ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Δίκτυα Διασύνδεσης Υψηλής Επίδοσης σε Εικονικά Περιβάλλοντα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Στράτος Ψωμαδάκης

Επιβλέπων: Νεκτάριος Κοζύρης,
Αν. Καθηγητής ΕΜΠ

Αθήνα, Νοέμβριος 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Δίκτυα Διασύνδεσης Υψηλής Επίδοσης σε Εικονικά Περιβάλλοντα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Στράτος Ψωμαδάκης

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 28 Νοεμβρίου 2011.

.....	.....	.....
Ν. Κοζύρης	Δ. Σούντρης	Ν. Παπασπύρου
Αν. Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2011

.....

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

Περίληψη

Στην παρούσα διπλωματική εργασία μελετάται η συμπεριφορά των δικτύων διασύνδεσης υψηλής επίδοσης σε εικονικά περιβάλλοντα.

Η εργασία βασίστηκε στο δίκτυο διασύνδεσης που υλοποιήθηκε κατά τη διάρκεια της διπλωματικής εργσίας "Σχεδίαση και Υλοποίηση μηχανισμού απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10GbE", στο Εργαστήριο Υπολογιστικών Συστημάτων. Στα πλαίσια της εργασίας αυτής, σχεδιάστηκε και υλοποιήθηκε το πρωτόκολλο SLURPoE (Simple RDMA Protocol over Ethernet), η υλοποίηση του οποίου τροποποιήθηκε ώστε να ενταχθεί σε εικονικά περιβάλλοντα, και συγκεκριμένα στον ελεγκτή εικονικής μηχανής Xen, με χρήση του μοντέλου διαχωρισμένου οδηγού (split driver model), στην διπλωματική εργασία "Ένταξη Σημασιολογίας Δικτύων Διασύνδεσης Υψηλής Επίδοσης σε Εικονικές Μηχανές".

Αρχικά, εξετάζονται παράγοντες που ενδεχομένως να περιορίζουν την απόδοση του δικτύου διασύνδεσης, και του πρωτοκόλλου, τόσο στην αρχική όσο και στην εικονικοποιημένη υλοποίησή του. Με χρήση μετροπρογραμμάτων (benchmarks) έγινε προσπάθεια να εντοπιστούν και να βελτιστοποιηθούν πιθανά bottlenecks, τόσο σε επίπεδο υλικού, όσο και σε επίπεδο υλοποίησης πρωτοκόλλου.

Στην συνέχεια, ακολουθώντας μια διαφορετική προσέγγιση, υλοποιείται το εικονικοποιημένο πρωτόκολλο SLURPoE σε επίπεδο πυρήνα (kernel level).

Στις δύο προηγούμενες εργασίες η υλοποίηση του πρωτοκόλλου γινόταν, στο μεγαλύτερο κομμάτι της πάνω σε έναν 'έξυπνο' προσαρμογέα δικτύου (smart NIC), ενώ ο πυρήνας των κόμβων (hosts) αναλάμβανε μόνο την αρχικοποίηση της επικοινωνίας μεταξύ των εφαρμογών σε χώρο χρήστη και του προσαρμογέα δικτύου. Στόχος ήταν η όσο το δυνατόν μικρότερη επιβάρυνση των επεξεργαστών των κόμβων.

Στην παρούσα διπλωματική εργασία το πρωτόκολλο ενσωματώνεται στο κάτω μέρος (backend) του διαχωρισμένου οδηγού του Xen, υλοποιώντας ουσιαστικά το πρωτόκολλο στον πυρήνα του privileged guest του Xen, αντικαθιστώντας έτσι τον προσαρμογέα δικτύου.

