει τη δυνατότητα στον προγραμματιστή να δημιουργεί πρωτότυπες νοητές μηχανές και να προσομοιώσει είτε λειτουργίες που παραδοσιακά προσέφερε το υλισμικό, είτε φαινόμενα που λαμβάνουν χώρα στην πραγματική ζωή. Μερικά παραδείγματα είναι οι εφαρμογές "εικονικής πραγματικότητας", η "εικονική μνήμη" και η "εικονική επιφάνεια εργασίας". Ένα ακόμα παράδειγμα που περικλείει τις προαναφερθείσες προγραμματιστικές δυνατότητες και που θα αναλυθεί παρακάτω σε περισσότερο βάθος είναι οι "εικονικές μηχανές".

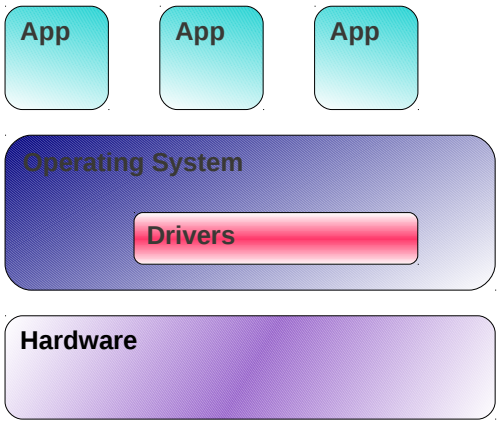
Εικονική μηχανή είναι η υλοποίηση μιας μηχανής σε λογισμικό, που εκτελεί προγράμματα όπως μια φυσική μηχανή[2]. Ο όρος της εικονικής μηχανής ορί-

στηκε αρχικά από τους Popek και Goldberg ως εξής: “Εικονική μηχανή θεωρείται ένα αποδοτικό και απομονωμένο αντίγραφο μιας πραγματικής μηχανής”[3]. Ωστόσο η έννοια αυτή σήμερα περιλαμβάνει υλοποιήσεις που δεν αποτελούν αντίγραφα πραγματικών μηχανών. Πάνω σε αυτό το διαχωρισμό βασίζεται η κατηγοριοποίηση των εικονικών μηχανών σε εικονικές μηχανές συστήματος και εικονικές μηχανές διεργασίας.

Μια εικονική μηχανή διεργασίας εκτελείται σαν μια συνηθισμένη εφαρμογή, μέσα σε ένα λειτουργικό σύστημα και υποστηρίζει μόνο ένα πρόγραμμα. Υλοποιείται με τη χρήση ενός διερμηνέα και σκοπός της είναι να παρέχει ένα προγραμματιστικό περιβάλλον ανεξάρτητο του λειτουργικού συστήματος και του υλισμικού. Χαρακτηριστικό παράδειγμα αποτελεί το Java virtual machine, το οποίο διερμηνεύει τον μεταγλωττισμένο κώδικα της Java για έναν επεξεργαστή, έτσι ώστε να μπορούν να εκτελεστούν οι εντολές κάθε προγράμματος Java. Με αυτό τον τρόπο οι προγραμματιστές μπορούν να συντάσσουν προγράμματα Java σε οποιοδήποτε υπολογιστή, αφού σε αυτόν θα υπάρχει η κατάλληλη εικονική μηχανή που θα διερμηνεύει τον τελικό κώδικα σε κώδικα που υποστηρίζει ο αντίστοιχος επεξεργαστής.

Από την άλλη πλευρά, μια εικονική μηχανή συστήματος είναι σαν ”μηχανή μέσα σε μια άλλη μηχανή” και λειτουργεί σαν να της ανήκουν όλοι οι πόροι του υπολογιστή, αν και την ίδια στιγμή μπορεί να συνυπάρχει με μία ή περισσότερες εικονικές μηχανές συστήματος. Πιο συγκεκριμένα, η εικονική μηχανή συστήματος, που στη συνέχεια του κειμένου θα αποκαλείται απλώς ”εικονική μηχανή” (virtual machine - VM), αποτελεί μία προσομοίωση ενός πραγματικού υπολογιστικού συστήματος κατά τη δημιουργία της οποίας ο χρήστης μπορεί να ορίσει γνωστές παραμέτρους συστήματος, όπως τον αριθμό των (εικονικών) πυρήνων, το μέγεθος της μνήμης, το μέγεθος της cache, τον αριθμό mac της κάρτας δικτύου καθώς και τη διεύθυνση IP των διεπαφών της. Επίσης όπως κάθε υπολογιστής υποστηρίζει ένα λειτουργικό σύστημα καθώς και την εκτέλεση οποιαδήποτε εφαρμογής του χρήστη. Για παράδειγμα, σε ένα πραγματικό σύστημα που περιέχει δύο πυρήνες και 4GB μνήμη μπορούν να δημιουργηθούν δύο εικονικές μηχανές με δύο (εικονικούς) πυρήνες και 4GB μνήμη η κάθε μία, που η μία να υποστηρίζει το λειτουργικό Linux, η άλλη λειτουργικό Windows και να λειτουργούν ταυτόχρονα χωρίς να παρεμβαίνουν οι εκτελέσεις των εφαρμογών της μίας σε αυτές της άλλης. Αυτό που περιγράφτηκε διαφέρει από τη δυνατότητα επιλογής λειτουργικού συστήματος κατά την έναρξη του υπολογιστή (dual-boot/multiboot environment) καθώς ο χρήστης πρέπει να επιλέγει μόνο ένα λειτουργικό σύστημα να εκτελείται κάθε φορά.

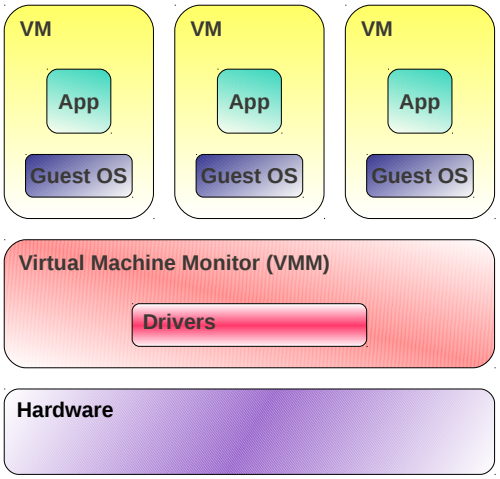
Όμως, πώς επιτυγχάνεται αυτή η ταυτόχρονη συνύπαρξη των εικονικών μηχανών; Η απάντηση ονομάζεται ”ελεγκτής εικονικής μηχανής” (virtual machine monitor - VMM). Ο ελεγκτής εικονικής μηχανής είναι ένα λογισμικό που εικονικοποιεί το υλισμικό για κάθε εικονική μηχανή μεσολαβώντας στην επικοινωνία των λειτουργικών συστημάτων των εικονικών μηχανών, που στη συνέχεια θα αποκαλούνται φιλοξενούμενα λειτουργικά συστήματα (guest operating systems), με το υλισμικό. Μερικές από τις λειτουργίες του είναι η διασφάλιση του απομονωμένου περιβάλλοντος εκτέλεσης των εικονικών μηχανών, η πολυπλεξία των αιτημάτων για πρόσβαση στους πόρους του συστήματος από τα φιλοξενούμενα λειτουργικά συστήματα και η αντιστοίχηση των διακοπών υλισμικού σε διακοπές λογισμικού έτσι ώστε να ειδοποιείται το κατάλληλο φιλοξενούμενο λειτουργικό σύστημα για διάφορα συμβάντα του υλισμικού. Για την απόκτηση μιας αρχικής διαίσθησης, ακολουθούν δύο σχηματικά διαγράμματα:



Σχήμα 2.1: Συμβατικός υπολογιστής

Στο Σχήμα 2.1, αναπαριστάται η ιεραρχική τοποθέτηση των τμημάτων του λογισμικού σε σχέση με το υλισμικό ενός υπολογιστή που δεν περιέχει εικονικό περιβάλλον. Σε αυτή τη περίπτωση το λειτουργικό σύστημα συνδέεται άμεσα με το υλισμικό, καθώς περιέχει τους οδηγούς του και επικοινωνεί με αυτό μέσω διακοπών. Επίσης, έχει τον πλήρη έλεγχο των πόρων του υπολογιστή, όπως τη μνήμη, οργανώνει την εκτέλεση των εφαρμογών που δημιουργούνται από τον χρήστη και δρα ως διεπαφή μεταξύ των εφαρμογών και του υλισμικού. Με άλλα λόγια, το λειτουργικό σύστημα κατέχει τον πλήρη έλεγχο του συστήματος και βρίσκεται, όπως λέγεται, στο δακτυλίδι 0 (ring 0).

Από την άλλη πλευρά, στο Σχήμα 2.2, παρουσιάζεται η παραπάνω ιεραρχία με τη διαφορά ότι τώρα υπάρχει εικονικό περιβάλλον στον υπολογιστή. Σε αυτή

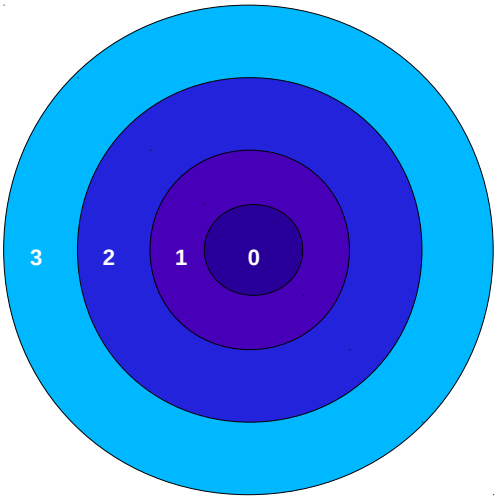


Σχήμα 2.2: Υπολογιστής με εικονικό περιβάλλον

τη περίπτωση ο ελεγκτής εικονικής μηχανής κατέχει τον πλήρη έλεγχο του συστήματος. Αυτό σημαίνει ότι εκτελεί τόσες λειτουργίες όσες εκτελούσε το λειτουργικό σύστημα στο Σχήμα 1 και επιπλέον οργανώνει την εκτέλεση των εικονικών μηχανών που δημιουργούνται. Μέρος της οργάνωσης αυτής είναι ο τρόπος με τον οποίο οι εικονικές μηχανές θα επικοινωνούν με το υλισμικό αλλά και ο τρόπος με τον οποίο αυτές θα μοιράζονται τους πόρους του υπολογιστικού συστήματος. Από τη πλευρά τους, τα φιλοξενούμενα λειτουργικά συστήματα (guest operating systems) οργανώνουν την εκτέλεση των εφαρμογών τους και επικοινωνούν με τον ελεγκτή εικονικής μηχανής σε ζητήματα πρόσβασης στο πραγματικό υλισμικό[4].

Σε αυτό το σημείο θα εξηγηθούν τα επίπεδα δικαιωμάτων (CPU privilege levels ή rings) που παρέχει ο επεξεργαστής x86, καθώς θα υπάρχουν συχνές αναφορές σε αυτά στη συνέχεια του κειμένου. Τα πιο μοντέρνα λειτουργικά συστήματα δεν επιτρέπουν στις εφαρμογές να εκτελούν συγκεκριμένες λειτουργίες. Για παράδειγμα, μόνο το λειτουργικό σύστημα μπορεί να φορτώσει τους οδηγούς ή να προσπελάσει απευθείας το υλισμικό. Για να περιοριστούν όλες οι εφαρμογές υπό εκτέλεση μόνο σε ένα υποσύνολο των πόρων, το λειτουργικό σύστημα και ο επεξεργαστής συνεργάζονται χρησιμοποιώντας τα επίπεδα δικαιωμάτων (Σχήμα 2.3).

Σε κάθε στιγμή ο επεξεργαστής x86 λειτουργεί σε ένα από αυτά τα επίπεδα. Το επίπεδο (δακτυλίδι) 0 έχει τα περισσότερα δικαιώματα και το επίπεδο 3 τα λιγότερα. Οι πόροι που προστατεύονται μέσω των επιπέδων είναι: η μνήμη, οι θύρες εισόδου/εξόδου και οι εντολές του επεξεργαστή. Το λειτουργικό σύστημα εκτελείται στο επίπεδο 0 (κατάσταση πυρήνα - kernel mode), οι εφαρμογές του χρήστη στο επίπεδο 3 (κατάσταση χρήστη - user mode) και τα



Σχήμα 2.3: CPU privilege levels

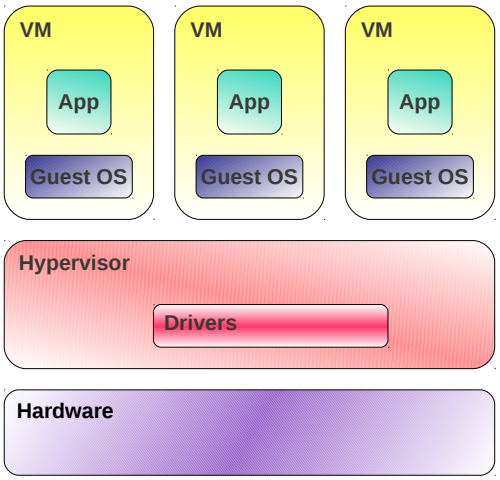
επίπεδα 1 και 2 παραμένουν αχρησιμοποίητα. Όμως, σε έναν υπολογιστή με εικονικό περιβάλλον, ο ελεγκτής εικονικής μηχανής θα πρέπει να μπορεί να προσπελαύνει την μνήμη, τον επεξεργαστή και τις συσκευές εισόδου/εξόδου. Αυτό σημαίνει ότι θα πρέπει να εκτελείται στο επίπεδο με τα περισσότερα δικαιώματα (επίπεδο 0). Από την άλλη πλευρά, τα φιλοξενούμενα λειτουργικά συστήματα περιμένουν να έχουν κι αυτά πρόσβαση σε όλους τους πόρους. Επειδή, όμως, μόνο ένας πυρήνας μπορεί να βρίσκεται στο επίπεδο 0, τα φιλοξενούμενα λειτουργικά συστήματα θα πρέπει να εκτελούνται είτε σε άλλο επίπεδο, με λιγότερα δικαιώματα, (επίπεδο 1) είτε θα πρέπει να τροποποιηθούν για να εκτελούνται στο επίπεδο 3.

Ο τρόπος με τον οποίο δημιουργείται ένα εικονικό περιβάλλον δεν είναι μοναδικός και εξαρτάται κυρίως από τον σχεδιασμό του ελεγκτή εικονικής μηχανής. Ο δημιουργός ενός εικονικού περιβάλλοντος καλείται να απαντήσει σε δύο σχεδιαστικά ερωτήματα, που αφορούν τον ελεγκτή εικονικής μηχανής, προκειμένου το τελικό προϊόν να ικανοποιεί τις απαιτήσεις του. Οι ερωτήσεις είναι οι εξής:

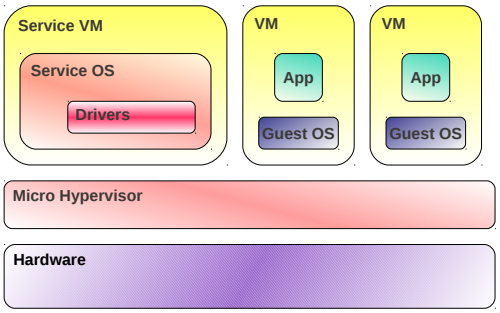
- Ο ελεγκτής εικονικής μηχανής θα είναι σε άμεση επαφή με το υλισμικό ή θα παρεμβάλλεται ένα αρχικό λειτουργικό σύστημα;
- Ο ελεγκτής εικονικής μηχανής θα προσομοιώνει πλήρως το υλισμικό προς τη πλευρά των εικονικών μηχανών ή όχι;

Ανάλογα με την απάντηση στη πρώτη ερώτηση, δημιουργούνται δύο κατηγορίες ελεγκτών εικονικής μηχανής: ο τύπος Ι και ο τύπος ΙΙ. Στον τύπο Ι ανήκουν οι ελεγκτές εικονικής μηχανής που τουλάχιστον ένα μέρος τους βρίσκεται σε άμεση επαφή με το υλισμικό, δηλαδή εκτελείται απευθείας πάνω

στο υλισμικό. Σε αυτόν τον τύπο υπάρχουν δύο υποτύποι ελεγκτών εικονικής μηχανής: ο επιβλέπων (Hypervisor), που εκτελείται εξολοκλήρου πάνω στο υλισμικό (Σχήμα 2.4) και το λειτουργικό σύστημα υπηρεσίας (Service OS), που ένα μικρό μέρος εκτελείται πάνω στο υλισμικό ενώ το υπόλοιπο στη πρωτεύουσα εικονική μηχανή (Σχήμα 2.5).



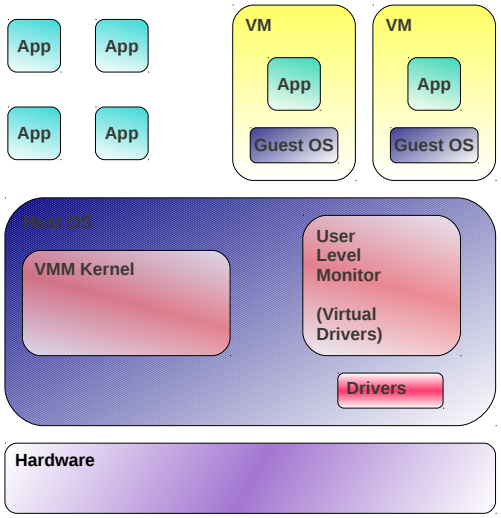
Σχήμα 2.4: Hypervisor



Σχήμα 2.5: Service OS

Στον τύπο II ανήκουν οι ελεγκτές εικονικής μηχανής που εκτελούνται πάνω από ένα ήδη εγκατεστημένο λειτουργικό σύστημα (Host OS). Ο ελεγκτής και το υπάρχον λειτουργικό σύστημα βρίσκονται στο δακτυλίδι 0 (ring 0). Μία αίτηση για πρόσβαση των εικονικών μηχανών στις συσκευές εισόδου/εξόδου, ανακατευθύνεται μέσω του ελεγκτή (VMM) στους εικονικούς οδηγούς, που βρίσκονται στον ελεγκτή επιπέδου χρήστη (User Level Monitor) και εκείνοι με τη σειρά τους την κατευθύνουν προς τους πραγματικούς οδηγούς του λειτουργικού συστήματος. Στο Σχήμα 2.6 παρουσιάζεται η δομή του.

Όσο απομακρύνεται το λογισμικό του ελεγκτή από το υλισμικό, τόσο μικραίνει η εξάρτησή του από αυτό και τόσο μεγαλώνει η φορητότητά του. Αυτό



Σχήμα 2.6: Host OS

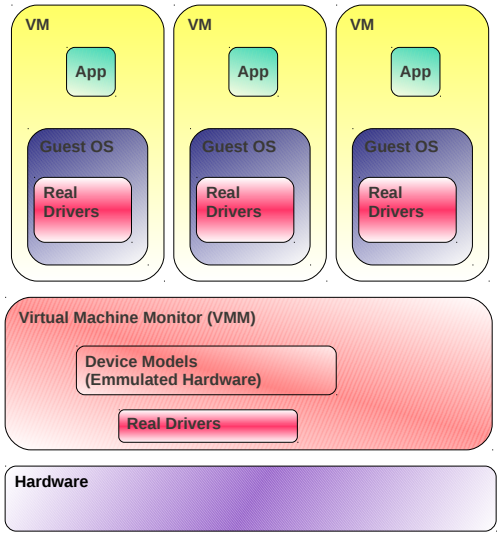
οφείλεται στο ότι οι οδηγοί μεταβιβάζουν μέρος των λειτουργιών τους στον ελεγκτή των εικονικών μηχανών και τείνουν να αποκτήσουν τον χαρακτήρα των οδηγών ενός απλού λειτουργικού συστήματος. Πιο συγκεκριμένα, οι οδηγοί ενός ελεγκτή τύπου ”επιβλέπων” θα πρέπει να επικοινωνούν με επιτυχία με τις αντίστοιχες συσκευές, να οργανώνουν κατάλληλα τα αιτήματα από τις υπάρχουσες εικονικές μηχανές και να ανακατευθύνουν τις διακοπές και τα δεδομένα, που προέρχονται από τις συσκευές, στην κατάλληλη εικονική μηχανή. Σε αυτή τη περίπτωση, οι οδηγοί έχουν δημιουργηθεί ειδικά για το συγκεκριμένο υλισμικό και έτσι η φορητότητά τους σε άλλο υλισμικό θα είναι περιορισμένη. Στον ελεγκτή τύπου ”λειτουργικό σύστημα υπηρεσίας”, οι οδηγοί έχουν μεταφερθεί στο λειτουργικό σύστημα υπηρεσίας, με το οποίο πρέπει να επικοινωνούν τα υπόλοιπα φιλοξενούμενα λειτουργικά συστήματα έτσι ώστε να αποκτήσουν πρόσβαση στο υλισμικό. Το λειτουργικό σύστημα υπηρεσίας, όμως είναι ένα κοινό, ελαφρώς τροποποιημένο λειτουργικό σύστημα. Αυτό σημαίνει ότι ως προς την επικοινωνία με το υλισμικό, οι οδηγοί θα είναι πανομοιότυποι με αυτούς που υπάρχουν στα δημοφιλή λειτουργικά συστήματα, ενώ ως προς την επικοινωνία με τις υπόλοιπες εικονικές μηχανές θα υπάρχουν τροποποιήσεις. Το γεγονός αυτό, καθιστά τον τύπο αυτό πιο ευέλικτο, αφού βασίζεται σε οδηγούς ευρείας χρήσης. Τέλος, στον ελεγκτή τύπου II οι οδηγοί βρίσκονται στο ήδη υπάρχον λειτουργικό σύστημα. Αυτό σημαίνει ότι είναι ανεξάρτητοι του εικονικού περιβάλλοντος και ταυτίζονται πλήρως με τους συμβατικούς οδηγούς των κοινών λειτουργικών συστημάτων. Οι λειτουργίες της οργάνωσης των αιτημάτων και της ανακατεύθυνσης των δεδομένων και των διακοπών έχουν πλήρως μεταβιβαστεί στον ελεγκτή. Συνεπώς, όσο απομακρύνεται το λογισμικό του ελεγκτή από το υλισμικό, τόσο

αυξάνεται η ευελιξία.

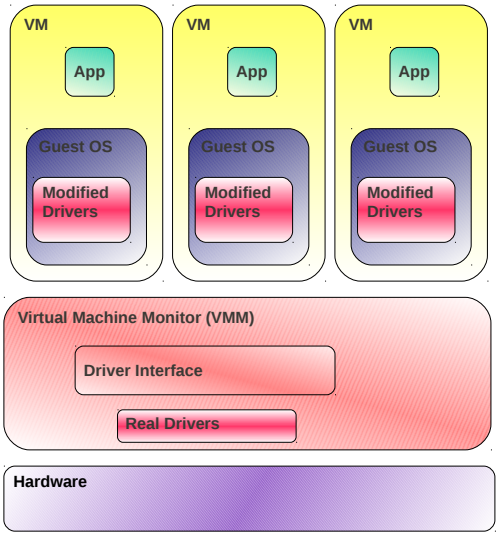
Όμως, ταυτόχρονα αυξάνεται η πολυπλοκότητα του συστήματος και μειώνεται η απόδοσή του. Αυτό συμβαίνει διότι προσθέτονται επιπλέον επίπεδα ανάμεσα σε μία εικονική μηχανή και στο πραγματικό υλισμικό. Στον τύπο "επιβλέπον", η εικονική μηχανή συνομιλεί απευθείας με τον ελεγκτή και τους οδηγούς. Στον τύπο "λειτουργικό σύστημα υπηρεσίας", η εικονική μηχανή συνομιλεί πρώτα με την την εικονική μηχανή υπηρεσίας και έπειτα αυτή επικοινωνεί με το υλισμικό. Τέλος, στον τύπο II, η εικονική μηχανή επικοινωνεί με τον ελεγκτή, ο ελεγκτής με τον ελεγκτή επιπέδου χρήστη κι εκείνος με το υλισμικό.

Λαμβάνοντας υπόψη όλα τα παραπάνω, γίνεται αντιληπτό ότι ο σχεδιαστής θα πρέπει να επιλέξει έναν από τους τρεις τύπους ανάλογα με τις απαιτήσεις του σε ευελιξία και σε απόδοση.

Ανάλογα με την απάντηση στη δεύτερη ερώτηση (ο ελεγκτής εικονικής μηχανής θα προσομοιώνει πλήρως το υλισμικό προς τη πλευρά των εικονικών μηχανών ή όχι;), δημιουργούνται δύο κατηγορίες τεχνικών εικονικοποίησης: η πλήρης εικονικοποίηση (full virtualization) και η μερική εικονικοποίηση (paravirtualization). Οι τεχνικές της πρώτης κατηγορίας (Σχήμα 2.7), απαιτούν ελεγκτές που προσομοιώνουν πλήρως το υλισμικό προς τη πλευρά των εικονικών μηχανών, ενώ οι τεχνικές της δεύτερης κατηγορίας (Σχήμα 2.8) απαιτούν ελεγκτές που παρουσιάζουν εικονικούς οδηγούς στη πλευρά των εικονικών μηχανών και οργανώνουν την συνομιλία τους με τους πραγματικούς.



Σχήμα 2.7: Full virtualization



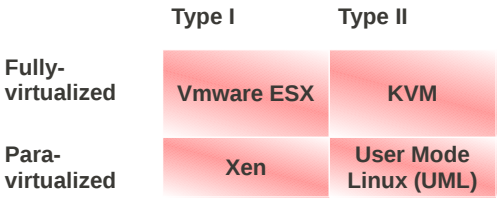
Σχήμα 2.8: Paravirtualization

Στην τεχνική της πλήρους εικονικοποίησης, όπως αναφέρθηκε παραπάνω, ο ελεγκτής προσομοιώνει πλήρως το υλισμικό. Αυτό σημαίνει ότι χρησιμοποιεί το λογισμικό για να προσομοιώσει τις λειτουργίες του υλισμικού, έτσι ώστε να δίνεται στα φιλοξενούμενα λειτουργικά συστήματα η ψευδαίσθηση ότι επικοινωνούν απευθείας με τις συσκευές, αν και στη πραγματικότητα παρεμβάλλεται ο ελεγκτής. Σε αυτή τη περίπτωση, λοιπόν, η λειτουργία του ελεγκτή είναι να ανακατευθύνει και να οργανώνει τις κλήσεις των οδηγών των φιλοξενούμενων λειτουργικών συστημάτων στις πραγματικές συσκευές. Σε αυτό το σημείο, γίνεται κατανοητό ότι η τεχνική αυτή επιτρέπει στις εικονικές μηχανές να έχουν μη τροποποιημένα λειτουργικά συστήματα και μη τροποποιημένους οδηγούς. Δηλαδή, θα μπορούσε να φιλοξενηθεί οποιοδήποτε κοινό λειτουργικό σύστημα, αφού θα έχει την εντύπωση ότι εκτελείται πάνω σε πραγματικό υλισμικό και όχι μέσα σε εικονικό περιβάλλον. Από την άλλη πλευρά, η τεχνική αυτή παρουσιάζει μειωμένη απόδοση, αφού η διαδικασία της προσομοίωσης του υλισμικού είναι χρονοβόρα και απαιτητική σε θέμα υλοποίησης.

Από την άλλη πλευρά, στην τεχνική μερικής εικονικοποίσης τα φιλοξενούμενα λειτουργικά συστήματα έχουν επίγνωση ότι εκτελούνται μέσα σε εικονικό περιβάλλον. Τα ίδια, αλλά και οι οδηγοί που περιέχουν, έχουν τροποποιηθεί σε σχέση με τα κοινά λειτουργικά συστήματα έτσι ώστε να επικοινωνούν με τις κατάλληλες διεπαφές των πραγματικών οδηγών, που βρίσκονται στον ελεγκτή. Με άλλα λόγια, η κλήση ενός οδηγού στην αντίστοιχη συσκευή αντικαθίσταται από μια κλήση στην αντίστοιχη διεπαφή του ελεγκτή και από μια κλήση της διεπαφής αυτής στη συσκευή. Συνεπώς, η τεχνική αυτή είναι πιο

αποδοτική, αφού ο ελεγκτής απλώς αποκρίνεται σε κλήσεις των φιλοξενούμενων εικονικών μηχανών, αλλά απαιτεί την τροποποίηση των λειτουργικών συστημάτων και των οδηγών, κάτι που είναι λιγότερο ευέλικτο και πολλές φορές χρονοβόρο. Η διαδικασία της μερικής εικονικοποίησης μπορεί να αποφευχθεί, αν ο επεξεργαστής του υπολογιστικού συστήματος υποστηρίζει το εικονικό περιβάλλον, όπως ο επεξεργαστής Intel VT και ο AMD-V [5].

Συνδυάζοντας τις τέσσερις παραπάνω κατηγοριοποιήσεις των ελεγκτών εικονικής μηχανής δημιουργούνται τέσσερα σύνολα ελεγκτών στα οποία ανήκουν σύγχρονα και δημοφιλή εικονικά περιβάλλοντα. Στο Σχήμα 2.9 παρουσιάζονται αντιπροσωπευτικά παραδείγματα των συνόλων αυτών.



Σχήμα 2.9: Ταξινόμηση των ελεγκτών εικονικής μηχανής

Ο ελεγκτής εικονικής μηχανής UML είναι γνωστός από το λειτουργικό σύστημα Linux, αφού αποτελεί εργαλείο δοκιμής λογισμικού που προορίζεται για τον πυρήνα του λειτουργικού, ενώ ο ελεγκτής VMware ESX είναι ένας από τους πιο δημοφιλείς, εμπορικούς, ελεγκτές. Οι άλλοι δύο ελεγκτές, Xen και Kvm, θα αναλυθούν στη συνέχεια με περισσότερη λεπτομέρεια.

Τα ερωτήματα που έμειναν αναπάντητα για το τέλος είναι:

- Ποια είναι η χρησιμότητα των εικονικών μηχανών;
- Ποια είναι τα πλεονεκτήματα των εικονικών μηχανών;

Ως προς το πρώτο ερώτημα, οι εικονικές μηχανές προήλθαν από την επιθυμία να συνυπάρχουν πολλά και διαφορετικά λειτουργικά συστήματα στο ίδιο υπολογιστικό σύστημα, μοιράζοντας τον χρόνο και τους πόρους του με τον καλύτερο τρόπο. Υπάρχει επίσης η τάση σε μεγάλα υπολογιστικά κέντρα (κέντρα στα οποία οι πελάτες ζητάνε πόρους για την εκτέλεση των εφαρμογών τους) να χρησιμοποιούνται συστοιχίες από εικονικές μηχανές αντί συστοιχίες από υπολογιστές, διότι με τις πρώτες επιτυγχάνεται μεγαλύτερη χρησιμοποίηση των πόρων και μεγαλύτερη ευελιξία στις απαιτήσεις των πελατών. Για παράδειγμα, στις συστοιχίες υπολογιστών υπάρχει πιθανότητα να μην χρησιμοποιούνται όλοι οι υπολογιστές, κάτι που δεν συμβαίνει στις συστοιχίες εικονικών μηχανών αφού ο ελεγκτής διαμοιράζει τους πόρους έτσι ώστε να

χρησιμοποιούνται όλοι. Επίσης, οι πελάτες έχουν την δυνατότητα να ζητούν εικονικές μηχανές με παραμέτρους που θα καθορίζουν αυτοί, χωρίς να περιορίζονται στις πραγματικές (μνήμη, αριθμός επεξεργαστών).

Ως προς το δεύτερο ερώτημα, μπορούν να αναφερθούν τα εξής πλεονεκτήματα:

- Συνύπαρξη πολλαπλών λειτουργικών συστημάτων.
- Σταθερότητα και Ασφάλεια, λόγω της απομονωμένης εκτέλεσης των εικονικών μηχανών.
- Ευελιξία στην ανάπτυξη λογισμικού, καθώς μπορεί να δοκιμάζονται καινούρια προγράμματα και λειτουργικά συστήματα.
- Μετανάστευση και Κλωνοποίηση, αφού οι εικονικές μηχανές μπορούν να αντιγράφονται και να μετακινούνται σε διαφορετικό εξυπηρετητή εύκολα και αποδοτικά.
- Εικονικοποίηση επιφάνειας εργασίας, σύμφωνα με την οποία οι εφαρμογές, τα δεδομένα και το λειτουργικό σύστημα ενός χρήστη βρίσκονται σε έναν απομακρυσμένο εξυπηρετητή και ο ίδιος αποκτά πρόσβαση σε αυτά μέσω ενός απλού προγράμματος "πελάτη", αφήνοντας τη διαχείριση του λογαριασμού του στο υπολογιστικό κέντρο.

Επειδή, όμως, τίποτα δεν έχει μόνο πλεονεκτήματα, θα πρέπει να αναφερθεί σε αυτό το σημείο το βασικό μειονέκτημα των εικονικών μηχανών, το οποίο είναι η καθυστέρηση επικοινωνίας τους με το υλισμικό. Αυτό οφείλεται στα στρώματα που παρεμβάλλονται μεταξύ των φιλοξενούμενων λειτουργικών συστημάτων και του υλισμικού, λόγω της εικονικοποίησης και κατ' επέκταση λόγω του ελεγκτή εικονικής μηχανής. Το μειονέκτημα αυτό προσπαθούν να ελαττώσουν οι σύγχρονες έρευνες, κάτι το οποίο θα είναι μέρος και της παρούσας διπλωματικής.

2.1.1 KVM

Το σύστημα KVM (Kernel-based Virtual Machine) είναι η πρώτη μέθοδος εικονικοποίησης που ενσωματώθηκε στον πυρήνα του λειτουργικού συστήματος Linux. Το KVM αρχικά αναπτύχθηκε από τη Qumranet, μια μικρή εταιρία στο Ισραήλ. Η εταιρία Redhat απέκτησε τη Qumranet το Σεπτέμβριο του 2008, όταν το KVM έγινε έτοιμο να εισέλθει στην παραγωγή, αφού έβλεπαν σε αυτό την επόμενη γενιά της τεχνολογίας της εικονικοποίησης. Σήμερα είναι

ο προκαθορισμένος ελεγκτής εικονικής μηχανής στο Redhat Enterprise Linux (RHEL)[6].

Η Qumranet διένειμε τον κώδικα του KVM στην κοινότητα του ανοιχτού λογισμικού. Γι' αυτόν τον λόγο επωφελείται από την παγκόσμια ομάδα ανάπτυξης λογισμικού για το λειτουργικό σύστημα Linux, αφού αν το Linux αποκτήσει καλύτερη απόδοση λόγω των καινούριων αλγορίθμων, θα αποδίδει και το KVM καλύτερα.

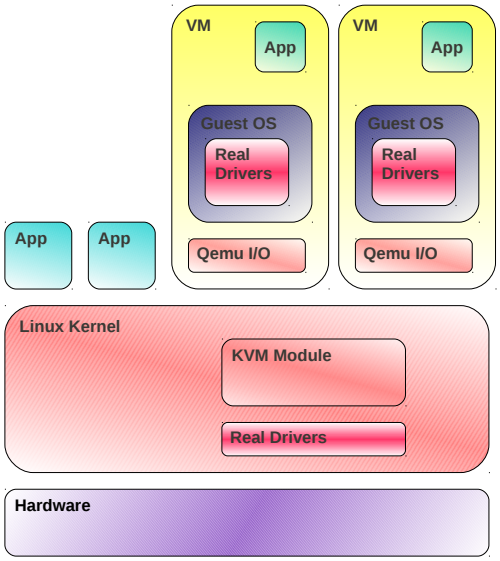
Το KVM, όπως παρουσιάστηκε παραπάνω, χρησιμοποιεί τη τεχνική της πλήρους εικονικοποίησης για την εκτέλεση των εικονικών μηχανών. Αποτελείται από έναν μικρό βασικό κώδικα, αφού έχει σχεδιαστεί να εκμεταλλεύεται πλήρως τις δυνατότητες που παρέχει το υλισμικό υποστήριξης της εικονικοποίησης, κάτι το οποίο θα εξηγηθεί στη συνέχεια. Το σύστημα αυτό εκτελείται σε αρχιτεκτονικές x86, αλλά πρόσφατα προστέθηκε υποστήριξη για τις αρχιτεκτονικές IA64 και IBM s390.

Αρχιτεκτονική

Στο σύστημα του KVM, τον ρόλο του ελεγκτή εικονικής μηχανής τον παίζει το ίδιο το λειτουργικό Linux, καθώς αυτό έχει πολλούς από τους μηχανισμούς που χρειάζεται για να υποστηριχθούν οι εικονικές μηχανές (χρονοδρομολόγηση, διαμοιρασμός πόρων, διασφάλιση απομονωμένης εκτέλεσης). Οι προγραμματιστές έπρεπε απλώς να προσθέσουν τις υπόλοιπες λειτουργίες που θα ολοκλήρωναν τη δημιουργία ενός εικονικού περιβάλλοντος. Αυτές οι επιπρόσθετες λειτουργίες αποτελούν το KVM, το οποίο προσαρτάται στον πυρήνα του λειτουργικού μέσω μιας λειτουργικής μονάδας (kernel mudule). Με την προσάρτηση αυτή, το λειτουργικό σύστημα Linux μετατρέπεται σε ελεγκτή εικονικής μηχανής, της οποίας η μορφή φαίνεται στο Σχήμα 2.10.

Όταν φορτώνεται η μονάδα του KVM, εμφανίζεται στο σύστημα των αρχείων (filesystem) η συσκευή /dev/kvm, που αναπαριστά τη διεπαφή του KVM. Επιτρέπει τον έλεγχο του εικονικού περιβάλλοντος μέσω ενός συνόλου κλήσεων ioctl. Αυτές οι κλήσεις χρησιμοποιούνται ευρύτατα στα λειτουργικά συστήματα για την επικοινωνία μεταξύ των διεργασιών που βρίσκονται σε κατάσταση χρήστη και των οδηγών του υλισμικού, που βρίσκονται σε κατάσταση πυρήνα. Μερικές από τις λειτουργίες των κλήσεων αυτών είναι η δημιουργία μιας καινούριας εικονικής μηχανής, ο καθορισμός μνήμης σε μια εικονική μνήμη και η εκκίνηση εικονικών επεξεργαστών.

Σε ένα συμβατό περιβάλλον Linux, κάθε διεργασία εκτελείται είτε σε κατάσταση χρήστη (user-mode) είτε σε κατάσταση πυρήνα (kernel-mode). Το σύστη-



Σχήμα 2.10: KVM

μα KVM εισάγει μια τρίτη κατάσταση εκτέλεσης, που ονομάζεται κατάσταση φιλοξενούμενου (guest-mode) και θα είναι αυτή στην οποία θα εκτελούνται τα φιλοξενούμενα λειτουργικά συστήματα.

Διαχείριση πόρων

Οι βασικοί πόροι που διαχειρίζεται ο ελεγκτής εικονικής μηχανής, έτσι ώστε να κατανέμονται δίκαια στις εικονικές μηχανές είναι: η μνήμη, ο επεξεργαστής (χρόνος εκτέλεσης) και οι συσκευές υλισμικού.

Λαμβάνοντας υπόψη ότι οι εικονικές μηχανές έχουν τις ίδιες ελάχιστες απαιτήσεις με αυτές ενός συμβατού υπολογιστή, θα πρέπει να τους ανατίθεται τόση μνήμη όση θα είχε μια φυσική μηχανή. Ο ελεγκτής εικονικής μηχανής (VMM) διαχωρίζει την εικονική μνήμη του συστήματος σε συνεχόμενα κομμάτια σταθερού μεγέθους και τα αντιστοιχεί στο χώρο διευθύνσεων που προορίζεται για κάθε εικονική μηχανή. Στην πραγματικότητα, τα κομμάτια αυτά βρίσκονται διασκορπισμένα στη φυσική μνήμη του συστήματος αλλά και στον σκληρό του δίσκο. Επειδή η εικονική μηχανή δεν έχει επίγνωση της φυσικής μνήμης του συστήματος, διότι θεωρεί ότι η μνήμη που της έχει ανατεθεί είναι η φυσική της μνήμη, ο ελεγκτής εικονικής μνήμης είναι υπεύθυνος για την αποθήκευση των διευθύνσεων που ανήκουν σε κάθε εικονική μηχανή αλλά και για τη επαναφόρτωση σελίδων, που χρειάζονται οι εικονικές μηχανές, από τον σκληρό δίσκο στη μνήμη. Για να επιτευχθεί αυτό, οι προγραμματιστές

εκμεταλλεύτηκαν τη διαχείριση μνήμης που υπάρχει ήδη στα Linux και αποθήκευσαν τις αντιστοιχίσεις εικονικών-φυσικών διευθύνσεων των εικονικών μηχανών σε ειδικές δομές που ονομάζονται shadow page tables.

Στα μοντέρνα λειτουργικά συστήματα εκτελούνται πολύ περισσότερες διεργασίες από τον αριθμό των επεξεργαστών που διατίθενται. Αυτό οφείλεται στον χρονοδρομολογητή που περιέχουν, ο οποίος καθορίζει τη σειρά με την οποία κάθε διεργασία θα ανατίθεται στους διαθέσιμους επεξεργαστές. Αφού οι προγραμματιστές του KVM επιθυμούσαν να εκμεταλλευτούν τους μηχανισμούς του Linux, υλοποίησαν κάθε κάθε εικονική μηχανή ως μία διεργασία και βασίστηκαν στον χρονοδρομολογητή για την ανάθεση υπολογιστικής ισχύς στις εικονικές μηχανές.

Τέλος, το σύστημα KVM για να παρέχει προσομοιωμένες συσκευές υλισμικού στις εικονικές μηχανές, όπως σκληρό δίσκο, οδηγούς CD και κάρτες δικτύου, χρησιμοποιεί ένα τροποποιημένο λογισμικό QEMU. Το λογισμικό αυτό είναι ένα εργαλείο εικονικοποίησης, το οποίο επιτρέπει την προσομοίωση ενός ολόκληρου υπολογιστικού συστήματος, συμπεριλαμβανομένου των γραφικών, του δικτύου και των δίσκων. Για κάθε εικονική μηχανή, μια διεργασία QEMU ξεκινάει σε κατάσταση χρήστη και παρέχει τις προσομοιωμένες συσκευές. Όταν μια εικονική μηχανή εκτελεί μια εντολή εισόδου/εξόδου, λαμβάνεται από το KVM και ανακατευθύνεται στην αντίστοιχη διεργασία QEMU.

Επίδοση

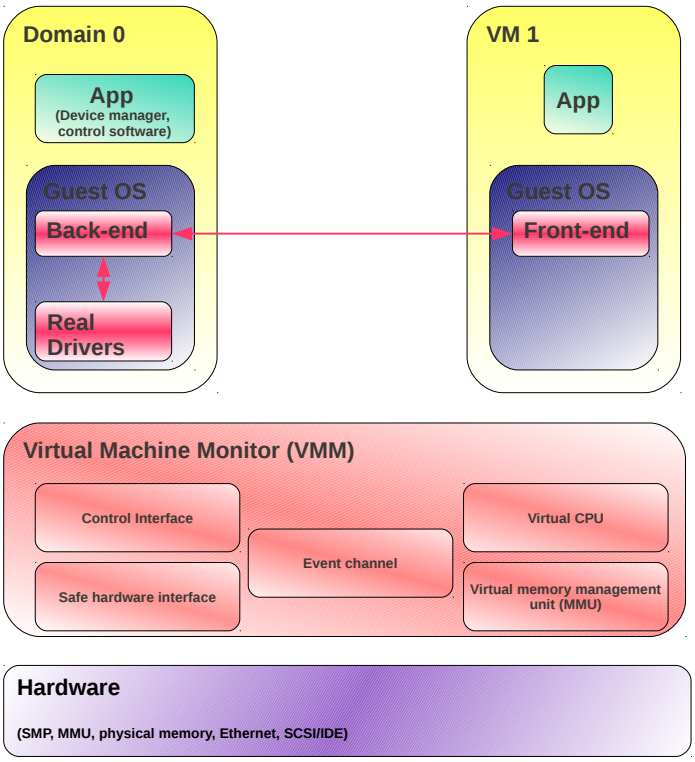
Ένα από τα βασικά πλεονεκτήματα του συστήματος KVM είναι ότι βασίζεται σε μεγάλο βαθμό από τον πυρήνα του Linux και έτσι επωφελείται άμεσα από τις βελτιστοποιήσεις που εφαρμόζονται σε αυτόν. Επίσης, εφαρμόζοντας την τεχνική της πλήρους εικονικοποίησης (full-virtualization), μπορεί να υποστηρίξει όλα τα λειτουργικά συστήματα, είτε είναι ανοιχτού κώδικα είτε είναι κλειστού, αφού δεν απαιτούνται αλλαγές σε αυτά. Επιπλέον, διευκολύνεται σε μεγάλο βαθμό η μετανάστευση των εικονικών μηχανών ανάμεσα σε συστήματα με διαφορετικό υλισμικό και παρέχει πλήρη απομόνωση των διάφορων εφαρμογών αυξάνοντας την ασφάλεια του συστήματος. Από την άλλη πλευρά, λόγω της προσομοίωσης του υλισμικού μειώνεται η ταχύτητα επικοινωνίας με αυτό και απομακρύνεται από τη φυσική επίδοση του συστήματος. Γι' αυτό το λόγο, έχουν ήδη βρεθεί λύσεις τύπου μερικής εικονικοποίησης που προσαρτώνται στο σύστημα KVM.

2.1.2 Xen

Το σύστημα Xen προήλθε από μία επιστημονική εργασία του πανεπιστημίου του Cambridge το 2003[7]. Είναι ένας ελεγκτής εικονικής μηχανής, για επεξεργαστή x86, που επιτρέπει σε πολλαπλά, συμβατικά λειτουργικά συστήματα να μοιράζονται ένα συμβατικό υλισμικό με ασφαλή και αποδοτικό τρόπο. Είναι λογισμικό ανοιχτού κώδικα και βασίζεται στη τεχνική της μερικής εικονικοποίησης.

Αρχιτεκτονική

Στο Σχήμα 2.11 παρουσιάζεται η αρχιτεκτονική του Xen.



Σχήμα 2.11: Xen

Όπως φαίνεται από το σχήμα, αλλά και όπως είχε αναφερθεί παραπάνω, το Xen είναι τύπου I και πιο συγκεκριμένα, ανήκει στον υποτύπο ”λειτουργικό σύστημα υπηρεσίας”. Το ρόλο της εικονικής μηχανής υπηρεσίας τον παίζει η εικονική μηχανή Domain 0 (Dom0), ενώ η εικονική μηχανή VM 1 είναι μια κοινή εικονική μηχανή (DomU), που επικοινωνεί με το Dom0 για τη προσπέλαση στο υλισμικό. Το Dom0 δημιουργείται κατά την εκκίνηση του συστήματος (boot time) και έχει δικαίωμα να χρησιμοποιεί τη διεπαφή ελέγχου

του συστήματος (control interface), με την οποία μπορεί ανάμεσα στα άλλα να δημιουργεί και να τερματίζει εικονικές μηχανές, να δημιουργεί εικονικές διεπαφές δικτύου και να διαχειρίζεται τη πρόσβαση των εικονικών μηχανών στο υλισμικό.

Όσον αφορά τα επίπεδα δικαιωμάτων, ο ελεγκτής εκτελείται στο επίπεδο 0, το Dom0 καθώς και τα DomUs εκτελούνται στο επίπεδο 1 και οι εφαρμογές χρήστη στο επίπεδο 3. Όταν τα φιλοξενούμενα λειτουργικά συστήματα επιθυμούν να εκτελέσουν μια εντολή σε αυξημένα δικαιώματα, επικοινωνούν με τον Xen μέσω μιας υπερκλήσης (Hypercall) και εκτελείται η εντολή από τη πλευρά του Xen.

Διαχείριση πόρων

Η αρχική ανάθεση μνήμης για κάθε εικονική μηχανή καθορίζεται τη στιγμή της δημιουργίας της. Αυτό σημαίνει ότι η μνήμη διαχωρίζεται στατικά μεταξύ των εικονικών μηχανών (DomU), παρέχοντας ισχυρή απομόνωση. Αν όμως ένα DomU χρειαστεί περισσότερη μνήμη μπορεί να διεκδικήσει επιπρόσθετες σελίδες από το Xen, μέχρι να φτάσει κάποιο καθορισμένο όριο. Αντίθετα, αν το DomU επιθυμεί να αποταμιεύσει μνήμη που δε χρησιμοποιεί μπορεί να απελευθερώσει σελίδες και να τις επιστρέψει στο Xen. Τα τροποποιημένα φιλοξενούμενα λειτουργικά συστήματα περιέχουν έναν ειδικό οδηγό, τον ballon driver, ο οποίος παρέχει τις δύο λειτουργίες που μόλις περιγράφηκαν. Επειδή το Xen δεν εγγυάται ότι θα παραχωρήσει συνεχόμενες περιοχές μνήμης (φυσική μνήμη, physical memory) στα φιλοξενούμενα λειτουργικά συστήματα, τα ίδια δημιουργούν την ψευδαίσθηση ότι κατέχουν συνεχόμενη φυσική μνήμη (ψευδο-φυσική μνήμη, pseudo-physical memory). Οι αντιστοιχίσεις μεταξύ φυσικών και ψευδοφυσικών διευθύνσεων είναι στην αποκλειστική ευθύνη του φιλοξενούμενου λειτουργικού συστήματος, το οποίο μπορεί απλά να τις συντηρεί σε έναν πίνακα. Το Xen παρέχει αποδοτική αντιστοίχηση μεταξύ των διευθύνσεων αυτών, αφού συντηρεί έναν διαμοιραζόμενο πίνακα μεταξύ των Doms, μέσω του οποίου ελέγχει την εγκυρότητά τους. Θα πρέπει να τονιστεί, ότι τα φιλοξενούμενα λειτουργικά συστήματα έχουν επίγνωση ότι η μνήμη που τους ανατίθεται δεν είναι συνεχής, σε αντίθεση με αυτό που συμβαίνει στο KVM και έτσι αποφεύγεται το επιπλέον κόστος συντήρησης δομών, όπως οι shadow page tables.

Όσον αφορά τον διαμοιρασμό της επεξεργαστικής ισχύς, το Xen χρησιμοποιεί τον αλγόριθμο χρονοδρομολόγησης Borrowed Virtual Time (BVT) [8]. Επιλέχθηκε αυτός ο αλγόριθμος διότι είναι γρήγορος και έχει μία ειδική τεχνική

που επιτρέπει την άμεση ενεργοποίηση (dispatch) ενός dom όταν αυτό λαμβάνει ένα γεγονός (event).