Με την χρήση απλών μετρο-προγραμμάτων (micro-benchmarks), αξιολογήθηκε η διαφορά στην επίδοση μεταξύ των δύο υλοποιήσεων, σε ό,τι αφορά τόσο στο ρυθμό μεταφοράς δεδομένων (throughput) και τους χρόνους απόρκισης (latency), όσο και στην επιβάρυνση των επεξεργαστών των κόμβων. Η υλοποίηση σε επίπεδο πυρήνα

εμφανίζει μικρή επιβάρυνση στον επεξεργαστή του privileged guest του Xen (34% utilization), και σημαντική βελτίωση στους ρυθμούς μεταφοράς (έως 681 Mib/sec για μεγάλα μηνύματα).

Λέξεις-Κλειδιά: Δίκτυα Διασύνδεσης, Υπολογιστικά Συστήματα Υψηλής Απόδοσης, 10G Ethernet, Απευθείας Απομακρυσμένη Πρόσβαση στη Μνήμη, Πρωτόκολλο Δικτύωσης στο Χώρο Χρήστη, SLURPoE, Εικονική Μηχανή, Εικονικοποίηση, Xen, Μοντέλο Διαχωρισμένου Οδηγού

Abstract

The objective of this thesis is the study and evaluation of the performance of HPC interconnects in virtualized environments.

This thesis was based on the interconnect and protocol (SLURPoE) developed in the Computing Systems Laboratory during a previous diploma thesis entitled: "Design and Implementation of an RDMA mechanism over programmable 10GbE Interfaces", and its modified implementation, which intergrated the protocol in the Xen environment, using the split driver model, which was developed during the thesis "Integration of HPC Interconnect Semantics in virtualized environments".

Both the native and the virtualized implementations of the protocol are benchmarked, in order to identify and optimize possible bottlenecks, both in hardware, and in the protocol's implementation.

As a next step, the virtualized implementation of the protocol is ported to the host's kernel.

In the two previous theses, the core protocol implementation was done on a smart network adapter, while the hosts' kernel was involved only in the initialization of the communication between the user space applications and the network adapter. The goal was to minimize the overhead on the hosts' CPUs.

In this thesis, the protocol is integrated in the backend driver, of Xen's split driver, and in essence the protocol is implemented in the kernel of Xen's privileged guest, thus replacing the smart network adapter.

The performance of the new kernel-level implementation was evaluated with the use of simple microbencmakrs, in terms of throughput, latency, and overall CPU utilization. Preliminary results show that the kernel-level implementation achieves better throughput (up to 681 Mib/sec for large messages), while limiting the privileged guests CPU utilization to 34%.

Keywords: Interconnects, High Performance Computing, 10G Ethernet, Remote Direct Memory Access, User Level Networking, SLURPoE, Virtual Machine, Virtualization, Xen, Split Driver Model

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 για την ευκαιρία που μου έδωσε να ασχοληθώ με το συγκεκριμένο τομέα της επιστήμης των υπολογιστών στο Εργαστήριο Υπολογιστικών Συστημάτων, καθώς και για τις πολύτιμες συμβουλές του καθόλη τη διάρκεια περάτωσης της εργασίας.

Η εκπόνηση της διπλωματικής αυτής αποτελεί έμπνευση του υποψήφιου διδάκτορα Αναστάσιου Νάνου. Η βοήθειά του ήταν ανεκτίμητη τόσο για τις καίριες παρεμβάσεις του σε επίπεδο σχεδιασμού και υλοποίησης όσο και για τις τεχνικές γνώσεις που μου μετέδωσε. Χωρίς τη συμβολή του η διπλωματική εργασία δε θα ήταν εφικτή.

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

Στις μέρες μας, οι υποδομές Cloud Computing προσφέρουν μεγάλη ευελιξία και απομόνωση για την εκτέλεση ενός μεγάλου πλήθος εφαρμογών και υπηρεσιών. Οι υποδομές αυτές αποτελούνται κατά κανόνα από συστοιχίες υπολογιστών (clusters), με πολυπύρηνους επεξεργαστές, στις οποίες αναπτύσσεται ένα περιβάλλον εικονικών μηχανών, το οποίο αναλμβάνει την ασφάλεια, απομόνωση και τον κατάλληλο διαμοιρασμό των πόρων στις διάφορες υπηρεσίες. Οι συστοιχίες αυτές προσφέρουν πολύ μεγάλη επεξεργαστική ισχύ, γεγονός που τις καθιστά ιδανικές για πληθώρα εφαρμογών που απαιτούν υπολογιστική ισχύ (High Perfomance Computing applications).