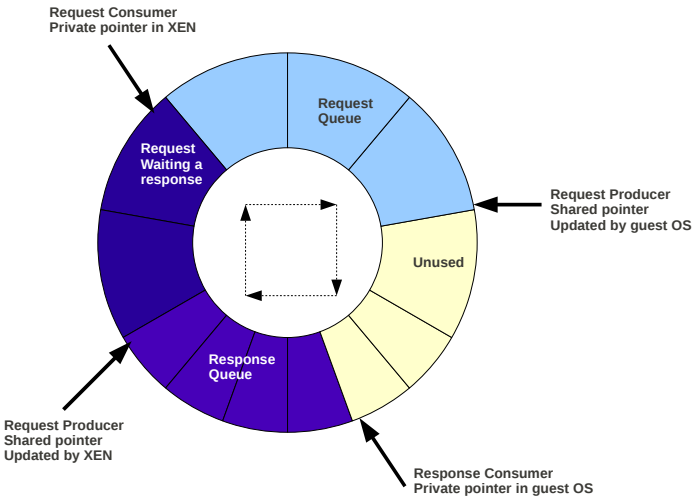
Η διαμοιραζόμενη πρόσβαση στις συσκευές εισόδου/εξόδου γίνεται μέσω των τροποποιημένων οδηγών, οι οποίοι ακολουθούν το πρότυπο του διαχωρισμένου οδηγού συσκευής (split driver model). Αυτό σημαίνει ότι ο οδηγός χωρίζεται σε δύο κομμάτια: το front-end που βρίσκεται στο φιλοξενούμενο λειτουργικό σύστημα ενός DomU και το back-end που βρίσκεται στο λειτουργικό σύστημα υπηρεσίας του Dom0. Πιο συγκεκριμένα ένας οδηγός συσκευής του συστήματος Xen αποτελείται από τέσσερις βασικές μονάδες:

- Τον πραγματικό οδηγό.
- Το κάτω μισό του διαχωρισμένου οδηγού.
- Τους μοιραζόμενους αποθηκευτικούς χώρους σε δομή δακτυλιδιού.
- Το πάνω μισό του διαχωρισμένου οδηγού.

Ο πραγματικός οδηγός, που βρίσκεται στο λειτουργικό σύστημα υπηρεσίας του Dom0 όπως έχει αναφερθεί, είναι ένας κοινός οδηγός που υπάρχει στα συμβατικά λειτουργικά συστήματα. Αυτό που αλλάζει είναι ότι ο ελεγκτής λαμβάνει τις διακοπές που προέρχονται από τη συσκευή και τις μετατρέπει σε γεγονότα του Xen (Xen events). Το κάτω μισό του διαχωρισμένου οδηγού επιτελεί δύο λειτουργίες: παρέχει πολυπλεξία και μία γενική διεπαφή του οδηγού. Το πρώτο επιτρέπει σε πάνω από μία εικονική μηχανή να χρησιμοποιεί τη συσκευή ενώ το δεύτερο παρέχει μια διεπαφή που είναι ανεξάρτητη από το υλισμικό, όπως η ανάγνωση και η εγγραφή μπλοκ δεδομένων σε μια συσκευή μπλοκ. Το πάνω μισό του διαχωρισμένου οδηγού είναι συνήθως πολύ απλό, αφού αρμοδιότητά του είναι να παρέχει στον χρήστη μια εικονική διεπαφή του οδηγού (τις διαθέσιμες λειτουργίες του οδηγού) και να μεταβιβάζει τις εντολές και τα δεδομένα του χρήστη από το DomU, στο οποίο βρίσκεται, στο Dom0. Η μεταβίβαση αυτή γίνεται μέσω των κοινών αποθηκευτικών χώρων (ring buffers), των καναλιών γεγονότων (event channels) και της βάσης δεδομένων XenStore του συστήματος Xen, τα οποία θα εξηγηθούν στη συνέχεια.

Επικοινωνία μεταξύ Doms

Όπως ειπώθηκε παραπάνω, απαραίτητο στοιχείο της υλοποίησης των διαχωρισμένων οδηγών είναι η επικοινωνία μεταξύ του Dom0 και των DomUs. Η μεταφορά των δεδομένων μεταξύ τους, δηλαδή η μεταφορά των εντολών του χρήστη αλλά και της καθαρής πληροφορίας γίνεται μέσω μιας διαμοιραζόμενης θέσης μνήμης που είναι σε δομή δακτυλιδιού.



Σχήμα 2.12: Ring

Η δομή του δακτυλιδιού είναι μια κυκλική ουρά με τέσσερις δείκτες και δυο ειδών τύπου δεδομένων ως στοιχεία της (Σχήμα 2.12). Πιο συγκεκριμένα, τα στοιχεία του δακτυλίου μπορεί να είναι είτε αιτήματα (requests) είτε απαντήσεις (responses). Για παράδειγμα, αν ο ρόλος του δακτυλίου είναι να μεταφέρει δεδομένα από το DomU στο Dom0, τα αιτήματα είναι δομές (structs) που μεταφέρουν δεδομένα προς αυτή τη κατεύθυνση και οι απαντήσεις είναι δομές που μεταφέρουν αποκρίσεις των αιτημάτων προς την αντίστροφη κατεύθυνση. Τα ονόματα των τεσσάρων δεικτών που συμμετέχουν είναι : παραγωγός αιτημάτων (request producer - rp), καταναλωτής αιτημάτων (request consumer - rc), παραγωγός απαντήσεων (response producer - rp), καταναλωτής απαντήσεων (response consumer - rc). Σε αντιστοιχία με το παραπάνω παράδειγμα, ο πρώτος και ο τελευταίος θα ανήκουν στο DomU και οι άλλοι δύο στον Dom0. Όταν προστίθεται ένα αίτημα ή μια απάντηση ο αντίστοιχος παραγωγός δείχνει μια θέση μπροστά (όπου και τοποθετείται η νέα δομή) και όταν εξυπηρετείται ένα αίτημα ή μια απάντηση ο αντίστοιχος καταναλωτής προωθείται μια θέση μπροστά, δείχνοντας στην επόμενη δομή προς εξυπηρέτηση.

Η έννοια της διαμοιραζόμενης μνήμης σημαίνει ότι και τα δύο Dom που συμμετέχουν έχουν δικαίωμα να προσπελάσουν τη μνήμη αυτή ανά πάσα στιγμή. Για να οριστεί ένα μέγεθος μνήμης (συνήθως σελίδες) ως διαμοιραζόμενο χρησιμοποιείται ο μηχανισμός της παραχώρησης σελίδων (grant pages) του Xen, το οποίο συντηρεί έναν πίνακα παραχώρησης (grant table) όπου αναγράφεται ποια σελίδα ανήκει σε ποιο Dom. Συνήθως, λοιπόν, το DomU δεσμεύει μια σελίδα, την αρχικοποιεί ως δομή δακτυλίδι, ορίζει τις δομές των αιτημάτων και

των απαντήσεων, την παραχωρεί στον Dom0 και έπειτα ο Dom0 την αποδέχεται.

Πως όμως ο Dom0 γνωρίζει ποια σελίδα να αποδεχτεί και πότε; Γι' αυτό το σκοπό χρησιμοποιείται η βάση δεδομένων του Xen, η XenStore. Η XenStore έχει μια δεντρική μορφή, στα φύλλα των οποίων υπάρχουν πεδία με τις αντίστοιχες τιμές τους. Τα πεδία αυτά αποτελούν παραμέτρους λειτουργίας των Doms και είναι ορατά από όλα τα Doms. Επίσης κάθε Dom μπορεί να τοποθετήσει στους κόμβους που το ενδιαφέρουν έναν παρατηρητή (watch) με την αντίστοιχη συνάρτηση εξυπηρέτησης, ο οποίος θα πυροδοτείται κάθε φορά που μεταβάλλεται ένα από τα πεδία των φύλλων που ακολουθούν. Μόλις πυροδοτείται ο παρατηρητής, η αντίστοιχη συνάρτηση εξυπηρέτησης εκτελείται. Στη συγκεκριμένη περίπτωση λοιπόν δημιουργείται ένα φύλλο στο XenStore στο οποίο θα αποθηκευτεί ο αριθμός της σελίδας που πρόκειται να παραχωρηθεί. Το Dom0 τοποθετεί έναν παρατηρητή σε αυτό το φύλλο και προσαρμόζει την κατάλληλη συνάρτηση εξυπηρέτησης. Το DomU ενημερώνει το φύλλο με τον αριθμό της σελίδας που παραχωρεί. Μόλις συμβεί αυτό, ο παρατηρητής ενεργοποιείται και εκτελείται η συνάρτηση εξυπηρέτησης. Μέσα στη συνάρτηση αυτή γίνεται η ανάγνωση του πεδίου και έπειτα η αποδοχή της σελίδας αυτής από το Dom0. Δεν μπορεί να χρησιμοποιηθεί η XenStore για την ανταλλαγή δεδομένων μεταξύ των Doms, δηλαδή δε μπορεί να υποκαταστήσει τον δακτύλιο, αφού στα φύλλα της αποθηκεύονται τιμές περιορισμένου μεγέθους και πάντα είναι τύπου συμβολοσειράς (string). Στη XenStore αποθηκεύονται μόνο λειτουργικές παράμετροι.

Τέλος, πως ο Dom0 ενημερώνεται για την έλευση καινούριων αιτήσεων στο δακτύλιο; Για τον σκοπό αυτό δημιουργείται ένα σύστημα ενημέρωσης μεταξύ του Dom0 και του DomU,που αποτελείται από έναν δίαυλο γεγονότων, δύο θύρες (μία σε κάθε άκρο του διαύλου) και δύο συναρτήσεις εξυπηρέτησης (μία σε κάθε θύρα του διαύλου). Στο συγκεκριμένο παράδειγμα, ο Dom0 δημιουργεί μία θύρα (port) στην οποία προσαρμόζει έναν δίαυλο γεγονότων, που η δεύτερη πόρτα δεν είναι ακόμα γνωστή και ενημερώνει το κατάλληλο φύλλο της XenStore με τον αριθμό της θύρας. Επειδή το DomU έχει ήδη τοποθετήσει έναν παρατηρητή στο φύλλο αυτό, ενεργοποιείται η αντίστοιχη συνάρτηση εξυπηρέτησης, στην οποία διαβάζεται το φύλλο αυτό, δημιουργείται η θύρα του DomU (η δεύτερη θύρα του διαύλου) και ενώνεται με τη θύρα του Dom0 μέσω του διαύλου. Όταν, λοιπόν, το DomU τοποθετεί ένα αίτημα στο δακτύλιο, στέλνει ένα γεγονός μέσω του διαύλου στο Dom0. Η συνάρτηση εξυπηρέτησης του γεγονότος ενεργοποιείται και το Dom0 καταναλώνει το αίτημα.

Επίδοση

Ο ελεγκτής εικονικής μηχανής Xen υιοθετεί τη τεχνική της μερικής εικονικοποίησης και έτσι υιοθετεί να μειονεκτήματα και τα πλεονεκτήματα της τεχνικής αυτής. Αν και δεν μπορεί να φιλοξενήσει λειτουργικά κλειστού κώδικα, μπορεί να επιτύχει απόδοση που πλησιάζει αυτή των φυσικών συστημάτων. Αυτό συμβαίνει διότι οι εικονικές μηχανές έχουν επίγνωση ότι εκτελούνται σε εικονικό περιβάλλον και έτσι υπάρχει δυνατότητα να παρακάμψουν επίπεδα που παρεμβάλονται ανάμεσα σε αυτά και το υλισμικό. Με άλλα λόγια υπάρχει δυνατότητα άμεσης πρόσβασης στο υλισμικό, ειδικά αν αυτό υποστηρίζει εικονικά περιβάλλοντα. Για τον λόγο αυτό ο Xen χρησιμοποιείται από προγραμματιστές που επιθυμούν να επιτύχουν υψηλές επιδόσεις σε εικονικά περιβάλλοντα.

2.2 Linux

Σε αυτό το κεφάλαιο θα γίνει μια σύντομη περιγραφή του λειτουργικού συστήματος Linux και των λειτουργιών αυτών που χρησιμοποιήθηκαν στην παρούσα διπλωματική.

Το λειτουργικό σύστημα Linux είναι ένα δωρεάν και ανοιχτό λογισμικό, του οποίου ο κώδικας μπορεί να χρησιμοποιηθεί, να τροποποιηθεί και να επαναδιανεμηθεί, έχοντας την άδεια του GNU General Public License. Ακριβώς επειδή είναι ανοιχτού λογισμικού, το Linux χρησιμοποιείται για εκπαιδευτικούς αλλά και για ερευνητικούς σκοπούς.

Ένα από τα βασικά κομμάτια του Linux είναι η δημιουργία ενός οδηγού (driver). Οι οδηγοί είναι ξεχωριστά "μαύρα κουτιά" που κάνουν ένα συγκεκριμένο κομμάτι υλισμικού να αποκρίνεται σε μια καλώς ορισμένη εσωτερική διεπαφή και αποκρύπτουν πλήρως τις λεπτομέρειες της λειτουργίας της συσκευής προς τον χρήστη. Οι ενέργειες του χρήστη, που έχουν σχέση με αυτή τη συσκευή, εκτελούνται μέσω ενός συνόλου προκαθορισμένων κλήσεων που είναι ανεξάρτητες από τον συγκεκριμένο οδηγό. Η αντιστοίχιση αυτών των κλήσεων σε συγκεκριμένες λειτουργίες της συσκευής, που δρουν στο πραγματικό υλισμικό είναι αρμοδιότητα του οδηγού. Γενικά, ένας οδηγός μπορεί να αντιστοιχεί σε πραγματική συσκευή υλισμικού αλλά μπορεί να αποτελεί απλώς μέσο επικοινωνίας μεταξύ των εφαρμογών του χρήστη και του πυρήνα. Στη συνέχεια παρουσιάζονται τα δομικά στοιχεία των οδηγών χαρακτήρων, μίας δημοφιλούς κατηγορίας οδηγών, αλλά και απαραίτητες πληροφορίες για τη

διαχείριση της μνήμης, η οποία εμπλέκεται πολλές φορές στη δημιουργία των οδηγών.

2.2.1 Οδηγοί χαρακτήρων

Οι οδηγοί χαρακτήρων χρησιμοποιούνται για συσκευές στις οποίες ή από τις οποίες τα δεδομένα μεταδίδονται ανά χαρακτήρα κάθε χρονική στιγμή (ποντίκι, πληκτρολόγιο). Επίσης χρησιμοποιούνται συχνά για τη δημιουργία επικοινωνίας μεταξύ πυρήνα και εφαρμογής χρήστη. Για τον σκοπό αυτό, δεν προτιμούνται οι κλήσεις συστημάτων, διότι για να ενσωματωθούν στον πυρήνα θα πρέπει να μεταγλωττιστούν μαζί του, ενώ ένας οδηγός χαρακτήρων μπορεί απλά να εισαχθεί ως λειτουργική μονάδα (module) κατά τη διάρκεια εκτέλεσης του πυρήνα.

Οι συσκευές χαρακτήρων (πραγματικές ή όχι) έχουν δύο αναπαραστάσεις. Η μία προορίζεται για το περιβάλλον χρήστη, έτσι ώστε ο ίδιος να γνωρίζει ποια και πως να τη προσπελάσει και η άλλη προορίζεται για τον πυρήνα. Η πρώτη ονομάζεται αρχείο συσκευής (device file), αποτελεί κόμβο του συστήματος αρχείων (filesystem) και ταυτίζεται με το όνομα του κόμβου αυτού. Βρίσκεται συνήθως στον φάκελο /dev και χαρακτηρίζεται μοναδικά από δύο αριθμούς, τον κύριο (major) και τον ελάσσονα (minor). Ο κύριος αριθμός καθορίζει τον οδηγό που ελέγχει τη συσκευή και ο ελάσσων αριθμός ξεχωρίζει τη συγκεκριμένη συσκευή από τις υπόλοιπες που μπορεί να ελέγχει ο ίδιος οδηγός. Η δεύτερη αναπαράσταση είναι η δομή με όνομα cdev (struct cdev). Ο πυρήνας, δηλαδή, συντηρεί μια δομή cdev για κάθε συσκευή χαρακτήρων. Ένα από τα πεδία που ορίζονται κατά τη δημιουργία της δομής είναι το ops. Το πεδίο αυτό περιέχει δείκτες στις συναρτήσεις (κλήσεις συστήματος) που παρέχουν τις λειτουργίες του οδηγού και τις οποίες μπορεί να καλεί ο χρήστης από τις εφαρμογές του. Οι συναρτήσεις αυτές είναι υποσύνολο ενός συγκεκριμένου συνόλου συναρτήσεων που πρέπει να παρέχει ένας οδηγός. Μερικές από αυτές τις συναρτήσεις είναι η open, που καλείται όταν ο χρήστης επιθυμεί να προσπελάσει την αντίστοιχη συσκευή και ανοίγει το αντίστοιχο αρχείο (/dev/), η read με την οποία ο χρήστης μπορεί να διαβάσει δεδομένα από τη συσκευή, η write με την οποία ο χρήστης μπορεί να γράψει δεδομένα στη συσκευή, και η close που καλείται όταν ο χρήστης τερματίσει την επεξεργασία της συσκευής και κλείσει το αντίστοιχο αρχείο.

Όταν καλούνται οι συναρτήσεις ενός οδηγού, το λειτουργικό σύστημα περνάει ως επιπρόσθετη παράμετρο μία δομή τύπου file. Η δομή αυτή δημιουργείται από το λειτουργικό σύστημα κατά την κλήση της συνάρτησης open,

διαγράφεται πάλι από το ίδιο κατά την κλήση της συνάρτησης close και περιέχει πεδία που περιγράφουν το συγκεκριμένο άνοιγμα της συσκευής. Η δομή αυτή δεν αφορά μόνο τους οδηγούς, αλλά κάθε ανοιχτό αρχείο που βρίσκεται στο σύστημα. Για να γίνει πιο κατανοητό, πρέπει να τονιστεί ότι υπάρχει μία δομή τύπου struct cdev που χαρακτηρίζει την συσκευή (πραγματική ή όχι) αλλά πολλές δομές τύπου file που αναφέρονται στη συσκευή αυτή και δημιουργούνται κάθε φορά που μία εφαρμογή χρήστη ζητά να προσπελάσει τη συσκευή. Αυτό συμβαίνει διότι κάθε προσπέλαση είναι μοναδική (η ανάγνωση ή η εγγραφή δεδομένων μπορεί να γίνεται με διαφορετικό ρυθμό) και έτσι πρέπει να ξεχωρίζει. Μερικά από τα πεδία που περιέχει είναι το f-mode, που περιγράφει το δικαίωμα ανάγνωσης ή εγγραφής της συσκευής, το f-pos, που δείχνει την τρέχουσα θέση εγγραφής ή ανάγνωσης στη συσκευή, το f-op, που δείχνει στο σύνολο των συναρτήσεων που περιγράφηκαν παραπάνω και το private-data, που δείχνει σε δεδομένα που αφορούν το συγκεκριμένο άνοιγμα της συσκευής.

Μία από τις συναρτήσεις των οδηγών που χρησιμοποιείται συχνά από τους προγραμματιστές είναι η ioctl (input output control). Κατά την κλήση της ο χρήστης περνάει έναν ακέραιο αριθμό και μία θέση μνήμης, η οποία πρόκειται να γραφτεί ή να διαβαστεί από το σώμα της ioctl συνάρτησης. Το σώμα της συνάρτησης αυτής αποτελείται από ένα βασικό switch ως προς τον ακέραιο αριθμό που λαμβάνει από τον χρήστη. Ο ακέραιος αυτός αριθμός ονομάζεται εντολή (command) και οι εναλλακτικοί ακέραιοι που μπορεί να χρησιμοποιήσει ο χρήστης αντιστοιχίζονται εννοιολογικά με ένα όνομα εντολής (define). Ανάλογα με την εντολή που έχει επιλέξει ο χρήστης εκτελείται το αντίστοιχο σύνολο εντολών του switch. Όπως γίνεται αντιληπτό, η συνάρτηση αυτή είναι αρκετά γενική και μπορεί να χρησιμοποιηθεί όταν είναι επιθυμητή η επικοινωνία με τον οδηγό ή τη συσκευή χωρίς την ανταλλαγή δεδομένων.

Μία ακόμα δημοφιλής συνάρτηση που χρησιμοποιείται από τους δημιουργούς των οδηγών είναι η mmap (memory mapping). Σκοπός της είναι η ταχύτερη επικοινωνία των εφαρμογών του χρήστη με το υλισμικό ή το περιβάλλον του πυρήνα. Η λειτουργία της συνάρτησης αυτής είναι να αντιστοιχίσει σε καθορισμένο μέγεθος μνήμης της περιοχής διευθύνσεων της διεργασίας το ίδιο μέγεθος μνήμης της περιοχής διευθύνσεων του οδηγού ή της συσκευής. Πιο συγκεκριμένα, ένα συνεχές μέρος των εικονικών διευθύνσεων (A) της διεργασίας αντιστοιχίζεται με τις φυσικές διευθύνσεις ενός ισομεγέθους κομματιού μνήμης (B) που ανήκει στον οδηγό (διεργασία του πυρήνα) ή με τις φυσικές διευθύνσεις ενός ισομεγέθους κομματιού (B') που ανήκει στην τοπική μνήμη της συσκευής (πραγματική). Όταν λοιπόν η διεργασία προσπελάσει το A θα προσπελαύνει ουσιαστικά το B ή το B'. Με άλλα λόγια δημιουργείται ένα κομ-

μάτι κοινής μνήμης μέσω της οποίας μπορούν να ανταλλαχθούν δεδομένα.

Κλείνοντας αυτή την υποενότητα θα αναφερθεί ένας μηχανισμός που υιοθετείται από όλες τις κατηγορίες οδηγών και που επιταχύνει τη διαδικασία μεταφοράς δεδομένων από το περιβάλλον χρήστη στη συσκευή υλισμικού και αντίστροφα. Ο μηχανισμός αυτός ονομάζεται DMA (Direct Memory Access). Πιο συγκεκριμένα ο DMA είναι ένας μηχανισμός υλισμικού που επιτρέπει στις περιφερειακές μονάδες να μεταφέρουν τα δεδομένα εισόδου/εξόδου απευθείας από και προς τη κύρια μνήμη, χωρίς να παρεμβάλλεται ο επεξεργαστής του συστήματος. Βασικές προϋποθέσεις για την έναρξη του μηχανισμού αυτού είναι η δέσμευση ενός κομματιού μνήμης (συνήθως σε σελίδες) στο χώρο χρήστη, η ενημέρωση του υλισμικού που θα ξεκινήσει τον μηχανισμό για τις αντιστοιχίσεις εικονικών-φυσικών διευθύνσεων που αποτελούν το κομμάτι και το ”καρφίτσωμα” των σελίδων (pin) στη κύρια μνήμη, έτσι ώστε να μην μεταφερθούν στο σκληρό δίσκο κατά τη διάρκεια της μεταφοράς των δεδομένων.

2.3 Επικοινωνία σε επίπεδο χρήστη

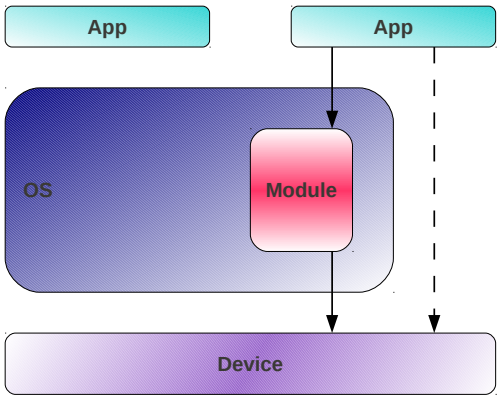
Όπως είναι γνωστό οι πολυνηματικές εφαρμογές προκειμένου να επωφεληθούν από την παραλληλοποίηση μπορούν να εκτελεστούν είτε σε έναν υπολογιστή με πολλούς πυρήνες, είτε σε μία ομάδα υπολογιστών (cluster) κάθε ένας από τους οποίους έχει μικρό αριθμό πυρήνων. Λαμβάνοντας υπόψη ότι οι πολυπύρηνοι υπολογιστές είναι ακριβότεροι από τα clusters, το ενδιαφέρον των προγραμματιστών έχει στραφεί στη βελτίωση του χρόνου εκτέλεσης των εφαρμογών αυτών κατά την εκτέλεσή τους στα clusters. Όμως, μεγάλο μέρος του χρόνου εκτέλεσης των πολυνηματικών εφαρμογών σπαταλάται στη μεταφορά των δεδομένων μεταξύ των κόμβων (υπολογιστών) των clusters, κάτι που γίνεται προσπάθεια να βελτιωθεί. Για τον λόγο αυτό έχουν αναπτυχθεί συστήματα διασύνδεσης υψηλής ταχύτητας που βελτιστοποιούν την μετάδοση δεδομένων μεταξύ των κόμβων.

Ένα από τα προβλήματα που καλούνται να επιλύσουν είναι η αντικατάσταση παραδοσιακών πρωτοκόλλων επικοινωνίας, όπως το TCP/IP με άλλα απλούστερα αφού η έκταση και η πολυπλοκότητα των δικτύων αυτών είναι πολύ μικρότερη από αυτή του διαδικτύου και έτσι δε δικαιολογείται η χρήση τους. Στις περισσότερες περιπτώσεις τα παραδοσιακά πρωτόκολλα απαιτούν η πρόσβαση στο δίκτυο να γίνεται μέσω του λειτουργικού συστήματος, το οποίο προσθέτει σημαντική καθυστέρηση στο μονοπάτι εκπομπής (transmission path) και στο μονοπάτι λήψης (receive path). Κατά την εκπομπή δεδομένων η εφαρμογή στο χώρο χρήστη διενεργεί μια κλήση συστήματος στη μέθοδο του πυρήνα που είναι υπεύθυνη για την αποστολή δεδομένων, έπειτα αντιγράφο-

νται τα δεδομένα από το χώρο χρήστη στο χώρο πυρήνα και τελικά η μέθοδος αυτή μεταφέρει τα δεδομένα στην κάρτα δικτύου. Κατά τη λήψη δεδομένων η κάρτα δικτύου ενεργοποιεί μια διακοπή προς τον πυρήνα, η κατάλληλη μέθοδος την εξυπηρετεί και λαμβάνει τα δεδομένα από την κάρτα, η εφαρμογή στον χώρο χρήστη διενεργεί μια κλήση συστήματος για να δηλώσει ότι αναμένει δεδομένα και έπειτα αντιγράφονται τα δεδομένα από το χώρο πυρήνα στο χώρο χρήστη [9]. Συνεπώς, προστίθεται καθυστέρηση λόγω των κλήσεων συστήματος, αφού αυτό συνεπάγεται αλλαγή της διεργασίας υπό εκτέλεση (context switch) και λόγω των πολλαπλών αντιγραφών των δεδομένων.

Για να αντιμετωπιστεί το πρόβλημα αυτό της απόδοσης έχουν αναπτυχθεί διάφορες αρχιτεκτονικές επικοινωνίας σε επίπεδο χρήστη (user level communication), που αφαιρούν το λειτουργικό σύστημα από το μονοπάτι της επικοινωνίας. Ένα από τα βασικότερα δομικά στοιχεία των αρχιτεκτονικών αυτών είναι ο μηχανισμός DMA, που αναφέρθηκε παραπάνω διότι αποτρέπει τις περιττές αντιγραφές δεδομένων μεταξύ χώρου χρήστη και χώρου πυρήνα και κάνει δυνατή την απευθείας μεταφορά δεδομένων μεταξύ χώρου χρήστη και κάρτα δικτύου.

Ένα από τα σημαντικά χαρακτηριστικά της επικοινωνίας σε επίπεδο χρήστη είναι λοιπόν η παράκαμψη του λειτουργικού συστήματος (OS-bypass). Σε αυτό το σημείο πρέπει να σημειωθεί ότι το OS-bypass δεν σημαίνει ότι όλες οι λειτουργίες εισόδου/εξόδου παρακάμπτουν το λειτουργικό σύστημα. Συνήθως, οι συσκευές επιτρέπουν το OS-bypass για συχνές και κρίσιμες σε θέμα χρόνου λειτουργίες, ενώ λειτουργίες που αφορούν εγκατάσταση και διαχείριση της επικοινωνίας περνούν μέσα από το λειτουργικό σύστημα μέσω μιας ειδικής λειτουργικής μονάδας (module) (Σχήμα 2.13).



Σχήμα 2.13: OS-bypass

Μία από τις μεγαλύτερες προκλήσεις της υλοποίησης του OS-bypass είναι η

ασφαλής πρόσβαση σε μια συσκευή που μοιράζεται από πολλές εφαρμογές. Ένας τρόπος για να επιτευχθεί αυτό είναι η χρήση συσκευών υλισμικού πιο έξυπνες από τις συμβατικές, που μπορούν να υποστηρίξουν το OS-bypass. Συνήθως, μια τέτοια συσκευή μπορεί να παρουσιάσει εικονικά σημεία πρόσβασης σε διαφορετικές εφαρμογές του χώρου χρήστη, μέσω των οποίων ρυθμίζεται η μεταφορά δεδομένων. Τα σημεία αυτά πρόσβασης που βρίσκονται στη μνήμη της συσκευής αντιστοιχίζονται στις εικονικές διευθύνσεις των εφαρμογών και έτσι αυτές μπορούν να τα προσπελαύνουν γρήγορα και με ασφάλεια, κάτι που το διασφαλίζει ο μηχανισμός της εικονικής μνήμης.

Το χαρακτηριστικό του OS-bypass έχει υιοθετηθεί από εμπορικά προϊόντα, πολλά από τα οποία έχουν γίνει δημοφιλή στη περιοχή των υπολογιστών υψηλών επιδόσεων (high performance computing), όπου η μικρή καθυστέρηση είναι ουσιαστική για τις εφαρμογές. Στα εμπορικά αυτά προϊόντα συγκαταλέγονται τα συστήματα διασύνδεσης υψηλής ταχύτητας που αναφέρθηκαν στην αρχή της ενότητας. Χαρακτηριστικά παραδείγματα αυτών είναι το InfiniBand και το Myrinet.

2.4 Δίκτυα υψηλών επιδόσεων σε εικονικό περιβάλλον

Παραπάνω αναφέρθηκε ότι οι πολυνηματικές εφαρμογές συνήθως εκτελούνται σε πολυπύρηνα μηχανήματα ή σε συστοιχίες υπολογιστών (clusters). Λαμβάνοντας υπόψη τις εισαγωγικές πληροφορίες των εικονικών μηχανών που ανέφεραν ότι υπάρχει τάση δημιουργίας συστοιχιών εικονικών μηχανών (clusters of virtual machines), γίνεται κατανοητό ότι οι πολυνηματικές εφαρμογές μπορούν να εκτελεστούν και σε ομάδες εικονικών μηχανών. Συνεπώς τα θέματα επιδόσεων μεταφέρονται από τις ομάδες υπολογιστών στις ομάδες εικονικών μηχανών. Με άλλα λόγια, γίνεται προσπάθεια να επιτευχθούν επιδόσεις παραπλήσιες με αυτές των πραγματικών συστοιχιών υπολογιστών, κυρίως στον τομέα της δικτυακής διασύνδεσης των εικονικών μηχανών. Σε αυτή την ενότητα, λοιπόν, θα παρουσιαστούν υλοποιημένες βελτιστοποιήσεις της επίδοσης των λειτουργιών δικτύου στις εικονικές μηχανές. Οι βελτιστοποιήσεις αυτές βασίζονται στον ελεγκτή εικονικής μηχανής Xen, οπότε θα χρησιμοποιείται αυτό το μοντέλο στα επεξηγηματικά διαγράμματα.

Πριν όμως αναφερθούν οι βελτιστοποιήσεις, θα πρέπει να περιγραφεί ο μη βελτιστοποιημένος τρόπος μετάδοσης των δεδομένων από τις εικονικές μηχανές στο διαδίκτυο. Το πάνω μισό του εικονικού οδηγού (front-end), που

βρίσκεται στο φιλοξενούμενο λειτουργικό σύστημα, αρχικά οργανώνει τα δεδομένα προς μετάδοση σε μηνύματα. Διατρέχοντας όλη τη στοίβα των πρωτοκόλλων δικτύου, τα μηνύματα αυτά καταλήγουν να μετατρέπονται σε πλαίσια του επιπέδου ζεύξης (όπως γίνονται στα συμβατικά υπολογιστικά συστήματα). Έπειτα, το front-end αποστέλλει να μηνύματα αυτά στο κάτω μισό του εικονικού οδηγού (back-end), που βρίσκεται στο λειτουργικό σύστημα υπηρεσίας. Η αποστολή των μηνυμάτων γίνεται μέσω της τεχνικής page sharing που περιγράφηκε στην ενότητα 2.1.2. Δηλαδή, κάθε φορά το front-end παραχωρεί στο back-end τη σελίδα που περιέχει το πλαίσιο προς αποστολή και το back-end την αποδέχεται. Όταν το back-end λαμβάνει τα μηνύματα, τα πολυπλέκει χρησιμοποιώντας μια εικονική γέφυρα (bridge) και χρησιμοποιώντας τον πραγματικό οδηγό της κάρτας δικτύου, τα αποστέλλει σε αυτή. Τέλος, η κάρτα δικτύου μεταβιβάζει τα πλαίσια στο φυσικό επίπεδο και τα αποστέλλει στο τοπικό δίκτυο. Η διαδικασία αυτή ακολουθείται αντιστρόφως όταν τα δεδομένα εισέρχονται στο σύστημα. Σε αυτό το σημείο πρέπει να σημειωθεί ότι η διεύθυνση mac των πλαισίων είναι η εικονική διεύθυνση mac της εκάστοτε εικονικής μηχανής και έτσι ο προορισμός τους και η προέλευσή τους είναι σαφώς ορισμένοι.

2.4.1 Αύξηση μεγέθους του πακέτου δεδομένων

Το φιλοξενούμενο λειτουργικό σύστημα και συγκεκριμένα το front-end, θα μπορούσε να στέλνει και να λαμβάνει μηνύματα πολύ μεγαλύτερα από αυτό που ορίζει το MTU (Maximum Transmission Unit). Αυτό που θα μπορούσε να γίνει λοιπόν είναι το front-end να δημιουργεί μεγάλα πακέτα δεδομένων, που αποτελούνται από μικρότερα τεμάχια (πλαίσια). Αυτά τα μεγάλα, τεμαχισμένα πακέτα θα μεταφέρονται από την εικονική στη φυσική διεπαφή μέσω ενός τροποποιημένου καναλιού επικοινωνίας και μιας τροποποιημένης γέφυρας. Δηλαδή, οι σελίδες που θα παραχωρούνται δεν θα αποτελούνται πια από ένα μόνο πλαίσιο, αλλά από μια συστάδα πλαισίων. Επίσης η γέφυρα θα τροποποιηθεί έτσι ώστε να χειρίζεται το νέο μέγεθος πακέτου. Μετά την πολύπλεξη των νέων πακέτων, το back-end θα πρέπει να τα προωθήσει στην κάρτα, αν αυτή έχει τη δυνατότητα να τα διασπάσει σε μικρότερα (MTU) ή να τα διασπάσει το ίδιο και έπειτα να τα προωθήσει στη κάρτα. Συνεπώς, αυξάνοντας το μέγεθος του πακέτου η επεξεργασία και η καθυστέρηση λόγω εικονικοποίησης ανά byte θα μειωθεί αφού η ενέργεια της παραχώρησης σελίδας, η οποία προσθέτει καθυστέρηση, θα γίνεται λιγότερες φορές [10].

2.4.2 Δημιουργία καναλιού επικεφαλίδων

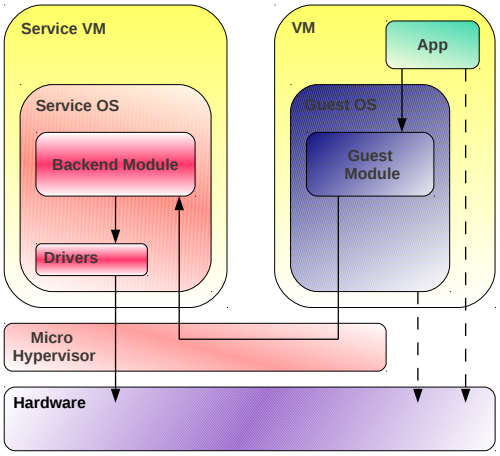
Έχει παρατηρηθεί ότι δεν χρειάζεται να παραχωρείται ολόκληρο το μήνυμα στο Dom0 από το DomU, μέσω της τεχνικής page grant αν το μήνυμα δεν προορίζεται για το Dom0 [10]. Πιο συγκεκριμένα, το Dom0 χρειάζεται μόνο την επικεφαλίδα των μηνυμάτων για να ελέγξει αν προορίζεται για το ίδιο και για να το προωθήσει, μέσω της γέφυρας, στην κάρτα δικτύου. Οι επικεφαλίδες των μηνυμάτων λαμβάνονται από το Dom0 μέσω ενός καναλιού επικεφαλίδων που δημιουργείται μεταξύ του ίδιου και του DomU, μέσω διαμοιραζόμενων σελίδων που προέκυψαν από τον μηχανισμό grant. Μόλις το backend λαμβάνει μια επικεφαλίδα, τη διαβάζει έτσι ώστε να προσδιορίσει αν θα πρέπει να λάβει στον χώρο της και το υπόλοιπο μήνυμα ή όχι (αν προορίζεται για αυτό ή όχι) και έπειτα δημιουργεί ένα μήνυμα δικτύου που αποτελείται από την επικεφαλίδα και τα κομμάτια του υπόλοιπου μηνύματος (φυσικές διευθύνσεις). Τέλος, η κάρτα δικτύου συγκεντρώνει τις φυσικές διευθύνσεις από τις οποίες αποτελείται το μήνυμα και εκκινεί τον μηχανισμό DMA για να μεταφερθεί ολόκληρο το μήνυμα σε αυτή. Συνεπώς, η τεχνική αυτή μειώνει τον χρόνο μετάδοσης των μηνυμάτων από το DomU στην κάρτα, αφού αντί των διαρκών, δαπανηρών παραχωρήσεων σελίδων που περιέχουν ολόκληρο το μήνυμα, χρησιμοποιείται ένα ήδη εγκατεστημένο κανάλι επικοινωνίας και ο μηχανισμός DMA.

2.4.3 Παράκαμψη του ελεγκτή εικονικής μηχανής

Η παράκαμψη του ελεγκτή εικονικής μηχανής (VMM-bypass) στις λειτουργίες εισόδου/εξόδου, επεκτείνει την ιδέα της παράκαμψης του λειτουργικού συστήματος (OS-bypass). Πιο συγκεκριμένα, η τεχνική VMM-bypass επιτρέπει σε λειτουργίες εισόδου/εξόδου ευαίσθητες σε καθυστερήσεις, να εκτελούνται άμεσα στις φιλοξενούμενες εικονικές μηχανές (guest VMs), χωρίς να παρεμβάλλεται ο ελεγκτής ή η εικονική μηχανή υπηρεσίας. Αυτή είναι μια σημαντική βελτιστοποίηση καθώς όταν παρεμβάλλεται ο ελεγκτής έχουμε αλλαγή στην κατάσταση και στη διεργασία εκτέλεσης (context switch), κάτι που προσθέτει καθυστέρηση και η απόδοση του συστήματος αποκλίνει κατά πολύ από την απόδοση του αντίστοιχου φυσικού συστήματος. Εκμεταλλευόμενοι επίσης την ευφυΐα των μοντέρνων, υψηλής ταχύτητας δικτυακών διεπαφών μπορεί να βελτιωθεί σημαντικά η επίδοση της επικοινωνίας χωρίς να θυσιαστεί η ασφάλεια ή η απομονωμένη εκτέλεση [11].

Για να εικονικοποιηθεί λοιπόν μια συσκευή προς την πλευρά της φιλοξενούμενης εικονικής μηχανής υλοποιείται ένας οδηγός συσκευής που ονομάζεται guest module και που τοποθετείται στο λειτουργικό σύστημα της φιλοξενούμενης εικονικής μηχανής. Η μονάδα αυτή είναι υπεύθυνη για τον χειρισμό των

προσβάσεων στη συσκευή, που απαιτούν αυξημένα δικαιώματα, καθώς επίσης, είναι υπεύθυνη για την αρχικοποίηση των δομών και την οργάνωση των λειτουργιών που απαιτούνται για την έναρξη της πρόσβασης στη συσκευή. Αυτό σημαίνει ότι θα πρέπει να έχει τη δυνατότητα να δημιουργεί εικονικά σημεία πρόσβασης και να τα αντιστοιχεί στις διευθύνσεις των εφαρμογών του χρήστη έτσι ώστε στη συνέχεια να μην παρεμβαίνει ο ελεγκτής. Από τη στιγμή όμως που η μονάδα αυτή αρχικά δεν έχει άμεση πρόσβαση στη συσκευή υλισμικού, θα πρέπει να εισαχθεί μια επιπλέον μονάδα, που ονομάζεται backend module και που θα παρέχει πρόσβαση στο υλισμικό για τις διαφορετικές μονάδες guest. Η μονάδα backend θα τοποθετηθεί στο λειτουργικό σύστημα υπηρεσίας και η επικοινωνία με τη μονάδα guest θα γίνεται μέσω των μηχανισμών που περιγράφτηκαν στην ενότητα 2.1.2 . Στο Σχήμα 2.14 παρουσιάζεται η αρχιτεκτονική της τεχνικής αυτής. Οι συνεχόμενες γραμμές συμβολίζουν το μονοπάτι που ακολουθούν οι λειτουργίες της αρχικοποίησης και της εγκατάστασης της επικοινωνίας μεταξύ φιλοξενούμενου λειτουργικού συστήματος και συσκευής υλισμικού, ενώ οι διακεκομμένες συμβολίζουν την άμεση επικοινωνία λειτουργικού και συσκευής στην ανταλλαγή δεδομένων.



Σχήμα 2.14: VMM-bypass

2.4.4 Έξυπνες κάρτες

Κάρτες δικτύου πολλαπλών ουρών

Μία συμβατική κάρτα δικτύου σχηματίζει στη μνήμη της δύο ουρές. Η μία προορίζεται για τα πακέτα που καταφθάνουν στην κάρτα και αναμένουν την μετάδοσή τους στο φυσικό μέσο (transmit queue), ενώ η άλλη προορίζεται για πακέτα που καταφθάνουν από το φυσικό μέσο και αναμένουν τη

προώθησή τους στο λειτουργικό σύστημα του υπολογιστή. Πρόσφατα, όμως, παρουσιάστηκαν κάρτες δικτύου νέας γενιάς, οι οποίες σχηματίζουν πολλαπλές ουρές μετάδοσης και λήψης για τα πακέτα. Αυτές ο κάρτες μπορούν να χρησιμοποιηθούν στα εικονικά περιβάλλοντα, αναθέτοντας κάθε ζεύγος ουρών αποκλειστικά σε μια εικονική μηχανή [12]. Η τεχνική αυτή απαλείφει τις καθυστερήσεις που προέρχονται από την αντιγραφή των μηνυμάτων, την πολύπλεξη/αποπολύπλεξη μηνυμάτων και την προστασία της μνήμης, αφού τα μηνύματα ανταλλάσσονται άμεσα μεταξύ των DomUs και των ουρών της κάρτας που τους αντιστοιχούν. Από τη άλλη πλευρά, η τεχνική αυτή προϋποθέτει την χρήση των συγκεκριμένων, εξειδικευμένων καρτών και θέτει ένα άνω όριο του αριθμού των εικονικών μηχανών που μπορεί να υποστηρίξει αφού ο αριθμός των ουρών είναι συγκεκριμένος.

Κάρτες SRIOV

Ένα άλλο είδος έξυπνης κάρτας που μπορεί να χρησιμοποιηθεί για τη βελτιστοποίηση της επικοινωνίας είναι οι SRIOV κάρτες. Οι κάρτες αυτές μπορούν να εξάγουν πολλαπλά σημεία πρόσβασης προς τις εικονικές μηχανές έτσι ώστε αυτές να έχουν τη ψευδαίσθηση ότι τους ανήκει ολοκληρωτικά η κάρτα. Έτσι, προκειμένου να υποστηρίξουν αυτή τη λειτουργία οι SRIOV κάρτες πολυπλέκουν τις διαφορετικές προσβάσεις στη κάρτα αποφορτίζοντας τον ελεγκτή εικονικής μηχανής από αυτή την αρμοδιότητα.

Κεφάλαιο 3

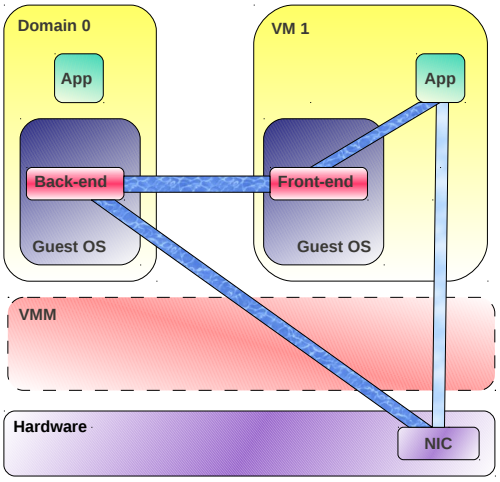
Σχεδιασμός συστήματος

Σε αυτό το κεφάλαιο περιγράφεται η σχεδίαση του συστήματος που πραγματεύεται η παρούσα διπλωματική. Αρχικά, δίνεται μια γενική οπτική της σχεδίασης έτσι ώστε ο αναγνώστης να αποκτήσει μια γενική εικόνα της λειτουργίας του συστήματος. Έπειτα παρατίθενται τα εργαλεία που χρησιμοποιήθηκαν για τη δημιουργία του συστήματος και αιτιολογείται η επιλογή τους. Τέλος, δίνεται μια πιο λεπτομερής περιγραφή της σχεδίασης καθώς αναφέρεται σε συγκεκριμένα εργαλεία και λειτουργίες.

3.1 Περιγραφή συστήματος

Σκοπός της σχεδίασης του συστήματος αυτού είναι η εισαγωγή των λειτουργιών των δικτύων υψηλής επίδοσης στο εικονικό περιβάλλον μέσω της δημιουργίας ενός κατάλληλου οδηγού. Με άλλα λόγια, σκοπός της είναι αρχικά η υποστήριξη αυτού του δικτύου και έπειτα η αύξηση της ταχύτητας μετάδοσης των δεδομένων από μία εφαρμογή του χρήστη, που βρίσκεται σε μια φιλοξενούμενη εικονική μηχανή, στη κάρτα δικτύου, που βρίσκεται στο υλισμικό και αντιστρόφως. Στο Σχήμα 3.1 φαίνεται το μονοπάτι που ακολουθούν τα δεδομένα στη συμβατική περίπτωση (σκουρόχρωμο) και το αντίστοιχο μονοπάτι που δημιουργεί το σύστημα αυτό (ανοιχτόχρωμο). Ο ελεγκτής εικονικής μηχανής (VMM) έχει σχεδιαστεί με διακοπτόμενες γραμμές διότι δε συμμετέχει στη μετάδοση δεδομένων.

Σύμφωνα με το Σχήμα 3.1 λοιπόν στη συμβατική περίπτωση τα δεδομένα μεταφέρονται από το χώρο χρήστη στο χώρο του πυρήνα και λαμβάνονται το πάνω μισό του οδηγού δικτύου. Το βήμα αυτό συνήθως περιλαμβάνει μια κλήση συστήματος προς τον πυρήνα και αντιγραφή των δεδομένων από το χώρο χρήστη στο χώρο πυρήνα. Έπειτα, τα δεδομένα μεταφέρονται από το πάνω



Σχήμα 3.1: Στόχος σχεδίασης

μισό στο κάτω μισό του οδηγού δικτύου, το οποίο βρίσκεται στην εικονική μηχανή υπηρεσίας ή πρωτεύουσα εικονική μηχανή. Η διαδικασία αυτή μπορεί να περιλαμβάνει κλήσεις στον ελεγκτή και μηχανισμούς ανταλλαγής δεδομένων μεταξύ των εικονικών μηχανών. Τέλος, τα δεδομένα μεταφέρονται από το κάτω μισό του οδηγού στη κάρτα δικτύου με την ίδια διαδικασία που ακολουθείται και στους συμβατικούς οδηγούς δικτύου. Συνεπώς, κατά τη διάρκεια μετάδοσης των δεδομένων παρεμβάλλονται χρονοβόρες διεργασίες όπως η αντιγραφή δεδομένων και οι κλήσεις στον πυρήνα και τον ελεγκτή που προσθέτουν καθυστερήσεις.

Από την άλλη πλευρά, στη βελτιστοποιημένη περίπτωση, τα δεδομένα μεταφέρονται απευθείας από τον χώρο χρήστη στη κάρτα δικτύου χωρίς επιπρόσθετες καθυστερήσεις. Για να επιτευχθεί όμως αυτό πριν από την μετάδοση λαμβάνουν χώρα αρχικοποιήσεις δομών και καταχωρήσεις πληροφοριών στον πυρήνα και τη κάρτα, έτσι ώστε να διασφαλιστεί η απομονωμένη και νόμιμη μεταφορά των δεδομένων. Συνεπώς, μειώνεται ο χρόνος μετάδοσης αλλά προστίθεται ένας χρόνος αρχικοποίησης, που όμως δεν επιβαρύνει τον πρώτο, διότι προηγείται της μετάδοσης και δε συμμετέχει στο κρίσιμο μονοπάτι. Επίσης, για να επιτευχθεί αυτή η βελτιστοποίηση χρησιμοποιούνται κατάλληλα εργαλεία που θα την υποστηρίξουν. Τα εργαλεία αυτά παρουσιάζονται στη συνέχεια.

3.2 Επιλογές σχεδίασης

3.2.1 Ελεγκτής εικονικής μηχανής

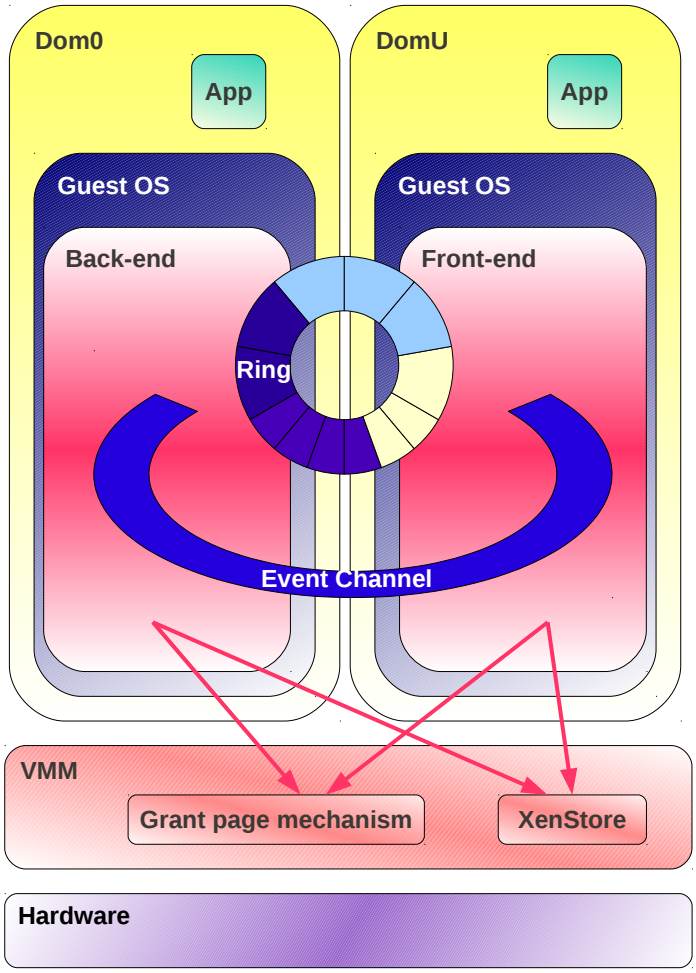
Το εικονικό περιβάλλον το οποίο χρησιμοποίησε το σύστημα είναι αυτό που δημιουργείται από τον ελεγκτή εικονικής μηχανής Xen. Όπως αναφέρθηκε στο Κεφάλαιο 2, ο ελεγκτής Xen είναι κυρίως τύπου I (υποτύπου Service OS) και υποστηρίζει την τεχνική της μερικής εικονικοποίησης. Επιλέχθηκε ο ελεγκτής Xen για τους εξής λόγους:

- Είναι ανοιχτού κώδικα και έτσι δίνεται η δυνατότητα να ενσωματωθεί σε αυτό το καινούριο σύστημα αλλά και να μελετηθεί ο κώδικας που έχει γραφτεί για παρόμοιες λειτουργίες.
- Η υποστήριξη της μερικής εικονικοποίησης παρέχει αφ' εαυτής καλύτερη απόδοση αφού μειώνει τα ενδιάμεσα επίπεδα που προσθέτει η εικονικοποίηση και προσδίδει μεγάλη ευελιξία στον προγραμματιστή κατά την συγγραφή των οδηγών υλισμικού.
- Είναι αρκετά δημοφιλής και έτσι υπάρχει πολύ καλή υποστήριξη σε θέματα συγγραφής οδηγών σε αυτόν.