Ιδιαίτερη σημασία για τις συστοιχίες υπολογιστών έχει το δίκτυο διασύνδεσης μεταξύ των κόμβων. Ένα δίκτυο διασύνδεσης υψηλής απόδοσης, το οποίο προσφέρει υψηλό εύρος ζώνης (bandwidth) και ρυθμούς μεταφοράς (throughput), χαμηλούς χρόνους απόκρισης (latency), και μεγάλη δυνατότητα κλιμάκωσης (scaling), είναι απαραίτητο, προκειμένου να αξιοποιηθεί στο έπακρο η επεξεργαστική ισχύς των υπολογιστών της συστοιχίας. Δυστυχώς, παρόλο που τα δίκτυα διασύνδεσης εξελίσσονται και αναβαθμίζονται συνεχώς, δεν συμβαδίζουν με την αύξηση στην επεξεργαστική ισχύ των κόμβων των συστοιχιών με αποτέλεσμα τη δημιουργία συμφόρησης (bottleneck), σε ό,τι αφορά στην επικοινωνία μεταξύ των κόμβων.

Σε εικονικά περιβάλλοντα, η επικοινωνία ενός κόμβου με το δίκτυο διασύνδεσης συνήθως απαιτεί πολλαπλά στρώματα λογισμικού (software layers), ώστε οι εικονικές μηχανές να αποκτήσουν πρόσβαση στο υλικό (προσαρμογέα δικτύου). Για παράδειγμα στον ελεγκτή εικονικής μηχανής (hypervisor) Xen, η πρόσβαση των εικονικών μηχανών στο υλικό υλοποιείται με την χρήση μιας εικονικής μηχανής με αυξημένα δικαιώματα (privileged guest). Αυτά τα στρώματα λογισμικού που απαιτούνται δημιουργούν μεγάλα και πολύπλοκα μονοπάτια δεδομένων (data paths) από την εικονική μηχανή μέχρι τον προσρμογέα δικτύου (και κατ' επέκταση το δίκτυο διασύνδεσης), μέσω πολλαπλών επιπέδων ανακατεύθυνσης (indirection), με αποτέλεσμα να μην επιτρέπουν την αξιοποίηση της επεξεργαστικής ισχύς στο μέγιστο.

Ακόμη και σε μη εικονικά περιβάλλοντα, η επικοινωνία μεταξύ των κόμβων παρουσιάζει αντίστοιχα προβλήματα απόδοσης, αν και σε μικρότερο βαθμό. Προκειμένου

μια εφαρμογή σε επίπεδο χρήστη να αποκτήσει πρόσβαση στον προσαρμογέα δικτύου, απαιτείται η παρέμβαση του λειτουργικού συστήματος (kernel), γεγονός που οδηγεί σε μη αποδοτικά μονοπάτια δεδομένων από την εφαρμογή στο δίκτυο διασύνδεσης. Η δικτύωση σε επίπεδο χρήστη (user level networking), η οποία εκμεταλλεύται δυνατότητες του λειτουργικού συστήματος, επιτρέπει στις εφαρμογές να παρακάμπτουν το λειτουργικό σύστυμα, και τους δίνει απευθείας πρόσβαση στον προσαρμογέα δικτύου. Επίσης, η χρησιμοποίηση πολύπλοκων πρωτοκόλλων με σημασιολογία (semantics) που δεν ενδείκνυνται για την επικοινωνία μεταξύ των κόμβων μια συστοιχίας πάνω από ένα δίκτυο διασύνδεσης υψηλής επίδοσης, δεν επιτρέπει την επίτευξη της μέγιστης δυνατής απόδοσης, και επιβαρύνει τους επεξεργαστές των κόμβων με το κόστος επεξεργασίας των πρωτοκόλλων. Για τον λόγο αυτό, έχουν δημιουργηθεί έξυπνοι προσαρμογείς δικτύου (smart NICs), οι οποίοι αναλαμβάνουν την επεξεργασία των πρωτοκόλλων, αφήνοντας τους επεξεργαστές των κόμβων ελεύθερους, αποκλειστικά για την εκτέλεση εφαρμογών (πχ TCP offloading engines), καθώς και πιο αποδοτικά πρωτόκολλα για χρήση σε δίκτυα διασύνδεσης υψηλής απόδοσης (πχ το πρωτόκολλο MX του Myrinet, το Verbs/RDMA για Infiniband κλπ).

Η χρήση αυτών των τεχνικών σε εικονικά περιβάλλοντα όμως παρουσιάζει τεχνικές δυσκολίες λόγων των επιπλέων επιπέδων λογισμικού που παρεμβάλονται μεταξύ της εφαρμογής σε επίπεδο χρήστη και του προσρμογέα δικτύου. Για παράδειγμα, στο περιβάλλον εικονικών μηχανών Xen, Προκειμένου να αποκτήσει μια εφαρμογή σε επίπεδο χρήστη απευθείας πρόσβαση στο υλικό απαιτείται η παρέμβαση του Xen hypervisor. Όμως σε αυτήν την περίπτωση ανακύπτουν θέματα ασφάλειας και απομόνωσης των διάφορων εφαρμογών που τρέχουν στις εικονικές μηχανές του κάθε κόμβου.

Σε επίπεδο υλικού έχουν υλοποιηθεί δύο τεχνολογίες, οι SR-IOV και MR-IOV, που προσφέρουν στις εικονικές μηχανές άμεση πρόσβαση στο υλικό, μειώνοντας έτσι τον αριθμό των στρωμάτων που παρεμβάλλονται μεταξύ της εφαρμογής και του υλικού, αντιμετωπίζοντας θέματα απομόνωσης και ασφάλειας μεταξύ εικονικών μηχανών. Επειδή όμως οι τεχνολογίες αυτές είναι υλοποιημένες σε υλικό, τίθεται το θέμα της δυνατότητας κλιμάκωσης (scaling) σε περίπτωση μεγάλου αριθμού εικονικών μηχανών. Επίσης, οι τεχνολογίες αυτές δεν λύνουν τα υπόλοιοπα προβλήματα που αναφέρθηκαν παραπάνω. Ακόμα και με τη χρησιμοποίηση SR-IOV, δεν παρακάμπτεται ο πυρήνας (kernel) της εικονικής μηχανής, ενώ το πρόβλημα των μη αποδοτικών πρωτοκόλλων δεν λύνεται, αφού η πιο συνηθισμένη λύση για την επικοινωνία μεταξύ εικονικών μηχανών/κόμβων σε μια συστοιχία υπολογιστών είναι η χρησιμοποίηση MPI πανώ από TCP/IP.

Εναλλακτικά, αντί για τεχνολογίες υλοποιημένες στο υλικό, χρησιμοποιώντας paravirtualized εικονικές μηχανές, και ένα μοντέλο διαχωρισμένου οδηγού (split driver model) μπορούμε να υλοποιήσουμε και να χρησιμοποιήσουμε αντίστοιχες τεχνολογίες σε λογιμισκό, επεκτείνοντας ουσιαστικά το user-level networking, σε VM-level networking. Αυτή η τεχνική χρησιμοποιήθηκε στην διπλωματική εργασία "Ένταξη Σημασιολογίας Δικτύων Διασύνδεσης Υψηλής Επίδοσης σε Εικονικές Μηχανές", παράλληλα με την χρησιμοποίηση του πρωτοκόλλου SLURPoE, το οποίο σχεδιάστηκε και υλοποιήθηκε στην διπλωματική εργασία "Σχεδίαση και Υλοποίηση μηχανισμού

απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10GbE". Και στις δύο προηγούμενες διπλωματικές εργασίες η υλοποίηση του πρωτοκόλλου έγινε σε έναν έξυπνο προσαρμογέα δικτύου ο οποίος αναλάμβανε το κόστος επεξεργασίας και υλοποίησης του πρωτοκόλλου. Σε αυτή την διπλωματική θα μελετήσουμε τις συνέπειες που έχει στην απόδοση, τόσο σε ρυθμούς μεταφοράς και χρόνους απόκρισης, όσο και στην επιβάρυνση του επεξεργαστή, η υλοποίηση του πρωτοκόλλου στον πυρήνα κάθε κόμβου σε σχέση με τον προσρμογέα δικτύου.

1.1 Σκοπός

Αντικείμενο της παρούσας εργασίας είναι η μελέτη και η ανάλυση της απόδοσης διαφορετικών τεχνικών υλοποιήσεων πρωτοκόλλων και δικτύων διασύνδεσης υψηλής επίδοσης σε εικονικά περιβάλλοντα.

Χρησιμοποιούμε ως αναφορά το πρωτόκολλο SLURPoE, καθώς και το δίκτυο διασύνδεσης, 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", και την υλοποίηση του συγκεκριμένου πρωτοκόλλου για χρήση σε εικονικά περιβάλλοντα, η οποία χρησιμοποεί το paravirtualized εικονικό περιβάλλον Xen, και το μοντέλο διαχωρισμένου οδηγού.

Αρχικά, με τη βοήθεια μετροπρογραμμάτων, γίνεται προσπάθεια να εντοπιστούν πιθανά σημεία συμφόρησης (bottlenecks), σε ό,τι αφορά τόσο το δίκτυο διασύνδεσης όσο και την υλοποίηση του πρωτοκόλλλου (SLURPoE), και προτείνονται ορισμένες βελτιστοποιήσεις.

Στη συνέχεια, παρουσιάζεται μια υλοποίηση του πρωτοκόλλου SLURPoE για εικονικά περιβάλλοντα σε επίπεδο πυρήνα (kernel-level).

Στις δύο προηγούμενες υλοποιήσεις του πρωτοκόλλου SLURPoE, έγινε χρήση έξυπνων προσαρμογέων δικτύου, οι οποίοι αναλάμβαναν το κόστος υλοποίησης και 'επεξεργασίας' του πρωτοκόλλου(protocol offloading). Εμείς σχεδιάζουμε και υλοποιούμε για το περιβάλλον Xen (και Linux), το κάτω μέρος (backend) του διαχωρισμένου οδηγού, έτσι ώστε η υλοποίηση του πρωτοκόλλου να μην γίνεται πλέον σε έναν έξυπνο προσαρμογέα δικτύου, αλλά εξ' ολοκλήρου στον πυρήνα (kernel) του privileged guest του Xen (kernel level implementation).

Με την χρήση μετρο-προγραμμάτων (micro-benchamrks) συγκρίνουμε την διαφορά στους ρυθμούς και τους χρόνους μεταφοράς, αλλά και την επιβάρυνση του επεξεργαστή του κόμβου, ανάμεσα στις δύο υλοποιήσεις.

Σκοπός μας είναι να πετύχουμε όσο το δυνατόν μεγαλύτερη βελτίωση στους ρυθμούς και τους χρόνους μεταφοράς, επιβαρύνοντας όσο το δυνατόν λιγότερο τους επεξεργαστές των κόμβων.

1.2 Η δομή του κειμένου

Στη συνέχεια παρουσιάζεται αναλυτικά η μεθοδολογία αξιολόγησης της απόδοσης των προηγούμενων υλοποιήσεων, η υλοποίηση του πρωτοκόλλου σε επίπεδο πυρήνα, καθώς και το απαραίτητο θεωρητικό υπόβαθρο.