Σε αυτό λοιπόν το εικονικό περιβάλλον δημιουργήθηκε ένας διαχωρισμένος οδηγός (split driver), που η λειτουργία του είναι η υποστήριξη ενός συγκεκριμένου δικτύου διασύνδεσης υψηλής επίδοσης που θα αναλυθεί στην επόμενη υποενότητα. Ο οδηγός αυτός αποτελείται από:

- το πάνω μισό οδηγού (front-end), που βρίσκεται στο λειτουργικό σύστημα μιας φιλοξενούμενης εικονικής μηχανής (DomU),
- το κάτω μισό οδηγού (back-end), που βρίσκεται στο λειτουργικό σύστημα της εικονικής μηχανής υπηρεσίας (Dom0),
- τους μηχανισμούς επικοινωνίας και ανταλλαγής δεδομένων μεταξύ των DomUs και του Dom0:
  - παραχώρηση σελίδας (grant page)
  - δακτύλιοι (rings)
  - δίαυλοι γεγονότων (event channels)
  - βάση δεδομένων του Xen (XenStore)

Το Σχήμα 3.2 παρουσιάζει τον γενικό σχεδιασμό του διαχωρισμένου οδηγού.



Σχήμα 3.2: Διαχωρισμένος οδηγός

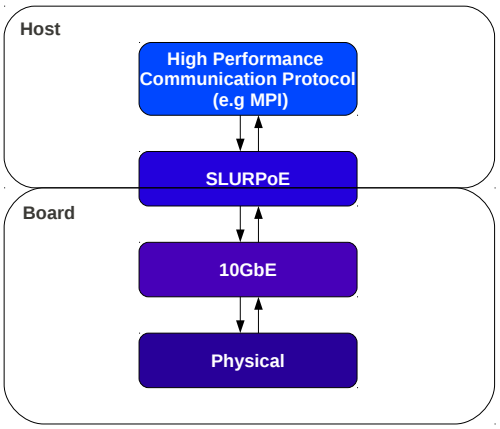
3.2.2 Δίκτυο διασύνδεσης υψηλής επίδοσης

Το δίκτυο διασύνδεσης υψηλής επίδοσης που χρησιμοποιήθηκε και του οποίου οι λειτουργίες εντάχθηκαν στο εικονικό περιβάλλον του Xen, είχε αναπτυχθεί από το εργαστήριο υπολογιστικών συστημάτων, κατά τη διάρκεια της εκπόνησης της διπλωματικής εργασίας με τίτλο 'Σχεδίαση και Υλοποίηση μηχανισμού απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10GbE' [13]. Το δίκτυο διασύνδεσης αυτό αποτελείτο από το πρωτόκολλο SLURPoE (Simple Lightweight User-level RDMA Protocol over Ethernet) και το υποκείμενο υλισμικό. Το πρωτόκολλο SLURPoE βρίσκεται πάνω από το επίπεδο ζεύξης και σκοπός του είναι η γρήγορη επικοινωνία μεταξύ των εφαρμογών υψηλών απαιτήσεων σε ταχύτητα δικτύου και του υλισμικού και θα αναλυθεί λεπτομερώς στην επόμενη υποενότητα.

Το υποκείμενο υλισμικό έχει επιλεχθεί έτσι ώστε να υποστηρίζει ως πρωτόκολλο επιπέδου ζεύξης το 10GbE (10 Gigabit per second Ethernet). Η τεχνολογία αυτή είναι το επόμενο βήμα στην κλιμάκωση της απόδοσης και της λειτουργικότητας των δικτύων Ethernet επειδή συνδυάζει το μεγάλο εύρος ζώνης με τη συμβατότητα του Ethernet με στόχο την επίτευξη κλιμακούμενων, έξυπνων και γρήγορων δικτύων. Πιο συγκεκριμένα, το υλισμικό που απαιτείται είναι συγκεντρωμένο σε μια κάρτα τύπου IQ81342SC (board). Η κάρτα αυτή είναι συνδεδεμένη με ένα γενικού σκοπού, 10GbE προσαρμογέα δικτύου και έχει τα εξής χαρακτηριστικά:

- επεξεργαστή XScale ARM,
- μνήμη RAM 256MB,
- τρεις μηχανές DMA για τη μεταφορά δεδομένων ανάμεσα στη κάρτα και στην εφαρμογή,
- μία μονάδα μηνυμάτων (Messaging Unit) για την ανταλλαγή μηνυμάτων ανάμεσα στη κάρτα και την αντίστοιχη λειτουργική μονάδα του λειτουργικού συστήματος,
- μία θύρα PCI-X,
- λειτουργικό σύστημα Linux 2.6.31 .

Το Σχήμα 3.3 παρουσιάζει πώς διαμορφώνεται η στοίβα των πρωτοκόλλων επικοινωνίας με τη χρήση αυτού του δικτύου διασύνδεσης και σε ποιά θέση του συστήματος εκτελείται το κάθε πρωτόκολλο.



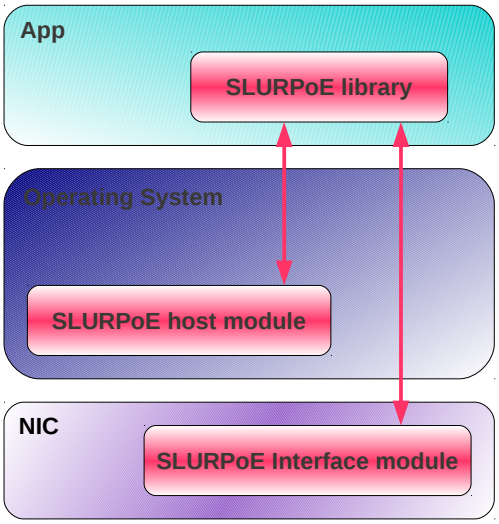
Σχήμα 3.3: Στοίβα πρωτοκόλλων

3.2.3 SLURPoE

Όπως αναφέρθηκε στην παραπάνω ενότητα, το πρωτόκολλο SLURPoE είναι ένα απλό, ”ελαφρύ” πρωτόκολλο επικοινωνίας που χρησιμοποιεί τη τεχνική RDMA (Remote Direct Memory Access) για τη μεταφορά δεδομένων και που βασίζεται στο πρωτόκολλο 10GbE. Λαμβάνοντας υπόψη κατά τη σχεδίαση ότι η συμμετοχή του λειτουργικού συστήματος στο κρίσιμο μονοπάτι (critical path) μετάδοσης δεδομένων μειώνει το ρυθμό μετάδοσης του 10GbE, τηρήθηκαν δύο αρχές:

- Το λειτουργικό σύστημα ελέγχει την νομιμότητα των λειτουργιών στο μονοπάτι ελέγχου (control path), που προηγείται της μετάδοσης δεδομένων.
- Η μετακίνηση των δεδομένων γίνεται απευθείας από το χώρο χρήστη στην κάρτα δικτύου, σχηματίζοντας το μονοπάτι δεδομένων (data path).

Το SLURPoE αποτελείται από το υλικολογισμικό (firmware) που βρίσκεται στον πυρήνα της κάρτας, μια λειτουργική μονάδα (module) που προσαρτάται στον πυρήνα του υπολογιστικού συστήματος και από μία βιβλιοθήκη χώρου χρήστη. Το Σχήμα 3.4 δείχνει τη στοίβα λογισμικού του SLURPoE.



Σχήμα 3.4: Στοίβα λογισμικού SLURPoE

Σκοπός του πρωτοκόλλου SLURPoE είναι η ασφαλής και απομονωμένη μεταφορά των δεδομένων από τον χώρο του χρήστη στη κάρτα και αντιστρόφως. Για να επιτευχθεί αυτό το ίδιο δημιουργεί δύο μονοπάτια, το μονοπάτι ελέγχου και το μονοπάτι δεδομένων. Στο μονοπάτι ελέγχου συγκαταλέγονται οι

λειτουργίες της αρχικοποίησης των δομών και της δέσμευσης της μνήμης που φιλοξενεί ή που θα φιλοξενήσει τα δεδομένα και συμμετέχουν και τα τρία κομμάτια λογισμικού (βιβλιοθήκη, λειτουργική μονάδα, υλικολογισμικό). Στο μονοπάτι δεδομένων συγκαταλέγεται μόνο η διαδικασία της μεταφοράς δεδομένων και συμμετέχει μόνο η βιβλιοθήκη (μέσω της εφαρμογής χρήστη) και το υλικολογισμικό.

Σε αυτό το σημείο θα πρέπει να τονιστεί η διαφορά του πρωτοκόλλου SLURPoE και γενικά των πρωτοκόλλων δικτύων υψηλής επίδοσης από τα συμβατά πρωτόκολλα επιπέδου δικτύου/μεταφοράς. Η λογική που διέπει τα δεύτερα είναι ότι τα δεδομένα οργανώνονται σε πλαίσια/δεδομενογράμματα και μεταβιβάζονται από τα ανώτερα στρώματα στο στρώμα ζεύξης, κατά την αποστολή, ή από το στρώμα ζεύξης στα ανώτερα στρώματα, κατά τη λήψη. Σε αντίθεση με αυτό το SLURPoE αρχικά δεσμεύει ένα κομμάτι μνήμης για να αποθηκευτούν τα δεδομένα που πρόκειται να σταλούν ή που πρόκειται να ληφθούν και έπειτα εκκινεί τη διαδικασία άμεσης μεταφοράς δεδομένων από το χώρο χρήστη στη κάρτα ή αντίστροφα. Η διαδικασία μετατροπής των δεδομένων σε πακέτα επιπέδου ζεύξης και αντιστρόφως αναλαμβάνεται από ευφυείς μηχανές που βρίσκονται στη κάρτα.

Αφού δόθηκε μια γενική περιγραφή του SLURPoE, τώρα θα παρατεθούν οι λειτουργίες που υποστηρίζει και θα ακολουθηθούν και τα δύο μονοπάτια που αυτό σχηματίζει (control path, data path).

Αρχικοποίηση

Το υλισμικό συσκευής που βρίσκεται στη κάρτα αρχικοποιεί τις μηχανές DMA, αρχικοποιεί τη διεπαφή του 10GbE, καθορίζει τις συναρτήσεις εξυπηρέτησης για τα αιτήματα που λαμβάνονται από το δίκτυο, το λειτουργικό σύστημα και τις εφαρμογές χρήστη και αρχικοποιεί τις θέσεις που λαμβάνουν τα αιτήματα και τις θέσεις που αποθηκεύουν τις αντιστοιχίσεις εικονική-φυσική διεύθυνση, που αντιστοιχούν στα τμήματα μνήμης υπό μεταφορά. Από την άλλη πλευρά η λειτουργική μονάδα που βρίσκεται στο λειτουργικό σύστημα, αρχικοποιεί τον μηχανισμό επικοινωνίας με τη κάρτα και ενημερώνεται για τη θέση μνήμης της κάρτας στην οποία θα πρέπει να γράφει τα αιτήματά της.

Δημιουργία σημείων πρόσβασης

Η λειτουργία αυτή συμμετέχει στο μονοπάτι ελέγχου και ως σκοπό έχει την ταυτοποίηση και την απόκτηση πρόσβασης στη κάρτα. Όταν ο χρήστης επιθυμεί να αποκτήσει πρόσβαση στη κάρτα έτσι ώστε στη συνέχεια να εκκινήσει μετακίνηση δεδομένων, τότε καλεί τη λειτουργία ανοίγματος άκρου

(open endpoint). Αμέσως ενεργοποιείται η κατάλληλη συνάρτηση εξυπηρέτησης της λειτουργικής μονάδας, η οποία μεταβιβάζει το αίτημα του χρήστη στην κάρτα στέλνοντάς της το κατάλληλο αίτημα. Έπειτα, το υλικολογισμικό της κάρτας διαβάζει το αίτημα, αρχικοποιεί τις δομές που αντιστοιχούν σε αυτό το άκρο, εκδίδει έναν μοναδικό αριθμό γι' αυτό το άκρο, το αντιστοιχεί με το PID της διεργασίας που το άνοιξε για λόγους ασφάλειας και επιστρέφει στη λειτουργική μονάδα, μέσω μηνύματος, τον μοναδικό αυτό αριθμό και τη θέση που πρέπει να ενημερωθεί με την αντιστοιχία εικονική-φυσική διεύθυνση της δεσμευμένης μνήμης. Μέσω της συνάρτησης εξυπηρέτησης διακοπών, η λειτουργική μονάδα λαμβάνει το μήνυμα, αποθηκεύει σε μια εσωτερική δομή τύπου άκρου τις πληροφορίες που συνέλεξε και επιστρέφει στον χρήστη τον μοναδικό αριθμό που αντιστοιχεί στο ανοιγμένο άκρο. Αντίστοιχη είναι η ροή κατά τη κατάργηση του άκρου όταν ο χρήστης έχει πια ολοκληρώσει τις ενέργειές του.

Δέσμευση μνήμης

Η λειτουργία της δέσμευσης μνήμης συμμετέχει κι αυτή στο μονοπάτι ελέγχου και σκοπός της είναι η καθήλωση του χώρου μνήμης που θα συμμετέχει στη μεταφορά δεδομένων και η καταχώρηση των αντιστοιχίσεων εικονική-πραγματική διεύθυνση του χώρου αυτού στη κάρτα έτσι ώστε αυτή να εκκινήσει τον μηχανισμό DMA για τη κατάλληλη φυσική διεύθυνση. Όταν ο χρήστης επιθυμεί να δεσμεύσει το χώρο μνήμης στον οποίο έχει ήδη εισάγει δεδομένα προς αποστολή ή στον οποίο πρόκειται να υποδεχτεί τα δεδομένα προς λήψη, τότε καλεί τη λειτουργία δέσμευσης μνήμης της βιβλιοθήκης (memory registration). Έτσι ενεργοποιείται η αντίστοιχη συνάρτηση εξυπηρέτησης της λειτουργικής μονάδας, η οποία καθηλώνει τον χώρο αυτό ώστε να μην μετακινηθεί από το λειτουργικό σύστημα και συμπληρώνει στη θέση μνήμης της κάρτας που αντιστοιχεί σε αυτό το άκρο την αντιστοιχία διευθύνσεων. Όταν ο χρήστης ολοκληρώσει τις εργασίες μεταφοράς δεδομένων, τότε καλεί την αντίστοιχη συνάρτηση αποδέσμευσης της μνήμης.

Εγγραφή δεδομένων

Η λειτουργία αυτή αποτελεί το μονοπάτι δεδομένων και διενεργείται κατά τη μεταφορά δεδομένων. Όταν ο χρήστης επιθυμεί να αποστείλει τα δεδομένα, τότε στέλνει απευθείας ένα αίτημα εγγραφής (RDMA write) στη κάρτα, που εκτός των άλλων περιέχει τον μοναδικό αριθμό του άκρου στο οποίο αποστέλλονται τα δεδομένα, την εικονική διεύθυνση του χώρου μνήμης που πρόκειται να αποθηκευτούν και την εικονική διεύθυνση του χώρου που βρίσκονται αυτή τη στιγμή. Έτσι, το υλικολογισμικό διαβάζει το μήνυμα, ανα-

τρέχει στην αντιστοιχία διευθύνσεων, βρίσκει τη φυσική διεύθυνση του χώρου και εκκινεί τον μηχανισμό DMA για τη μεταφορά των δεδομένων στη κάρτα. Αντιστρόφως, όταν τα δεδομένα λαμβάνονται από τη κάρτα, ανιχνεύεται ο αριθμός του άκρου και η εικονική διεύθυνση στην οποία προορίζονται, αναζητείται η αντίστοιχη φυσική διεύθυνση και εκκινείται ο μηχανισμός DMA για τη μεταφορά των δεδομένων στον δεσμευμένο χώρο του επιπέδου χρήστη.

3.3 Μεταφορά του πρωτοκόλλου SLURPoE στο περιβάλλον Xen

Σε αυτή την ενότητα παρουσιάζεται ο τρόπος με τον οποίο προσαρμόστηκε το πρωτόκολλο SLURPoE στο εικονικό περιβάλλον που δημιουργεί ο ελεγκτής εικονικής μηχανής Xen. Για την προσαρμογή αυτή παρέμεινε αμετάβλητο το υλισμικό του δικτύου διασύνδεσης και το υλικολογισμικό (firmware) που βρισκόταν στη κάρτα, καθώς αυτά είναι ανεξάρτητα του περιβάλλοντος που δημιουργείται στο κεντρικό σύστημα. Επίσης, οι λειτουργίες που παρέχονται από τη βιβλιοθήκη στον χρήστη παρέμειναν και αυτές αμετάβλητες, αφού η διαδικασία της εικονικοποίησης του πρωτοκόλλου θα πρέπει να είναι και είναι διαφανής προς τη πλευρά των εφαρμογών του χρήστη. Συνεπώς, αυτό που τροποποιήθηκε είναι η λειτουργική μονάδα (module) που ”ζούσε” στο λειτουργικό σύστημα και συμμετείχε στο μεγαλύτερο μέρος του μονοπατιού ελέγχου. Η τροποποίηση είχε να κάνει με τη διάσπαση των λειτουργιών της στο πάνω (front-end) και στο κάτω μισό (back-end) του νέου οδηγού που δημιουργήθηκε. Βασικά, λοιπόν, σχεδιαστικά ερώτημα είναι:

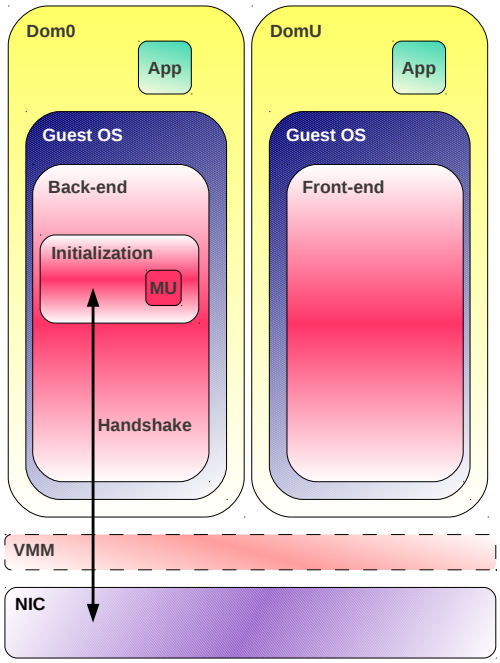
- Ποιες λειτουργίες εξαρτώνται από το υλισμικό, ώστε να μεταφερθούν στο back-end και ποιες από την εφαρμογή του χρήστη, ώστε να μεταφερθούν στο front-end;
- Ποιες δομές είναι κοινές για όλες τις προσβάσεις στην κάρτα, ώστε να μεταφερθούν στο back-end και ποιες δομές χαρακτηρίζουν και εξαρτώνται από τις προσβάσεις, ώστε να μεταφερθούν στο front-end;

Το ιδανικότερο σενάριο θα ήταν να μην μετατραπεί καθόλου η λειτουργική μονάδα του SLURPoE και να προσαρμοζόταν απευθείας στο λειτουργικό σύστημα των εικονικών μηχανών (DomUs). Όμως, όπως αναφέρθηκε στην παραπάνω ενότητα, υπάρχουν λειτουργίες που χρειάζεται να έχουν επίγνωση του πραγματικού συστήματος και έτσι θα πρέπει να μεταφερθούν στο Dom0, που παρέχει αυτή τη δυνατότητα. Επίσης, υπάρχουν δομές που χαρακτηρίζουν το υλισμικό και έτσι είναι πιο αποδοτικό να παραμείνουν στο Dom0 από το

ότι να μεταφέρονται κάθε φορά από το DomU στο Dom0, σε κάθε πρόσβαση στη κάρτα. Από την άλλη πλευρά υπάρχουν δομές που χαρακτηρίζουν τις προσβάσεις στη κάρτα (άκρα-endpoint) που θα πρέπει να βρίσκονται όσο πιο κοντά γίνεται στην εφαρμογή που τις δημιούργησε, δηλαδή θα πρέπει να μείνουν στο DomU και να μεταφέρονται στο Dom0 όποτε χρειάζεται. Με βάση λοιπόν αυτές τις σχεδιαστικές αποφάσεις προχώρησε η σχεδίαση των λειτουργιών που αποτελούν το SLURPoE και που αναφέρθηκαν στη προηγούμενη ενότητα.

3.3.1 Αρχικοποίηση

Το στάδιο της αρχικοποίησης εξυπηρετεί στην αρχικοποίηση των δομών που αφορούν την κάρτα και την έναρξη επικοινωνίας της λειτουργικής μονάδας με τη κάρτα. Αυτό το στάδιο λοιπόν αφορά το υλισμικό και γι΄ αυτό το λόγο μεταφέρθηκε στο back-end, στο Dom0. Το Σχήμα 3.5 παρουσιάζει αυτό το στάδιο.

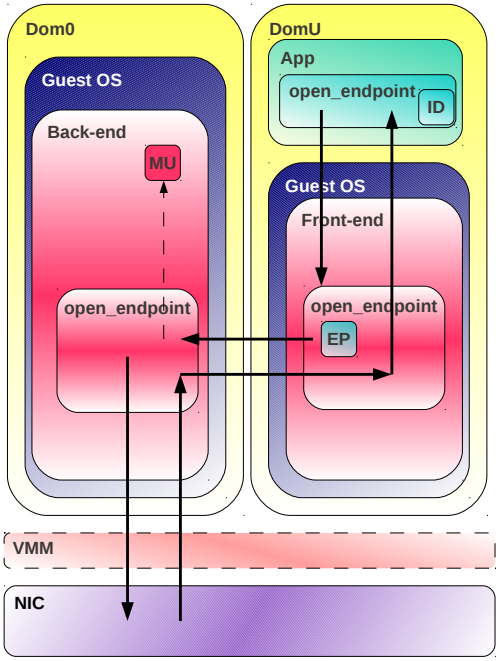


Σχήμα 3.5: Αρχικοποίηση

Το MU (Messaging Unit) συμβολίζει τη δομή που δημιουργείται στο back-end για να αποθηκεύσει τη θέση της κάρτα στην οποία αυτό θα μπορεί να γράφει τα μηνύματα που συμμετέχουν στο μονοπάτι ελέγχου.

3.3.2 Δημιουργία σημείων πρόσβασης

Όταν ο χρήστης επιθυμεί να αποκτήσει πρόσβαση στη κάρτα, καλεί την συνάρτηση βιβλιοθήκης που ανοίγει ένα άκρο πρόσβασης (open endpoint). Έπειτα, ενεργοποιείται η συνάρτηση εξυπηρέτησης του front-end, η οποία αρχικοποιεί τη δομή του άκρου που αντιστοιχεί σε αυτή τη πρόσβαση και μεταβιβάζει το αίτημα του χρήστη στο back-end. Στη συνέχεια, η κατάλληλη συνάρτηση εξυπηρέτησης του back-end ενεργοποιείται και γράφει στη θέση που υποδεικνύει το MU το μήνυμα-αίτημα του χρήστη, έτσι ώστε να το λάβει η κάρτα. Όταν η κάρτα επιστρέψει ένα μήνυμα-απόκριση, που περιέχει στοιχεία απαραίτητα για το άκρο (μοναδικός αριθμός, διεύθυνση εγγραφής των αντιστοιχιών), η κατάλληλη συνάρτηση εξυπηρέτησης της διακοπής ενεργοποιείται και μεταβιβάζει το μήνυμα-απόκριση στο front-end. Λαμβάνοντας το μήνυμα, το front-end, αποθηκεύει στη δομή άκρου που είχε δημιουργήσει τις πληροφορίες που ανέκτησε και επιστρέφει στην εφαρμογή του χρήστη τον μοναδικό αριθμό του άκρου που ανοίχτηκε.

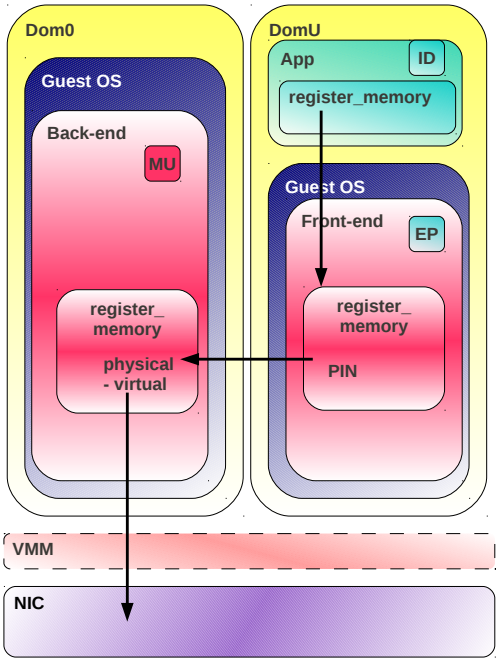


Σχήμα 3.6: Άνοιγμα άκρου

Στο Σχήμα 3.6 φαίνεται αυτή η λειτουργία. Το EP είναι η δομή endpoint που δημιουργείται και το ID είναι ο μοναδικός αριθμός που προσδίδεται στο άκρο.

3.3.3 Δέσμευση μνήμης

Όταν ο χρήστης επιθυμεί να δεσμεύσει μνήμη που περιέχει δεδομένα προς αποστολή ή που πρόκειται να δεχτεί δεδομένα, καλεί την αντίστοιχη συνάρτηση βιβλιοθήκης (register memory). Αμέσως, ενεργοποιείται η συνάρτηση εξυπηρέτησης του front-end, η οποία καθηλώνει τον χώρο μνήμης υπό δέσμευση (pin) και γνωστοποιεί στο back-end τον χώρο που καθηλώθηκε και τη θέση της κάρτας που πρέπει να γραφτεί η αντιστοίχηση διευθύνσεων. Έπειτα, το back-end βρίσκει την αντιστοίχιση φυσική-εικονική διεύθυνση του χώρου αυτού, αφού μόνο αυτό έχει επίγνωση της πραγματικής φυσικής μνήμης, και την καταχωρεί στην θέση που έλαβε από το DomU. Στο Σχήμα 3.7 παρουσιάζεται η λειτουργία αυτή.

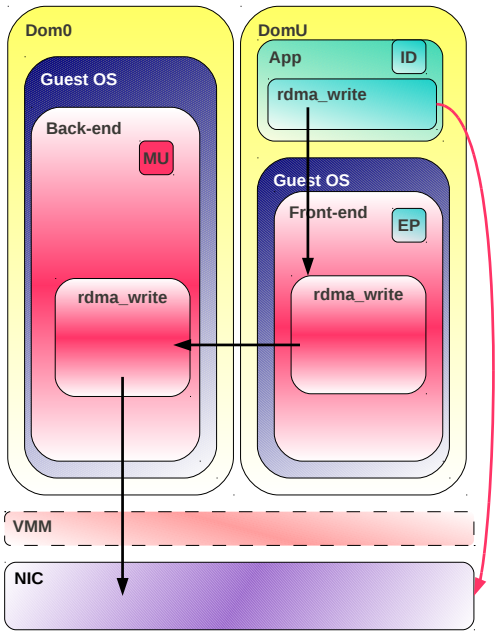


Σχήμα 3.7: Δέσμευση μνήμης

3.3.4 Εγγραφή δεδομένων

Για να μεταφερθούν τα δεδομένα από τον χώρο χρήστη στη κάρτα, ο χρήστης καλεί την συνάρτηση βιβλιοθήκης που θα πραγματοποιήσει αυτή τη μεταφορά (rdma write). Με τη κλήση αυτή περνάει σαν παραμέτρους την εικονική διεύθυνση του χώρου των δεδομένων, το άκρο στο οποίο προορίζονται τα δεδομένα και την εικονική διεύθυνση που αυτά πρόκειται να αποθηκευτούν. Έτσι, ενεργοποιείται η συνάρτηση εξυπηρέτησης του front-end, η οποία

μεταβιβάζει στο back-end το αίτημα του χρήστη για εγγραφή. Το back-end λαμβάνει το αίτημα και το γράφει στη θέση που ορίζεται από το MU ώστε να το λάβει η κάρτα. Μόλις συμβεί αυτό, η κάρτα εκκινεί τον μηχανισμό DMA για την αντίστοιχη φυσική μνήμη του χώρου, ώστε να μεταφερθούν τα δεδομένα απευθείας από τον χώρο χρήστη στην κάρτα. Στο Σχήμα 3.8 φαίνονται τα στάδια αυτής της λειτουργίας, όπου τα σκουρόχρωμα βέλη δείχνουν την πορεία του αιτήματος και το ανοιχτόχρωμο την πορεία των δεδομένων.



Σχήμα 3.8: Εγγραφή δεδομένων

Κεφάλαιο 4

Υλοποίηση συστήματος

Σε αυτό το κεφάλαιο εξηγείται λεπτομερώς ο τρόπος με τον οποίο υλοποιήθηκε η μεταφορά του πρωτοκόλλου SLURPoE στο εικονικό περιβάλλον του Xen. Η δομή που ακολουθείται είναι η κάθετη περιγραφή των λειτουργιών του SLURPoE που παρουσιάστηκαν στο προηγούμενο κεφάλαιο, δηλαδή ακολουθείται η ροή των μονοπατιών ελέγχου και δεδομένων καθώς αυτά διαπερνούν τους χώρους λειτουργίας (χώρος χρήστη, front-end, back-end, κάρτα). Έμφαση δίνεται σε τεχνικές λεπτομέρειες των οποίων η γνώση ήταν καθοριστική για την επίτευξη της υλοποίησης. Τέλος, σε ορισμένα, σημεία παρατίθενται εναλλακτικές υλοποιήσεις που θα μπορούσαν να είχαν γίνει έτσι ώστε να βελτιωθεί η απόδοση του συστήματος.

Στόχος της υλοποίησης ήταν:

- η όσο το δυνατόν μικρότερη τροποποίηση του κώδικα του SLURPoE,
- η επίτευξη διαφάνειας προς τη πλευρά του χρήστη και προς τη πλευρά της κάρτας, έτσι ώστε να μην έχουν επίγνωση ότι μεταξύ τους παρεμβάλλεται ένα εικονικό περιβάλλον,
- το στρώμα επικοινωνίας μεταξύ DomU και Dom0 να είναι όσο το δυνατόν πιο λεπτό ώστε οι λειτουργίες που προστίθενται να μην είναι χρονοβόρες και να μην επιβαρύνουν την απόδοση του αρχικού SLURPoE πρωτοκόλλου.

Ο διαχωρισμένος οδηγός που υλοποιήθηκε αποτελείται, όπως είναι γνωστό, από το front-end και το back-end. Το front-end και το back-end υλοποιήθηκαν σαν λειτουργικές μονάδες (modules), εισήχθησαν στον πυρήνα του DomU και στον πυρήνα του Dom0, αντίστοιχα, και ονομάστηκαν slurpoe-frontend και slurpoe-backend, αντίστοιχα. Τα λειτουργικά συστήματα που υποστήριζαν

όλα τα Doms ήταν έκδοσης Linux-2.6.31.13 και ο ελεγκτής Xen ήταν έκδοσης Xen-4.0-rc9.

4.1 Αρχικοποίηση

Σε αυτή την ενότητα εξηγούνται οι λειτουργίες που λαμβάνουν χώρα κατά την εισαγωγή των λειτουργικών μονάδων στους αντίστοιχους πυρήνες, έτσι ώστε να αρχικοποιήσουν τις δομές τους και τον τρόπο επικοινωνίας μεταξύ τους αλλά και ανάμεσα σε αυτά και την κάρτα.

4.1.1 Εισαγωγή slurpoe-back

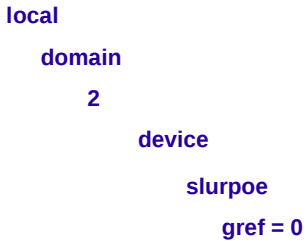
Η πρώτη ενέργεια που πραγματοποιείται κατά την εισαγωγή του slurpoe-back είναι η σύνδεση του, ως μια νοητή συσκευή, με το XenBus. Το XenBus είναι ένα εικονικό bus που χρησιμοποιεί το Xen έτσι ώστε οι εικονικοποιημένοι οδηγοί να επικοινωνούν μεταξύ των Doms. Δηλαδή, είναι ένας εικονικός δίαυλος (bus) που συνδέει τους οδηγούς που βρίσκονται στα διάφορα Doms έτσι ώστε να δημιουργείται μεταξύ τους ένα κανάλι ανταλλαγής ρυθμιστικών παραμέτρων, αφήνοντας τη μεταφορά δεδομένων να γίνεται μέσω της διαδικασίας παραχώρησης σελίδων (grant pages) και του διαύλου γεγονότων (event channel). Η εισαγωγή γίνεται μέσω της εντολής xenbus-register-backend, η οποία ως παράμετρο παίρνει μια δομή τύπου struct xenbus-driver. Η δομή αυτή περιέχει πεδία που χαρακτηρίζουν τον οδηγό που θα συνδεθεί με το XenBus, όπως το όνομά του (.name), τη συνάρτηση εξυπηρέτησης που θα κληθεί κατά τη σύνδεση (.probe) και τη συνάρτηση εξυπηρέτησης που θα κληθεί κατά την αποσύνδεση (.remove).

Ακολουθεί η εν μέρη εγκατάσταση ενός καναλιού δεδομένων μεταξύ του Dom0 και του DomU. Όπως έχει αναφερθεί στο Κεφάλαιο 2, το κανάλι αυτό αποτελείται από διαμοιραζόμενες σελίδες, που έχουν προέλθει από τον μηχανισμό grant και από τον δίαυλο δεδομένων. Ως προς τις διαμοιραζόμενες σελίδες, ακολουθούνται τα εξής βήματα:

- Το DomU δεσμεύει μια σελίδα μνήμης και την αρχικοποιεί έτσι ώστε να έχει δομή δακτυλίου.
- Μέσω μιας κλήσης στον ελεγκτή, ορίζει ότι την παραχωρεί στο Dom0.
- Για να ενημερωθεί ο Dom0 για την παραχώρηση, ο DomU γράφει σε ένα φύλλο του XenStore (Κεφάλαιο 2) τον αριθμό που ταυτοποιεί την σελίδα αυτή.

- Ο Dom0 έχει ήδη εγκαταστήσει ένα watch (Κεφάλαιο 2) στο φύλλο αυτό και έτσι με την ενημέρωσή του ενεργοποιείται η αντίστοιχη συνάρτηση εξυπηρέτησης η οποία αποδέχεται την σελίδα μέσω του αριθμού της.

Σε αυτό το σημείο πρέπει να επισημανθεί ότι αν δεν είχε εισαχθεί ο οδηγός στο XenBus, δε θα είχε τη δυνατότητα να τοποθετήσει το watch, να εγκαταστήσει διαύλους γεγονότων και να ενεργοποιήσει τις λειτουργίες τους. Αυτό, λοιπόν, που γίνεται κατά την εισαγωγή του slurpoe-back είναι η δημιουργία του φύλλου αυτού στο XenStore και η τοποθέτηση ενός watch σε αυτό. Η δημιουργία του φύλλου γίνεται με την εντολή xenbus-printf, η οποία παίρνει σαν παραμέτρους τη θέση στην οποία θα δημιουργηθεί το φύλλο, το όνομα του πεδίου (key) και τη τιμή του πεδίου (value). Η ίδια εντολή χρησιμοποιείται και για την ενημέρωση ενός πεδίου με την κατάλληλη τιμή. Στην προκειμένη περίπτωση, η θέση στην οποία θα δημιουργηθεί το φύλλο είναι της μορφής /local/domain/domU-id/device/slurpoe, όπου domU-id είναι ένας μοναδικός αριθμός που χαρακτηρίζει κάθε dom, που του προσδίδεται κατά την δημιουργία του. Σε αυτή τη περίπτωση το domU-id αναφέρεται στο DomU με το οποίο το Dom0 εγκαθιστά επικοινωνία. Το όνομα του πεδίου είναι gref (grant reference) και η τιμή του τίθεται ίση με 0 (θα ενημερωθεί αργότερα από το DomU). Το Σχήμα 4.1 δίνει μια εικονική αποτύπωση της θέσης του gref, υποθέτοντας ότι το id του DomU είναι 2.



Σχήμα 4.1: Το gref στο XenStore

Η τοποθέτηση του watch στο πεδίο gref γίνεται με την εντολή register-xenbus-watch, η οποία παίρνει σαν παράμετρο μια δομή τύπου struct xenbus-watch. Στη συγκεκριμένη περίπτωση το watch ονομάζεται xb-watch-gref και έχει τα εξής πεδία: το πεδίο στο οποίο τοποθετείται (/local/domain/domU-id/device/slurpoe/gref) και το όνομα της συνάρτησης που ενεργοποιείται κατά την ενημέρωση του gref, που ονομάζεται watch-gref.

Επόμενο βήμα της αρχικοποίησης είναι η δημιουργία του διαύλου γεγονότων (event channel). Για τον σκοπό αυτό χρησιμοποιείται η δομή τύπου struct

evtchn-alloc-unbound, η οποία περιγράφει ένα δίαυλο του οποίου έχει καθοριστεί μόνο το ένα άκρο. Η μεταβλητή που αναπαριστά αυτόν τον δίαυλο ονομάζεται evtchn και έχει τα εξής πεδία: το dom-id του Dom που εγκαθιστά τον δίαυλο και το dom-id του Dom με τον οποίο πρόκειται να συνδεθεί. Στη προκειμένη περίπτωση οι τιμές αυτές είναι 0 και 2, αντίστοιχα. Με την υπερκλήση (κλήση συστήματος που προορίζεται για τον ελεγκτή) HYPERVISOR-event-channel-op και με παραμέτρους την επιλογή EVTCHNOP-alloc- unbound και τον δίαυλο evtchn, εγκαθίσταται το ένα μέρος του διαύλου από τον ελεγκτή. Για να είναι χρήσιμος ο δίαυλος, στο ένα άκρο του πρέπει να συνδέεται μια συνάρτηση εξυπηρέτησης που θα επεξεργάζεται τα γεγονότα που καταφθάνουν, που στη προκειμένη περίπτωση ονομάζεται received-data-from-domU. Η σύνδεση αυτή πραγματοποιείται μέσω της κλήσης bind-evtchn-to-irqhandler, η οποία συγχρόνως επιστρέφει τον αριθμό της θύρας (port) στην οποία συνδέθηκε ο δίαυλος και στην οποία θα "ακούει" η received-data-from-domU.

Ο αριθμός αυτός της θύρας είναι μια πληροφορία την οποία πρέπει να γνωρίζει το DomU, αφού εκεί θα πρέπει να στέλνει τα γεγονότα. Πώς γίνεται όμως αυτό; Ακολουθείται η ίδια διαδικασία με αυτή του gref. Δηλαδή, στο μονοπάτι /local/domain/domU-id/device/slurpoe δημιουργείται το πεδίο port και του προσδίδεται η τιμή που επιστρέφεται από την κλήση bind-evtchn-to-irqhandler.

Σαν τελευταίο βήμα της εισαγωγής του slurpoe-back είναι η αρχικοποίηση του πρωτοκόλλου SLURPoE. Επειδή το SLURPoE αλληλεπιδρά με το υλισμικό είναι απαραίτητη η δημιουργία ενός PCI οδηγού. Η αναπαράσταση του οδηγού αυτού έρχεται μέσα από τη μεταβλητή slurpoe-pci-driver που είναι τύπου static struct pci-driver και έχει τα εξής πεδία: το όνομα του οδηγού (.name), την συνάρτηση εξυπηρέτησης που ενεργοποιείται όταν εντοπιστούν οι συσκευές που αντιστοιχούν σε αυτό τον οδηγό (.probe) και την περιγραφή των συσκευών που αντιστοιχούν στον οδηγό (.id-table), που στη προκειμένη περίπτωση είναι οι δύο κάρτες XScale. Για τη σύνδεση του οδηγού στο PCI καλείται η συνάρτηση pci-register-driver με παράμετρο το slurpoe-pci-driver.

Με την εισαγωγή του slurpoe-back, αμέσως εντοπίζονται οι δύο κάρτες και έτσι ενεργοποιείται η συνάρτηση probe. Με την ενεργοποίηση της probe αποθηκεύονται οι δομές που χαρακτηρίζουν τις συσκευές και πραγματοποιείται η πρώτη επικοινωνία με τις κάρτες (handshake). Πιο συγκεκριμένα, ορίζεται η συνάρτηση που θα εξυπηρετεί τις διακοπές που προέρχονται από τις κάρτες (slurpoe-mu-isr), προσδιορίζεται η θέση της κάθε κάρτας στη οποία θα εγγράφονται τα μηνύματα, την οποία θα ονομάζουμε MU (Messaging Unit) και

εκκινείται μια διαδικασία χειραψίας μεταξύ του οδηγού και των καρτών έτσι ώστε να επαληθευθεί η ορθότητα των ρυθμίσεων. Με άλλα λόγια, ο οδηγός επικοινωνεί με την κάρτα ως εξής:

- Ο οδηγός κατασκευάζει ένα μήνυμα και το γράφει στο MU.
- Η κάρτα ειδοποιείται με διακοπή ότι έλαβε μήνυμα και το διαβάζει.

Από την άλλη πλευρά, η κάρτα επικοινωνεί με τον οδηγό ως εξής:

- Η κάρτα κατασκευάζει ένα μήνυμα και το γράφει στο MU.
- Η κάρτα στέλνει διακοπή στον οδηγό.
- Ενεργοποιείται η slurpoe-mu-isr και διαβάζει το μήνυμα από το MU.

4.1.2 Εισαγωγή slurpoe-front

Το πρώτο βήμα της εισαγωγής του slurpoe-front, όπως και στο slurpoe-back, είναι η σύνδεσή του με το xenbus. Έπειτα εκκινείται η διαδικασία δημιουργίας του κοινού δακτυλίου, η οποία αποτελείται από τα εξής βήματα:

- Δέσμευση μνήμης, μεγέθους μίας σελίδας, μέσω της συνάρτησης get-free-pages.
- Μορφοποίηση της σελίδας σε δομή δακτυλίου, μέσω της μακροεντολής SHARED-RING-INIT.
- Απόκτηση μεταβλητής-χειριστή του δακτυλίου από την πλευρά του front-end, μέσω της μακροεντολής FRONT-RING-INIT.
- Δήλωση παραχώρησης της σελίδας στο Dom0, μέσω της συνάρτησης gnttab-grant-foreign-access, αφού πρώτα έχει βρεθεί το mfn (machine frame number) της σελίδας (virt-to-mfn).

Η συνάρτηση gnttab-grant-foreign-access επιστρέφει έναν αριθμό που ταυτοποιεί την σελίδα που παραχωρήθηκε. Αυτός ο αριθμός, στη συνέχεια, γράφεται στο πεδίο gref, η θέση του οποίου είναι γνωστή από το DomU. Με την ενημέρωσή του gref ενεργοποιείται η συνάρτηση watch-gref του Dom0, η οποία διαβάζει την τιμή του gref, μέσω της συνάρτησης xenbus-read, και αποδέχεται την σελίδα στην οποία αυτό αντιστοιχεί. Η αποδοχή αυτή αποτελείται από τα εξής βήματα:

- Δέσμευση εικονικής μνήμης μεγέθους μιας σελίδας, μέσω της alloc-vm-area, έτσι ώστε να αντιστοιχηθεί η κοινή σελίδα στην εικονική περιοχή διευθύνσεων του Dom0. Στην πραγματικότητα οι φυσικές διευθύνσεις της κοινής σελίδας ανήκουν στο DomU, όμως μέσω της διαδικασίας της παραχώρησης, το Dom0 αποκτά δικαίωμα προσπέλασης τους, αντιστοιχίζοντάς τες στις εικονικές διευθύνσεις του.

- Αρχικοποίηση δομής που περιγράφει την παραχώρηση. Η δομή αυτή είναι τύπου struct gnttab-map-grant-ref και τα πεδία της ορίζουν την αρχική εικονική διεύθυνση στην οποία αντιστοιχίζεται η κοινή σελίδα, τον αριθμό της κοινής σελίδας (gref), το id του Dom που την παραχώρησε και μία σημαία (flag) που ορίζει τον σκοπό για τον οποίο γίνεται η αποδοχή της σελίδας. Η σημαία μπορεί να είναι GNTMAP-device-map ή GNTMAP-host-map. Το πρώτο χρησιμοποιείται όταν η σελίδα προορίζεται για λειτουργίες εισόδου/εξόδου, ενώ το δεύτερο χρησιμοποιείται όταν το Dom0 επιθυμεί απλώς να προσπελάσει τα κοινά δεδομένα. Στην προκειμένη περίπτωση χρησιμοποιείται το δεύτερο. Βάζοντας τα παραπάνω πεδία σαν παραμέτρους της συνάρτησης gnttab-set-map-op, επιτυγχάνεται η αρχικοποίηση της δομής αυτής, που ονομάζεται ops.

- Αποδοχή της σελίδας μέσω της υπερκλήσης HYPERVISOR-grant-table-op, η οποία παίρνει σαν παράμετρο την σημαία GNTTABOP-map-grant-ref και τη δομή ops.

Αφού γίνει η αποδοχή, λαμβάνεται ένας χειριστής του δακτυλίου από την πλευρά του Dom0 καλώντας τη μακροεντολή BACK-RING-INIT, η οποία παίρνει ως παράμετρο την κοινή σελίδα.

Στο σημείο αυτό θα εξηγηθεί ο τρόπος με τον οποίο ορίζεται η δομή του δακτυλίου. Όπως είχε αναφερθεί στο Κεφάλαιο 2, ο δακτύλιος αποτελείται από αιτήματα (requests) και απαντήσεις (responses). Στην προκειμένη περίπτωση, οι δομές των αιτημάτων είναι ίδιες με τις δομές των απαντήσεων. Ορίζονται ως structs και περιέχουν δύο πεδία: το func, που ορίζει ποια λειτουργία εξυπηρετεί το μήνυμα (open endpoint, register memory, rdma write), και το data, που είναι ένωση (union) από δομές που αντιστοιχούν στη κάθε λειτουργία, αφού για διαφορετικό είδος μηνύματος μεταφέρονται διαφορετικά δεδομένα μεταξύ των Doms. Με την μακροεντολή DEFINE-RING-TYPES, ορίζεται η δομή του δακτυλίου σύμφωνα με την επιλογή των δομών των αιτημάτων και των απαντήσεων.

Επιστρέφοντας, τώρα, στην εισαγωγή του slurpoe-front, μετά την παραχώρηση του δακτυλίου ακολουθεί η τοποθέτηση ενός watch στο πεδίο port, που ενημερώνεται από το Dom0. Επειδή έχει προηγηθεί η εισαγωγή του Dom0 και αυτό έχει ενημερώσει το port, με το που τοποθετείται το watch, ενεργοποιείται η συνάρτηση εξυπηρέτησης (watch-port), η οποία διαβάζει το πεδίο port και ξεκινά τη διαδικασία ένωσης του διαύλου. Για αυτή τη διαδικασία χρησιμοποιείται η δομή struct evtchn-bind-interdomain μέσω της μεταβλητής evtchn. Αφού αρχικοποιηθούν τα πεδία της, που αφορούν το Dom με το οποίο θα συνδεθεί και με την απομακρυσμένη πόρτα (port) με την οποία θα συνδεθεί, πραγματοποιείται η υπερκλήση HYPERVISOR-event-channel-op, η οποία επιστρέφει την τοπική πόρτα του διαύλου. Πριν επιστρέψει η watch-port, συνδέεται η τοπική πόρτα του διαύλου με τη συνάρτηση εξυπηρέτησης received-data-from-dom0.

Τέλος, αρχικοποιείται η συσκευή χαρακτήρων (character device), με όνομα slurpoe-xen, μέσω της οποίας δημιουργείται η διεπαφή του πρωτοκόλλου SLURPoE προς τον χώρο χρήστη. Αφού επιλεχθεί ο MAJOR και ο MINOR αριθμός που χαρακτηρίζουν τη συσκευή και οριστεί η δομή fops που είναι τύπου struct file-operations (Κεφάλαιο 2), αρχικοποιείται η δομή της συσκευής χαρακτήρων, που είναι τύπου struct cdev, με την κλήση cdev-init και προσαρτάται στο λειτουργικό σύστημα με την κλήση cdev-add. Οι συναρτήσει που ορίζονται στη δομή fops είναι η open, που καλείται όταν ο χρήστης ανοίγει τη συσκευή slurpoe-xen, η ioctl, που καλείται κατά την έναρξη των λειτουργιών του SLURPoE, και η release, που καλείται όταν ο χρήστη κλείσει τη συσκευή slurpoe-xen.

4.2 Άνοιγμα άκρου

Όταν ο χρήστης επιθυμεί να αποκτήσει πρόσβαση στη κάρτα, τότε αρχικά ανοίγει τη συσκευή-αρχείο slurpoe-xen, μέσω της οποίας παρέχονται οι λειτουργίες του SLURPoE. Το άνοιγμα του αρχείου γίνεται μέσω της κλήσης συστήματος open, η συνάρτηση εξυπηρέτησης της οποίας αρχικοποιεί την δομή του άκρου (endpoint) που αντιστοιχεί σε αυτό το άνοιγμα. Τα πεδία που αρχικοποιούνται είναι η κατάσταση του άκρου σε SLURPOE-ENDPOINT-STATUS-FREE, το pid της διεργασίας που το άνοιξε και η κάρτα στην οποία πρόκειται να ανοιχτεί αυτό(αύξων αριθμός της κάρτας). Μετά την αρχικοποίηση, η δομή του άκρου τίθεται στο πεδίο private-data του περιγραφέα του αρχείου (struct file *).

Ο τρόπος με τον οποίο ο χρήστης ανοίγει ένα άκρο στο χώρο της εφαρμογής, είναι μέσω της κλήσης συστήματος ioctl (Κεφάλαιο 2). Πιο συγκεκριμένα, καλείται η ioctl με παραμέτρους το αρχείο που ανοίχθηκε (fd), την εντολή που αντιστοιχεί στο άνοιγμα του άκρου (SLURPOE-IOCTL-OPEN-ENDPOINT) και τη διεύθυνση της μεταβλητής που θα περιέχει τον μοναδικό αριθμό του άκρου, έτσι ώστε να ενημερωθεί από την κάρτα. Στη συνέχεια, ενεργοποιείται η συνάρτηση εξυπηρέτησης ioctl του οδηγού, η οποία με βάση τον κωδικό της εντολής οδηγείται στο αντίστοιχο μπλοκ εντολών. Αφού βρεθεί η δομή άκρου που αντιστοιχεί σε αυτό το άνοιγμα αρχείου και εφαρμοστούν ορισμένοι έλεγχοι ασφάλειας, η κατάσταση του άκρου τίθεται ίση με SLURPOE-ENDPOINT-STATUS-INITIALIZING και αρχίζει η διαδικασία κατασκευής μηνύματος για το Dom0. Καλείται, λοιπόν, η μακροεντολή RING-GET-REQUEST, η οποία επιστρέφει ένα δείκτη προς τη πρώτη διαθέσιμη δομή αιτήματος του δακτυλίου. Αφού αρχικοποιηθούν τα πεδία του αιτήματος, θέτοντας στο πεδίο func τον κωδικό MSG-ID-OPEN-ENDPOINT και στο πεδίο data δεδομένα που αντιστοιχούν σε αυτή τη λειτουργία (struct slurpoe-ring-msg-endpoint), όπως τον αύξοντα αριθμό της κάρτας που θα ανοιχτεί το άκρο, αποστέλλεται το αίτημα στο Dom0 με την εντολή RING-PUSH-REQUESTS. Για να ειδοποιηθεί το Dom0 ότι του στάλθηκε νέο αίτημα, αποστέλλεται ένα γεγονός μέσω του διαύλου που είχε δημιουργηθεί, με τη βοήθεια της κλήσης HYPERVISOR-event-channel-op, που συνοδεύεται από τη σημαία EVTCHNOP-send. Μετά την αποστολή, το DomU μπαίνει σε ένα βρόχο αναμονής (busy wait) μέχρι η κατάσταση του άκρου να γίνει SLURPOE-ENDPOINT-STATUS-OK, κάτι που θα προκύψει από την απάντηση του Dom0.

Μόλις λάβει το γεγονός το Dom0, ενεργοποιείται η συνάρτηση received-data-from-domU, η οποία αποκτά δείκτη στην πρώτη αίτηση προς εξυπηρέτηση μέσω της μακροεντολής RING-GET-REQUEST. Με βάση το πεδίο func της αίτησης έχει δημιουργηθεί ένα switch, που διαχωρίζει τις ενέργειας που αντιστοιχούν σε κάθε λειτουργία. Το μπλοκ εντολών που αντιστοιχούν στο άνοιγμα άκρου περιέχουν τις εξής ενέργειες:

- Αντιγραφή των πεδίων του αιτήματος, στα πεδία του μηνύματος που υποστηρίζει το SLURPoE. Τα μηνύματα που χρησιμοποιεί το SLURPoE για να επικοινωνήσει με την κάρτα είναι τύπου struct slurpoe-mu-msg-generic και έχουν δύο πεδία: το func που υποδεικνύει την λειτουργία και το data που είναι ένωση (union) δομών που αντιστοιχούν σε

κάθε λειτουργία. Στη συγκεκριμένη περίπτωση το data έχει τύπο struct slurpoe-mu-msg-endpoint και το func έχει κωδικό MSG-ID-OPEN-ENDPOINT.

- Αποστολή του μηνύματος στη κάρτα, στην οποία θα ανοιχτεί το άκρο. Το μήνυμα, δηλαδή, εγγράφεται στη θέση που υποδεικνύει η δομή MU της συγκεκριμένης κάρτας.

Όταν η κάρτα διαβάσει το μήνυμα και ολοκληρώσει τις διαδικασίες ανοίγματος του άκρου, τότε επιστρέφει στο Dom0 ένα μήνυμα τύπου struct slurpoe-mu-msg-generic, με πεδίο func τον κωδικό MSG-ID-ENDPOINT-READY και data δεδομένα που χαρακτηρίζουν το άκρο αυτό, όπως τον μοναδικό αριθμό του και τη θέση μνήμης της κάρτας στην οποία θα γραφτεί η αντιστοιχία των διευθύνσεων. Έτσι, ενεργοποιείται η συνάρτηση slurpoe-mu-isr, η οποία με βάση το πεδίο func επιλέγει το κατάλληλο μπλοκ εκτέλεσης. Οι λειτουργίες που επιτελούνται τότε είναι οι εξής:

- Κλήση μακροεντολής RING-GET-RESPONSE, που επιστρέφει δείκτη στην επόμενη διαθέσιμη απάντηση.
- Αντιγραφή των πεδίων του μηνύματος στα πεδία της απάντησης.
- Αποστολή απάντησης στο DomU, μέσω της μακροεντολής RING-PUSH-RESPONSES.
- Ενημέρωση του DomU για την καινούρια απάντηση, μέσω του διαύλου και της μακροεντολής HYPERVISOR-event-channel-op.

Αμέσως, ενεργοποιείται η συνάρτηση received-data-from-dom0 του DomU, η οποία αποκτά πρόσβαση στην απάντηση με την κλήση RING-GET-RESPONSE. Με βάση το πεδίο func (MSG-ID-ENDPOINT-READY), επιλέγεται το κατάλληλο μπλοκ εντολών, όπου ενημερώνεται η δομή του άκρου που ανοίχθηκε με τα καινούρια δεδομένα από τη κάρτα και η κατάσταση του άκρου γίνεται SLURPOE-ENDPOINT-STATUS-OK. Η αλλαγή αυτή στη κατάσταση του άκρου οδηγεί στο τερματισμό του βρόχου αναμονής που βρισκόταν το ioctl του DomU και έτσι η διεργασία αυτή τερματίζεται, επιστρέφοντας στο χρήστη τον μοναδικό αριθμό του άκρου που ανοίχθηκε.

4.3 Δέσμευση μνήμης

Αρχικά, ο χρήστης αποκτά ένα χώρο μνήμης στον οποίο αποθηκεύει δεδομένα προς αποστολή ή τον οποίο προετοιμάζει για να δεχτεί τα δεδομένα που θα λάβει. Η δέσμευση του χώρου αυτού γίνεται μέσω της κλήσης συστήματος ioctl, θέτοντας ως εντολή τον κωδικό SLURPOE-IOCTL-REGISTER-USER-REGION και περνώντας σαν παράμετρο την διεύθυνση που αρχίζει ο χώρος αυτός. Η συνάρτηση εξυπηρέτησης βρίσκει το άκρο που αντιστοιχεί σε αυτό το άνοιγμα, θέτει την κατάστασή του ίση με SLURPOE-ENDPOINT-REGISTERING, και διενεργεί ελέγχους ασφάλειας. Έπειτα υπολογίζει τον αριθμό των σελίδων από τις οποίες αποτελείται ο χώρος υπό δέσμευση και καλεί τη συνάρτηση get-user-pages, η οποία δίνει δικαίωμα πρόσβασης των σελίδων αυτών στο domU και ταυτόχρονα τις καθηλώνει. Το αποτέλεσμα της συνάρτησης αυτής είναι μια λίστα από δομές σελίδων (struct page *), οι οποίες δείχνουν στις σελίδες του χώρου αυτού. Προκειμένου να αποκτήσει και το Dom0 πρόσβαση σε αυτές, θα πρέπει να του παραχωρηθούν. Κάθε μία σελίδα που βρίσκεται στη λίστα παραχωρείται στο Dom0 και έτσι δημιουργείται μια αντίστοιχη λίστα που περιέχει του αριθμούς (grefs) των σελίδων αυτών. Πώς όμως θα περάσει η λίστα αυτή στο Dom0, ώστε να τις αποδεχτεί;

Η λύση που βρέθηκε είναι να παραχωρηθεί και η σελίδα στην οποία βρίσκεται η λίστα αυτή. Αφού γίνει αυτό, ο αριθμός σελίδας που επιστρέφεται γράφεται στο πεδίο gref του XenStore, το οποίο είχε αναφερθεί στη πρώτη ενότητα αυτού του κεφαλαίου. Όμως, το Dom0 εκτός από τις σελίδες χρειάζεται και ορισμένα πεδία του άκρου, έτσι ώστε να ολοκληρώσει τη δέσμευση, τα οποία έχουν σχέση με την εγγραφή των αντιστοιχιών διευθύνσεων στη μνήμη της κάρτας. Αυτά τα πεδία θα περάσουν μέσω αιτήματος του δακτυλίου. Δημιουργείται, λοιπόν, ένα αίτημα με πεδίο func τον κωδικό MSG-ID-REGISTER-USER-REGION και πεδίο data μια δομή τύπου struct slurpoe-ring-msg-register. Έπειτα, το ioctl εισέρχεται σε βρόχο αναμονής (busy wait), από τον οποίο θα βγει όταν η κατάσταση του άκρου γίνει SLURPOE-ENDPOINT-REGISTERED.

Όπως γίνεται φανερό από τα παραπάνω, θα ενεργοποιηθεί και η συνάρτηση received-data-from-domU και η συνάρτηση watch-gref του Dom0. Σε αυτό το σημείο πρέπει να ληφθούν υπόψη δύο παράγοντες:

- Η σειρά με την οποία ενεργοποιούνται οι συναρτήσεις είναι τυχαία.
- Η συνάρτηση watch-gref χρειάζεται τα δεδομένα της received-data-from-domU και έτσι πρέπει να την περιμένει να ολοκληρωθεί.

Γι' αυτό το λόγο τα πεδία που λαμβάνονται από τη received-data-from-domU αποθηκεύονται σε έναν προσωρινό αποθηκευτικό χώρο, που μπορεί να έχει πρόσβαση η watch-gref, και η watch-gref αναβάλλει την εκτέλεσή της μέχρι να γεμίσει αυτός ο χώρος. Μόλις η watch-gref αποκτήσει τα επιθυμητά δεδομένα, ξεκινάει η διαδικασία αποδοχής των σελίδων. Αρχικά, αποδέχεται τη σελίδα που υποδεικνύει το πεδίο gref και η οποία περιέχει τη λίστα με τα grefs των σελίδων του χώρου υπό δέσμευση. Στη συνέχεια, για κάθε αριθμό που διαβάζει από την αρχική σελίδα, εκτελεί τις εξής ενέργειες:

- Δεσμεύει μια εικονική σελίδα με τη συνάρτηση alloc-vm-area.
- Αρχικοποιεί τη μεταβλητή ops που είναι τύπου struct gnttab-map-grant-ref, με τη συνάρτηση gnttab-set-map-op. Όμως, αυτή τη φορά χρησιμοποιείται η σημαία GNTMAP-device-map, αντί της GNTMAP-host-map, αφού σε αυτή την περίπτωση το Dom0 δεν επιθυμεί να προσπελάσει τα δεδομένα της κοινής σελίδας, αλλά θέλει να τη χρησιμοποιήσει για λειτουργίες εισόδου/εξόδου. Όταν ολοκληρωθεί η κλήση αυτή, το πεδίο dev-bus-addr της ops περιέχει τη φυσική διεύθυνση της σελίδας.
- Η φυσική διεύθυνση που προκύπτει από το dev-bus-addr και η αντίστοιχη εικονική διεύθυνση εγγράφονται στη θέση μνήμης της κάρτας, που υποδεικνύεται από το αίτημα του DomU. Η αντίστοιχη εικονική διεύθυνση προκύπτει αν σε κάθε επανάληψη προστίθεται στην προηγούμενη εικονική διεύθυνση το μέγεθος της σελίδας, ξεκινώντας από την εικονική διεύθυνση που μετέφερε το αίτημα και που έδειχνε στην αρχή του χώρου. Ο υπολογισμός αυτός βασίζεται στη συνέχεια της εικονικής μνήμης και στο γεγονός ότι μια ολόκληρη φυσική σελίδα αντιστοιχίζεται σε μια ολόκληρη εικονική σελίδα.

Όταν επεξεργαστούν όλες οι σελίδες, το DomU στέλνει στο Dom0 μία απάντηση επιτυχίας της διαδικασίας, με πεδίο func το MSG-ID-REGISTE-USER-REGION. Έτσι ενεργοποιείται η συνάρτηση received-data-from-dom0 του DomU, η οποία απλώς θέτει την κατάσταση του άκρου σε SLURPOE-ENDPOINT -REGISTERED. Μόλις συμβεί αυτό, το ioctl εξέρχεται από το βρόχο αναμονής και επιστρέφει στο χώρο χρήστη έναν κωδικό επιτυχίας.

4.4 Εγγραφή δεδομένων

Έχοντας δεσμεύσει το χώρο στον οποίο υπάρχουν τα δεδομένα προς αποστολή,ο χρήστης αρκεί να κατασκευάσει το κατάλληλο μήνυμα για την κάρτα και να το περάσει ως παράμετρο στην ioctl, με εντολή την SLURPOE-IOCTL-RDMA-WRITE, έτσι ώστε να εκκινηθεί ο μηχανισμός DMA και να μεταφέρει τα δεδομένα. Πιο συγκεκριμένα, ο χρήστης δημιουργεί ένα μήνυμα τύπου struct slurpoe-mu-msg-generic-user, με πεδίο func τον κωδικό MSG-ID-RDMA-WRITE και πεδίο data μια δομή τύπου struct slurpoe-ring-msg-rdma-write. Στα πεδία αυτά αναγράφεται η εικονική διεύθυνση από την οποία ξεκινά ο χώρος, η εικονική διεύθυνση του χώρου που θα υποδεχτεί τα δεδομένα, το μέγεθος των δεδομένων, ο αριθμός του παρόντος άκρου και ο αριθμός του άκρου στο οποίο προορίζονται τα δεδομένα. Μετά την δημιουργία του μηνύματος καλείται η ioctl, η οποία αντιγράφει το μήνυμα σε ένα αίτημα του δακτυλίου και το αποστέλλει στο Dom0.

Το Dom0 λαμβάνει το αίτημα, μέσω της συνάρτησης received-data-from - domU, τα αντιγράφει σε μήνυμα τύπου struct slurpoe-mu-msg-generic και το στέλνει στην κάρτα. Η κάρτα διαβάζει το μήνυμα και εκκινεί την μεταφορά δεδομένων από τον χώρο χρήστη σε αυτή. Αφού λάβει τα δεδομένα, τα στέλνει στο δίκτυο. Από την άλλη πλευρά, η κάρτα προορισμού λαμβάνει τα δεδομένα, ανιχνεύει τον αριθμό του άκρου στο οποίο προορίζονται και την εικονική διεύθυνση στην οποία θα τοποθετηθούν, βρίσκει την αντίστοιχη φυσική διεύθυνση και εκκινεί τη μεταφορά δεδομένων από τη κάρτα στο χώρο χρήστη, όπου είχε δεσμευτεί το αντίστοιχο μέγεθος μνήμης.

4.5 Τερματισμός

Η διαδικασία του τερματισμού λαμβάνει χώρα όταν ο χρήστης κλείσει το αρχείο που αντιστοιχεί στη συσκευή slurpoe-xen. Η συνάρτηση εξυπηρέτησης του τερματισμού (close) στέλνει απλώς ένα αίτημα στο Dom0 με πεδίο func τον κωδικό MSG-ID-CLOSE-ENDPOINT και πεδίο data τον μοναδικό αριθμό του άκρου που πρέπει να κλείσει. Όταν το Dom0 λάβει το αίτημα, το προωθεί στην κάρτα δικτύου και εκείνη αποδεσμεύει τις δομές που ανήκαν στο άκρο αυτό.

Κεφάλαιο 5

Αξιολόγηση

Σε αυτό το κεφάλαιο παρουσιάζονται οι μετρήσεις που ελήφθησαν κατά την εκτέλεση του πρωτοκόλλου SLURPoE στο εικονικό περιβάλλον του Xen και συγκρίνονται με αυτές που προέκυψαν από την εκτέλεσή του σε μη εικονικό περιβάλλον. Οι μετρήσεις αυτές αναφέρονται στο ρυθμό μετάδοσης των δεδομένων (bandwidth) και στο χρόνο απόκρισης (latency), δηλαδή στο χρόνο που χρειάζεται το πρώτο byte δεδομένων να φτάσει στον προορισμό του. Αρχικά, λοιπόν, εξηγείται ο τρόπος και το πρόγραμμα που χρησιμοποιήθηκε για να ληφθούν οι μετρήσεις και έπειτα παρουσιάζονται τα διαγράμματα που δημιουργήθηκαν, τα οποία συνοδεύονται από τον κατάλληλο σχολιασμό.

5.1 Μεθοδολογία αξιολόγησης

Για την αξιολόγηση του συστήματος αναπτύχθηκε μια εφαρμογή που παίρνει ως παραμέτρους τον αριθμό των μηνυμάτων που θα σταλούν και το μέγεθος που θα έχει το κάθε ένα σε bytes, δημιουργεί δύο νήματα κατά την εκτέλεση του και εκτελείται σε εικονικό αλλά και σε μη εικονικό περιβάλλον, έτσι ώστε να συγκριθούν τα αντίστοιχα αποτελέσματα. Η περιγραφή της εκτέλεσης έχει ως εξής:

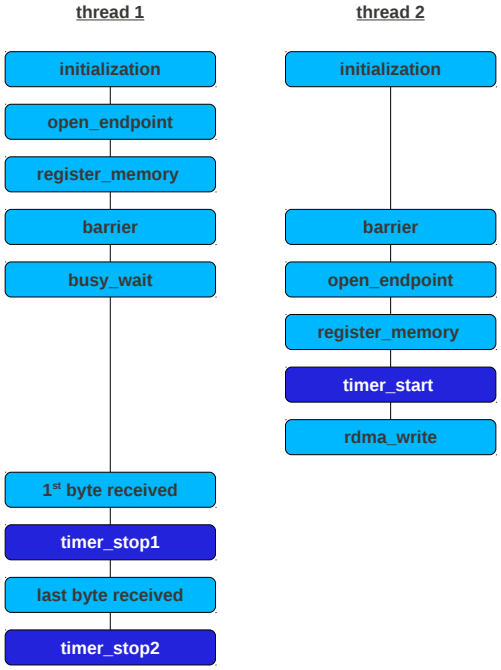
Το πρώτο νήμα δέχεται δεδομένα και ακολουθεί τα παρακάτω βήματα:

- Ανοίγει ένα άκρο στη κάρτα με αύξων αριθμό 0.
- Δεσμεύει τόσο χώρο μνήμης όσο είναι το μέγεθος των δεδομένων που πρόκειται να λάβει, κάτι που υποδεικνύεται από τις παραμέτρους.
- Εισέρχεται σε κατάσταση αναμονής (busy wait) έως ότου λάβει το πρώτο και το τελευταίο byte των δεδομένων.

Από την άλλη πλευρά, το δεύτερο νήμα στέλνει τα δεδομένα στο πρώτο νήμα, ακολουθώντας τα εξής βήματα:

- Ανοίγει ένα άκρο στη κάρτα με αύξων αριθμό 1.
- Δεσμεύει τόσο χώρο μνήμης όσο είναι το μέγεθος των δεδομένων που στέλνει, κάτι που υποδεικνύεται από τις παραμέτρους.
- Εκκινεί τον μηχανισμό μεταφοράς των δεδομένων.

Το Σχήμα 5.1 παρουσιάζει τη ροή του περιγράφηκε και τα σημεία που λαμβάνονται οι μετρήσεις για να υπολογιστεί ο ρυθμός μετάδοσης και ο χρόνος απόκρισης.



Σχήμα 5.1: Πρόγραμμα δοκιμής συστήματος

Συνεπώς ο ρυθμός μετάδοσης των δεδομένων είναι ίση με (μέγεθος μηνύματος σε Bytes)/(stopTime2-startTime) και ο χρόνος απόκρισης είναι ίσος με (stopTime1 - startTime).

Οι λόγοι για τους οποίους χρησιμοποιήθηκαν δύο νήματα και όχι δύο διαφορετικά προγράμματα σε δύο διαφορετικά DomUs αναφέρονται παρακάτω:

- Ο αποστολέας των δεδομένων πρέπει με κάποιο τρόπο να γνωρίζει την εικονική διεύθυνση του χώρου που έχει δεσμεύσει ο DomU για να δεχτεί τα δεδομένα. Επειδή δεν έχει υλοποιηθεί ακόμα ο μηχανισμός ενημέρωσης του αποστολέα μέσω μηνύματος, ο μόνος τρόπος που μπορεί

να γνωστοποιηθεί η διεύθυνση αυτή είναι μέσω κοινής μνήμης, δηλαδή η χρήση νημάτων.

- Η λήψη των μετρήσεων απαιτεί τη χρήση κοινού ρολογιού μεταξύ του αποστολέα και του παραλήπτη. Για τον λόγο αυτό χρησιμοποιείται πάλι κοινή μνήμη μεταξύ νημάτων.

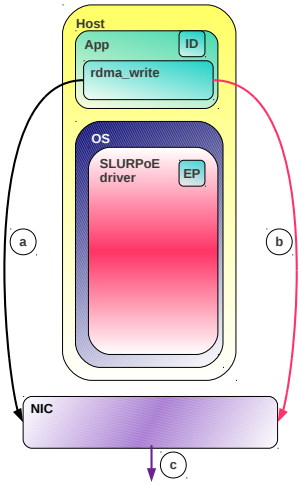
- Δεν έχει υλοποιηθεί ακόμα η πολύπλεξη του πρωτοκόλλου SLURPoE στο Dom0 και έτσι μέχρι στιγμής δεν μπορούν να υποστηριχθούν πάνω από ένα DomU.

Πάντως, η επιλογή αυτή δεν επηρεάζει τις μετρήσεις που συγκεντρώθηκαν, καθώς αυτές αναφέρονται μόνο στο στάδιο της μεταφοράς δεδομένων, ενώ η διαδικασία ενημέρωσης του αποστολέα με τη κατάλληλη εικονική διεύθυνση και η πολύπλεξη στο Dom0 θα επηρέαζε τα προηγούμενα στάδια. Επίσης, από τη στιγμή που τα νήματα απευθύνονται σε διαφορετικές κάρτες ακολουθείται το ίδιο μονοπάτι δικτύου που θα ακολουθούνταν και στη γενικότερη περίπτωση (διαφορετικές εφαρμογές σε διαφορετικά DomUs).

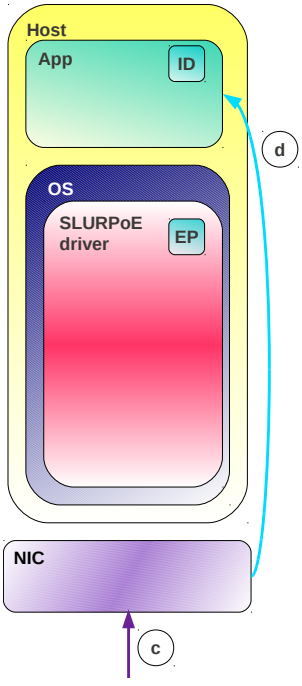
5.2 Αποτελέσματα αξιολόγησης

Σε αυτό το σημείο παρουσιάζονται τα αποτελέσματα που συγκεντρώθηκαν, υπό τη μορφή διαγραμμάτων. Το πρώτο διάγραμμα (Σχήμα 5.6) αναπαριστά τη μεταβολή του ρυθμού μετάδοσης δεδομένων συναρτήσει του μεγέθους του μηνύματος που μεταδίδεται, ενώ το δεύτερο διάγραμμα (Σχήμα 5.7) αναπαριστά τη μεταβολή του χρόνου απόκρισης συναρτήσει του μεγέθους του μηνύματος που μεταδίδεται. Γιατί, όμως, επιλέχθηκε να παρουσιαστούν αυτά τα δύο μεγέθη; Η γνώση της συμπεριφοράς του ρυθμού μετάδοσης των δεδομένων είναι χρήσιμη σε περίπτωση που απαιτείται η μεταφορά μεγάλου όγκου δεδομένων μεταξύ των κόμβων μιας συστοιχίας υπολογιστών (πραγματικών ή εικονικών). Αυτό μπορεί να συμβεί, για παράδειγμα, όταν ένας κόμβος-βοηθός επιθυμεί να στείλει τα δεδομένα που υπολόγισε (πίνακας) στον κόμβο-αρχηγό, έτσι ώστε αυτός να τα συνδέσει με τα υπόλοιπα που συνέλεξε. Από την άλλη πλευρά, η γνώση της συμπεριφοράς του χρόνου απόκρισης είναι χρήσιμη σε περίπτωση που ανταλλάσσονται μεταξύ των κόμβων μηνύματα μικρού μεγέθους, καθώς σε αυτή τη περίπτωση ο επιπλέον χρόνος μετάδοσης ολόκληρου του μηνύματος είναι αμελητέος σε σχέση με τη μετάδοση του πρώτου byte. Αυτό μπορεί να συμβεί κατά την ανταλλαγή συνοριακών τιμών των χώρων υπολογισμού μεταξύ των κόμβων-βοηθών, που συνήθως καταλαμβάνουν μικρό χώρο.

Προκειμένου να σχολιαστούν τα διαγράμματα, παραθέτονται τέσσερα σχήματα που περιγράφουν τα μονοπάτια που ακολουθούνται κατά τη διάρκεια του σταδίου rdma-write του πρωτοκόλλου SLURPoE, όταν αυτό εκτελείται σε εικονικό και σε μη εικονικό περιβάλλον.

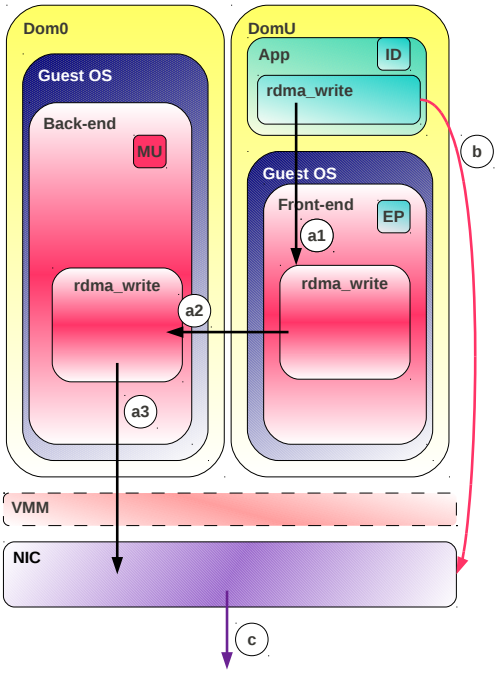


Σχήμα 5.2: rdma-write σε μη εικονικό περιβάλλον

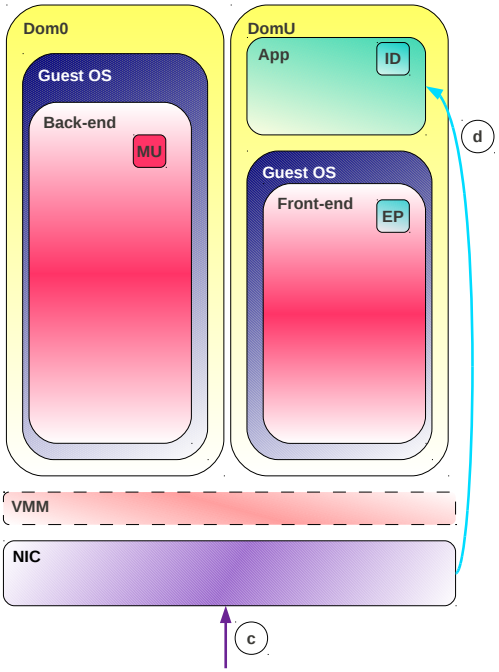


Σχήμα 5.3: Λήψη δεδομένων από rdma-write σε μη εικονικό περιβάλλον

Όπως έχει εξηγηθεί στα προηγούμενα κεφάλαια το στάδιο του rdma-write αποτελείται από δύο βήματα. Το πρώτο (a) είναι η αποστολή του μηνύματος



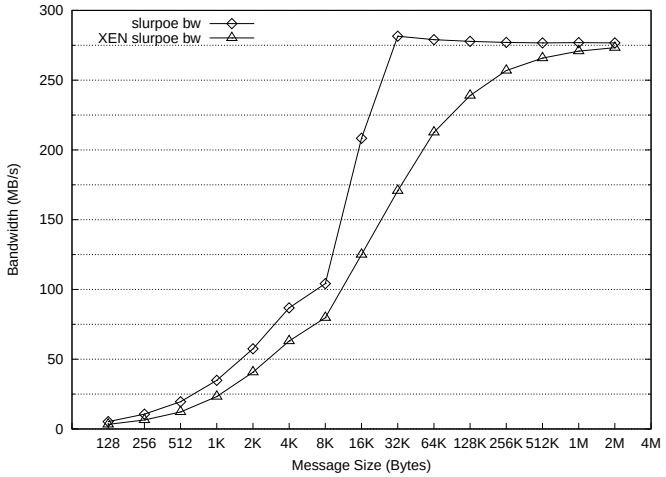
Σχήμα 5.4: rdma-write σε εικονικό περιβάλλον



Σχήμα 5.5: Λήψη δεδομένων από rdma-write σε εικονικό περιβάλλον

rdma-write από τον χώρο χρήστη στην κάρτα, για να γνωστοποιηθεί στη κάρ-

τα ο χώρος από τον οποίο θα λάβει τα δεδομένα με DMA, και το δεύτερο (b) είναι η μεταφορά των δεδομένων από τον χώρο χρήστη στη κάρτα. Όταν το SLURPoE εκτελείται σε μη εικονικό περιβάλλον, τότε η αποστολή του μηνύματος είναι άμεση από τον χώρο χρήστη στη κάρτα, χωρίς να παρεμβάλλεται το λειτουργικό σύστημα. Αυτό γίνεται διότι στις εικονικές διευθύνσεις της διεργασίας του χρήστη έχουν αντιστοιχηθεί οι φυσικές διευθύνσεις του χώρου της κάρτας, στον οποίο αυτή δέχεται τα μηνύματα ελέγχου (ioremap,mmap), και έτσι μπορεί να τις προσπελαύνει άμεσα. Επειδή σε εικονικό περιβάλλον τα DomUs δεν έχουν πρόσβαση στο υλισμικό, αυτή η αντιστοίχιση διευθύνσεων δε μπορούσε να γίνει εύκολα σε πρώτο στάδιο. Οπότε το μήνυμα θα πρέπει να περάσει από το Dom0 για να φτάσει στη κάρτα. Και στις δύο περιπτώσεις όμως το μονοπάτι της λήψης δεδομένων είναι το ίδιο καθώς η κάρτα μεταφέρει απλώς τα δεδομένα με DMA από το χώρο της στο χώρο χρήστη.

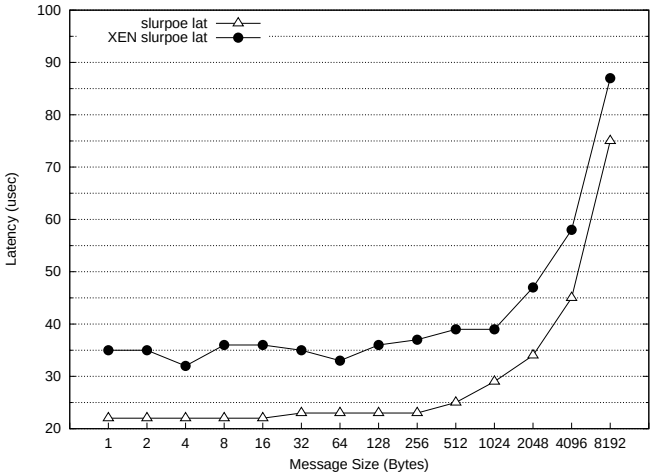


Σχήμα 5.6: Ρυθμός μετάδοσης συναρτήσει του μεγέθους μηνύματος

Από το Σχήμα 5.6 προκύπτουν τα εξής συμπεράσματα:

- Οι ταχύτητες μετάδοσης του Xen-SLURPoE είναι μικρότερες από αυτές του SLURPoE. Το ιδανικό θα ήταν οι ταχύτητες των δύο περιπτώσεων να συνέπιπταν και η διαδικασία της εικονικοποίησης να μην επηρέαζε την απόδοση, όμως όπως φαίνεται αυτό δε συμβαίνει στη πραγματικότητα. Η εξήγηση που μπορεί να δοθεί για αυτή την απόκλιση είναι ότι οφείλεται στο πέρασμα του μηνύματος από το DomU και το Dom0, αντί να αποστέλλεται άμεσα στη κάρτα.

- Οι δύο γραφικές παραστάσεις παρουσιάζουν σχεδόν όμοια συμπεριφορά, συγκλίνοντας στις οριακές τιμές. Η συμπεριφορά αυτή εξηγείται από το ότι στη μετάδοση μηνυμάτων μεγάλου μεγέθους κυριαρχούν σε



Σχήμα 5.7: Χρόνος απόκρισης συναρτήσει του μεγέθους μηνύματος

θέμα χρόνου τα μονοπάτια (b), (c) και (d). Δηλαδή, κυριαρχεί η μεταφορά των δεδομένων με τους μηχανισμούς DMA και η διάδοσή τους στο φυσικό μέσο. Επειδή τα μονοπάτια αυτά είναι ίδια και για τις δύο περιπτώσεις, προσδίδουν την ίδια συμπεριφορά στο ρυθμό μετάδοσης.

- Όσο αυξάνεται το μέγεθος του μηνύματος τόσο μειώνεται η επίδραση του χρόνου μεταφοράς του μηνύματος rdma-write, μέσω του Dom0 στη κάρτα, στον συνολικό χρόνο μετάδοσης των δεδομένων στον τελικό προορισμό. Αυτό εξηγεί το ότι όσο αυξάνεται το μέγεθος του μηνύματος, τόσο η διαφορά μεταξύ των αντίστοιχων ταχυτήτων μειώνεται.

Τέλος, τα συμπεράσματα που προκύπτουν από το Σχήμα 5.7 είναι τα εξής:

- Οι χρόνοι απόκρισης του Xen-SLURPoE είναι μεγαλύτεροι από αυτούς του SLURPoE. Όπως και παραπάνω, το ιδανικό θα ήταν οι τιμές αυτές να συνέπιπταν, όμως λόγω του περάσματος του μηνύματος rdma-write από το Dom0, αυτό δε γίνεται. Ο ισχυρισμός αυτός μπορεί να επαληθευτεί από το ότι η διαφορά στους χρόνους είναι σχεδόν σταθερή και περίπου ίση με 12us.

- Η συμπεριφορά των δύο γραφικών παραστάσεων είναι ίδια. Αυτό οφείλεται στον ίδιο λόγο που εξηγήθηκε παραπάνω. Από το διάγραμμα επαληθεύεται η σταθερή επιβάρυνση που προσδίδει το πέρασμα του μηνύματος στη κάρτα μέσω του Dom0.

Κεφάλαιο 6

Επίλογος

6.1 Σύνοψη

Σκοπός της παρούσας διπλωματικής εργασίας ήταν ο σχεδιασμός και η υλοποίηση της ένταξης ενός δικτύου διασύνδεσης υψηλής επίδοσης σε εικονικό περιβάλλον. Ως δίκτυο διασύνδεσης υψηλής επίδοσης χρησιμοποιήθηκε το SLURPoE, που αναπτύχθηκε από το εργαστήριο υπολογιστικών συστημάτων, κατά τη διάρκεια της διπλωματικής εργασίας "Σχεδίαση και Υλοποίηση μηχανισμού απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10GbE". Το εικονικό περιβάλλον δημιουργήθηκε από τον ελεγκτή εικονικής μηχανής Xen. Η ένταξη αυτή επιτεύχθηκε μέσω της δημιουργίας ενός διαχωρισμένου οδηγού (split driver), του οποίου το κάτω μισό τοποθετήθηκε στην εικονική μηχανή υπηρεσίας (Dom0) και το πάνω μισό σε μία φιλοξενούμενη εικονική μηχανή (DomU). Ο διαχωρισμένος οδηγός συνδυάζει τις λειτουργίες που προσφέρει το SLURPoE με τους μηχανισμούς εικονικοποίησης που παρέχει ο Xen, έτσι ώστε να επιτευχθεί η εικονικοποίηση του SLURPoE (Virtualization of SLURPoE).

Αρχικά παρουσιάστηκε στον αναγνώστη το θεωρητικό υπόβαθρο που θα έπρεπε να κατανοήσει έτσι ώστε να ακολουθήσει τη διαδικασία σχεδιασμού και υλοποίησης του συστήματος. Παρουσιάστηκε και αναλύθηκε η έννοια της εικονικής μηχανής και αναφέρθηκαν εργαλεία του λειτουργικού συστήματος Linux που χρησιμοποιήθηκαν στην υλοποίηση του συστήματος. Επίσης, εξηγήθηκε η έννοια της επικοινωνίας σε επίπεδο χρήστη και παρατέθηκαν τρόποι με τους οποίους βελτιώνεται η απόδοση της επικοινωνίας των εικονικών μηχανών.

Στη συνέχεια, εξηγήθηκε αναλυτικά ο σχεδιασμός του συστήματος. Δηλαδή, εξηγήθηκε ο τρόπος με τον οποίο οι λειτουργίες που παρέχει το SLURPoE

προσαρμόστηκαν στο εικονικό περιβάλλον έτσι ώστε η διεπαφή του προς τον χρήστη και τη κάρτα να μην αλλάξει.

Μετά το σχεδιασμό, ακολούθησε η παρουσίαση της υλοποίησης του συστήματος. Αναλύθηκε, λοιπόν, με λεπτομέρεια, πώς υλοποιήθηκαν οι λειτουργίες του SLURPoE στο εικονικό περιβάλλον ξεκινώντας από το χώρο χρήστη και φθάνοντας στη κάρτα. Ταυτόχρονα με την ανάλυση έγιναν φανερά τα εργαλεία και οι μηχανισμοί που παρέχει ο ελεγκτής Xen για την ενδοεπικοινωνία των εικονικών μηχανών, καθώς και οι μηχανισμοί που προσφέρει το Linux έτσι ώστε να γίνεται εφικτή η άμεση προσπέλαση του υλισμικού από τον χώρο χρήστη.

Τέλος, παρουσιάστηκε η αξιολόγηση του συστήματος. Δόθηκαν δύο γραφήματα που αναπαριστούσαν τη συμπεριφορά της ταχύτητας μετάδοσης και του χρόνου απόκρισης του πρωτοκόλλου SLURPoE όταν αυτό εκτελείται σε εικονικό και μη περιβάλλον. Αναλύθηκε σε βάθος το μονοπάτι που ακολουθεί η λειτουργία εγγραφής (rdma-write) και σχολιάστηκαν οι αποκλίσεις που προέκυψαν στις μετρήσεις.

6.2 Συμπεράσματα

Επιτεύχθηκε ο σχεδιαστικός στόχος που τέθηκε στο Κεφάλαιο 3; Η απάντηση είναι ναι, αφού τα δεδομένα μεταφέρονται απευθείας στη κάρτα δικτύου, χωρίς να δημιουργούνται αντίγραφα και χωρίς να παρεμβάλλεται το εκάστοτε λειτουργικό σύστημα και ο ελεγκτής εικονική μηχανής. Όμως, όπως από κάθε ταξίδι έτσι και από το ταξίδι της δημιουργίας της παρούσας διπλωματικής εργασίας προέκυψαν χρήσιμα διδάγματα και συμπεράσματα.

Η διαδικασία της εικονικοποίησης παρήγαγε ένα από αυτά. Γενικά, η δημιουργία ενός διαχωρισμένου οδηγού απαιτεί την κατανόηση της χρησιμότητάς του και των μηχανισμών που παρέχονται από τον ελεγκτή Xen, έτσι ώστε να επικοινωνούν μεταξύ τους τα δύο μισά του οδηγού. Η κατάλληλη χρησιμοποίηση των μηχανισμών αυτών (grant pages, event channels) μπορεί να βελτιώσει κατά πολύ την απόδοση και να απαλλάξει τον προγραμματιστή από την συγγραφή περιττού κώδικα. Από την άλλη πλευρά, απαιτείται πολύ καλή κατανόηση του οδηγού που πρόκειται να εικονικοποιηθεί, αφού θα πρέπει να διαχωριστούν οι λειτουργίες του μεταξύ του DomU και του Dom0. Η γενική γραμμή που ακολουθείται είναι: οι λειτουργίες που εξαρτώνται άμεσα από το υλισμικό και οι δομές που είναι μοναδικές για το σύστημα τοποθετούνται στο Dom0.

Σημαντικός παράγοντας της επιτυχίας της εικονικοποίησης είναι η διατήρηση της διεπαφής προς τον χρήστη και το υλισμικό. Αυτός ο παράγοντας υπάρχει στο σύστημα που υλοποιήθηκε, καθώς ο χρήστης και η κάρτα δεν έχουν επί-

γνωση ότι το πρωτόκολλο εκτελείται σε εικονικό περιβάλλον. Το υλικολογισμικό που βρίσκεται στη κάρτα διατηρήθηκε ακριβώς ίδιο με αυτό που χρησιμοποιούνταν στο μη εικονικό περιβάλλον. Η βιβλιοθήκη του χρήστη διατηρήθηκε κι αυτή αμετάβλητη, αφού ο χρήστης διενεργεί τόσες λειτουργίες όσες διενεργούσε και στο μη εικονικό περιβάλλον (open endpoint, register memory, rdma write).

Με τη βοήθεια του μηχανισμού DMA και των μηχανισμών επικοινωνίας μεταξύ των Doms που παρέχει ο Xen, και ειδικότερα του μηχανισμό παραχώρησης σελίδων, επιτεύχθηκε η απευθείας μεταφορά δεδομένων από το χώρο χρήστη στη κάρτα και αντιστρόφως. Επίσης, επιτεύχθηκε η απεμπλοκή του λειτουργικού συστήματος και του ελεγκτή εικονικής μηχανής κατά τη μεταφορά δεδομένων, αφού συμμετέχουν κατά τις λειτουργίες του ανοίγματος άκρου και της δέσμευσης μνήμης έτσι ώστε να διασφαλίσουν την απομονωμένη και νόμιμη μετακίνηση δεδομένων μεταξύ των χώρων.

Από τη άλλη πλευρά, η αξιολόγηση έδειξε ότι οι μετρήσεις που συγκεντρώθηκαν αποκλίνουν από τις ιδανικές, κάτι που οφείλεται στη μεταφορά του μηνύματος rdma-write μέσα από το DomU και το Dom0. Αυτό οφείλεται στο ότι ο χώρος χρήστη, που βρίσκεται στο DomU, δεν έχει επίγνωση του υλισμικού και έτσι δεν μπορεί να το προσπελάσει άμεσα, κάτι που γίνεται σε μη εικονικό περιβάλλον.

Τέλος, δεν υποστηρίζεται πολύπλεξη των DomUs στο Dom0, δηλαδή δεν μπορούν δύο οι περισσότερες εικονικές μηχανές να χρησιμοποιήσουν ταυτόχρονα τον διαχωρισμένο οδηγό του SLURPoE. Οι δύο τελευταίες παραλήψεις δεν οφείλονται στη πολυπλοκότητα ή στη δυσκολία της διαδικασίας, αλλά στο ότι επιλέχθηκε ο δεδομένος χρόνος της διπλωματικής να επενδυθεί στην ολοκλήρωση του συστήματος σε σχέση με τις λειτουργίες του SLURPoE. Δηλαδή, στόχος της διπλωματικής εργασίας ήταν η απόκτηση γνώσης για τη δημιουργία διαχωρισμένων οδηγών, για την επικοινωνία μεταξύ των δύο μισών του οδηγού, για τα δίκτυα διασύνδεσης υψηλής επίδοσης και για την προσαρμογή τους στο εικονικό περιβάλλον. Προτιμήθηκε να αποκτηθεί γνώση για όσο το δυνατόν περισσότερα πεδία, παρά να επιλυθούν συγκεκριμένα τεχνικά θέματα.

6.3 Μελλοντικές επεκτάσεις

Η παρούσα εργασία πραγματεύεται τη σχεδίαση και υλοποίηση ενός μηχανισμού ένταξης των δικτύων υψηλής επίδοσης που βασίζονται στο χώρο χρήστη σε εικονικά περιβάλλοντα. Υλοποιείται ένα μονοπάτι δεδομένων που δεν περιλαμβάνει τον ελεγκτή εικονικών μηχανών, ή την εικονική μηχανή υπηρεσίας, στρώματα δηλαδή που επηρεάζουν σημαντικά τη ρυθμαπόδοση του

συστήματος. Στόχος είναι η απόδειξη πως ένα τέτοιο μονομάτι είναι εφικτό και πως η επιβάρυνση που δημιουργείται λόγω των ενδιάμεσων στρωμάτων που αναγκαστικά παρεμβάλλονται δεν είναι σημαντική και δεν περιλαμβάνεται στο κρίσιμο μονοπάτι. Σε καμιά περίπτωση η υλοποίηση στο πλαίσιο της εργασίας δεν μπορεί να αποτελέσει μια εφ' όλης της ύλης προσέγγιση στο ζήτημα. Άλλωστε και η συγκεκριμένη υλοποίηση βασίστηκε σε ένα πρωτόκολλο που δεν περιλαμβάνει όλα τα στρώματα που χρειάζονται οι επιστημονικές εφαρμογές.

Όπως φαίνεται από τα παραπάνω, ο σχεδιασμός και η υλοποίηση του συστήματος αυτού επιδέχονται σημαντικές βελτιώσεις και επεκτάσεις που θα καθορίσουν σημαντικά την απόδοσή του. Μερικές από αυτές αναφέρονται παρακάτω:

- Πολύπλεξη: Η υλοποίηση ενός βέλτιστου μονοπατιού αρκεί για να αποδείξουμε ότι σε εικονικά περιβάλλοντα, τα δεδομένα μπορούν να κινούνται με απομονωμένο και ασφαλή τρόπο από το χώρο χρήστη στο δίκτυο χωρίς τη μεσολάβηση ενδιάμεσων στρωμάτων που προσδίδουν σημαντική επιβάρυνση. Η πολύπλεξη των σημείων πρόσβασης στο δίκτυο αποτελεί βασική επέκταση της εργασίας, αφού θα επιτρέψει πολλές εικονικές μηχανές να μοιράζονται το υλικό. Μια βασική σχεδιαστική επιλογή είναι αν η πολύπλεξη θα γίνεται στην εικονική μηχανή υπηρεσίας ή στη συσκευή. Στην πρώτη περίπτωση, το λειτουργικό σύστημα υπηρεσίας θα είναι υπεύθυνο να αρχικοποιεί τις αντιστοιχίσεις προς θέσεις μνήμης των εικονικών μηχανών και να εγκαθιστά την επικοινωνία με το υλικό. Το μονοπάτι ελέγχου δηλαδή θα διασχίζει το λειτουργικό σύστημα υπηρεσίας για να καταλήξει στο υλικό. Στη δεύτερη περίπτωση, η πολύπλεξη θα γίνεται στο υλικό: με την αρχικοποίηση της συσκευής, το υλικολογισμικό, που εκτελείται σ' αυτή, θα εξαγάγει μέσω του διαύλου καταχωρητές που το λειτουργικό σύστημα υπηρεσίας θα παραχωρεί στις εικονικές μηχανές. Έτσι, μετά την εγκατάσταση της επικοινωνίας, οι εικονικές μηχανές θα μπορούν να έχουν πρόσβαση στο υλικό, χωρίς να παρεμβάλλεται ο ελεγκτής ή το λειτουργικό σύστημα υπηρεσίας είτε στο μονοπάτι ελέγχου είτε στο μονοπάτι δεδομένων (χρήση τεχνικών I/O Virtualization).

- Επικοινωνία με το υλικό: Η επικοινωνία του πάνω-μισού του οδηγού με το αντίστοιχο κάτω-μισό και με το υλικό, στη συγκεκριμένη υλοποίηση γίνεται με χρήση ενός καναλιού γεγονότων (event channel) που εγκαθίσταται στην αρχικοποίηση καθώς και με μηχανισμούς παραχωρήσεων (grant mechanisms). Η βελτιστοποίηση της επικοινωνίας χρησιμοποιώντας μια μηχανή καταστάσεων και θέσεις μοιραζόμενης μνήμης

θα μπορούσε να αυξήσει την απόδοση του συστήματος, αφού στην ενδιάμεση επικοινωνία (μετά την αρχικοποίηση) δε θα χρειάζεται να χρησιμοποιείται κανένας από τους παραπάνω μηχανισμούς.

- Επεκτάσεις του πρωτοκόλλου: Το πιο σημαντικό μειονέκτημα του δικτύου διασύνδεσης που βασίστηκε η παρούσα εργασία είναι η μειωμένη του απόδοση (τόσο για χρόνους απόκρισης, όσο και για εύρος ζώνης). Αυτό εικάζουμε ότι εν μέρει οφείλεται στο υλικό και αποτελεί αντικείμενο διερεύνησης εργασίας που υλοποιείται στο εργαστήριο Υπολογιστικών Συστημάτων. Παρόλα αυτά, ο σχεδιασμός του πρωτοκόλλου επιδέχεται συγκεκριμένες επεκτάσεις ανεξάρτητες από το υλικό. Η υποστήριξη μιας βιβλιοθήκης επικοινωνίας (πχ MPI) πάνω από το SLURPoE αποτελεί μια τέτοια επέκταση του συστήματος. Με αυτόν τον τρόπο, θα μπορούσαμε να εκτελέσουμε πραγματικές επιστημονικές εφαρμογές πάνω από το σύστημά μας δοκιμάζοντας τόσο τα όρια, όσο και τις απαιτήσεις που δημιουργούνται όσον αφορά στο δίκτυο διασύνδεσης όταν μεγάλης κλίμακας εφαρμογές εκτελούνται σε εικονικά περιβάλλοντα. Ωστόσο, ο σχεδιασμός και η υλοποίηση αυτής της επέκτασης είναι μια χρονοβόρα και πολύπλοκη διαδικασία που ενδεχομένως να μπορεί να διαχωριστεί σε επιμέρους αυτόνομα βήματα: (α) Η αναγνώριση σημείων πρόσβασης σε συσκευές στο δίκτυο δεν έχει υλοποιηθεί. Αυτό σημαίνει πως δεν υπάρχει μηχανισμός που να μπορεί να ενημερώσει έναν κόμβο για τους γειτονικούς (ή μη). (β) Αφού το Ethernet, στο φυσικό επίπεδο, δεν υποστηρίζει έλεγχο ροής, θα πρέπει να υλοποιηθεί ένας μηχανισμός διόρθωσης και ελέγχου σφαλμάτων. Το ιδανικό μέρος να υφίσταται αυτός ο μηχανισμός είναι πάνω στο υλικό, οπότε θα πρέπει να τροποιηθεί το λογισμικό συσκευής. (γ) Ενδο-επικοινωνία: θα μπορούσαμε να επεκτείνουμε σημασιολογικά την προγραμματιστική διεπαφή του πρωτοκόλλου έτσι ώστε να υποστηρίζει μηχανισμούς μοιραζόμενης μνήμης όταν πρόκειται για επικοινωνία μέσα στο ίδιο φυσικό σύστημα. Έτσι, χωρίς η εφαρμογή να είναι ενήμερη, μπορεί το πρωτόκολλο να ανιχνεύει τους ”γείτονες” και να εγκαθιστά την επικοινωνία είτε με μοιραζόμενη μνήμη (αν πρόκειται για εικονικές μηχανές στο ίδιο φυσικό μηχάνημα), είτε με το υλικό.

Βιβλιογραφία

[1] Wikipedia, "Virtual," May 2010.

[2] Wikipedia, "Virtual machine," May 2010.

[3] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," *ACM*, vol. 17, pp. 412–421, 1974.

[4] C. D. Encyclopedia, "virtual machine monitor," 2010.

[5] C. D. Encyclopedia, "paravirualization," 2010.

[6] T. Hirt, "Kvm - the kernel-based virtual machine," Feb. 2010.

[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Prat, and A. Warfield, "Xen and the art of virtualization," *SOSP*, Oct. 2003.

[8] K.J.Duda and D.R.Cheriton, "Borrowed-virtual-time (bvt) scheduling: supporting latency-sensitive threads in a general-purpose scheduler," *ACM*, vol. 33, pp. 261–276, 1999.

[9] R. Bhoedjang, T. Ruhl, and H. Bal, "User-level network interface protocols," *IEEE Computer*, pp. 53–60, Nov. 1998.

[10] A. Menon, A. L.Cox, and W. Zwaenepoel, "Optimizing network virtualization in xen," *USENIX*, 2006.

[11] J. Liu, W. Huang, B. Abali, and D. K. Panda, "High performance vmm-bypass i/o in virtual machines," *ATEC*, 2006.

[12] K. K. Ram, J. R. Santos, Y. Turner, A. L.Cox, and S. Rixner, "Achieving 10 gb/s safe and transparent network interface virtualization," *ACM*, Mar. 2009.

[13] Ν. Νικολέρης, "Σχεδίαση και Υλοποίηση μηχανισμού απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10gbe," 2009.