Πιο συγκεκριμένα στο κεφάλαιο 2 γίνεται αναφορά στα απαραίτητα θεωρητικό υπόβαθρο, και συγκεκριμένα στα δίκτυα διασύνδεσης υψηλής επίδοσης και το 10GbE, καθώς και στις διάφορες τεχνικές εικονικοποίησης virtualization, και κυρίως στο περιβάλλον εικονικών μηχανών του Xen.

Στο κεφάλαιο 3 παρουσιάζεται μια ανάλυση της απόδοσης των δύο προηγούμενων υλοποιήσεων του πρωτοκόλλου, και προτείνονται βελτιστοποιήσεις.

Στο κεφάλαιο 4 παρουσιάζεται η υλοποίηση του πρωτοκόλλου SLURPoE σε επίπεδο πυρήνα, και αναφέρονται οι βασικότερες αλλαγές που απαιτήθηκαν.

Τέλος, στο κεφάλαιο 5 παραθέτουμε τα αποτελέσματα των μετρήσεων της υλοποίησής μας, με αναλυτικά διαγράμματα για όλες τις παραμέτρους που επηρεάζουν την απόδοση του συστήματος, ενώ στο κεφάλαιο 6 αναφέρονται τα τελικά συμπεράσματα για την υλοποίηση του πρωτοκόλλου σε επίπεδο πυρήνα, σε σχέση με την υλοποίηση με την χρήση έξυπνων προσαρμογέων δικτύου, καθώς και μια αναφορά σε παρόμοιες εργασίες και μελλοντικές επεκτάσεις της διπλωματικής εργασίας.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Το κεφαλαίο αυτό έχει ως σκοπό την περιγραφή των εννοιών πάνω στις οποίες βασίστηκε η παρούσα διπλωματική, και χωρίζεται σε δύο μέρη: το πρώτο αφορά στην περιγραφή της λειτουργίας των εικονικών μηχανών, και συγκεκριμένα του περιβάλλοντος εικονικών μηχανών Xen, ενώ το δεύτερο στα δίκτυα διασύνδεσης υψηλής επίδοσης και ειδικότερα στην χρήση τους σε εικονικά περιβάλλοντα.

2.1 Εικονικοποίηση

Ο όρος εικονική μηχανή έχει χρησιμοποιηθεί ευρέως από διάφορους τομείς της επιστήμης των υπολογιστών, για να περιγράψει ένα επίπεδο αφαίρεσης, το οποίο υλοποιώντας 'εικονικά' υπολογιστικούς πόρους, αποκρύπτει από τους πελάτες των πόρων αυτών λεπτομέρειες για την υλοποίηση ή την κατάστασή των πραγματικών πόρων που χρησιμοποιούνται, αναλαμβάνοτας να πολυπλέκει τις αιτήσεις από τους εικονικούς πόρους προς τους πραγματικούς, καθώς και την απομόνωση (isolation) μεταξύ τών πελατών.

Οι Popek και Goldberg όρισαν την εικονική μηχανή ως εξής: ``Εικονική μηχανή θεωρείται ένα αποδοτικό και απομονωμένο αντίγραφο μιας πραγματικής μηχανής''. [1] Ωστόσο, πλέον ο όρος εικονική μηχανή έχει επεκταθεί και περιλαμβάνει ένα ευρύτερο σύνολο υλοποιήσεων, όχι μόνο σε επίπεδο πραγματικών μηχανών, αλλά και σε επίπεδο λειτουργικών συστημάτων και εφαρμογών σε επίπεδο χρήστη.

Εικονικές μηχανές σε επίπεδο εφαρμογών χρήστη υποστηρίζουν κατά κανόνα, αλλά όχι απαραίτητα, μία συγκεκριμένη γλώσσα προγραμματισμού. Οι εφαρμογές γραμμένες στην γλώσσα αυτή χρησιμοποιούν ένα διερμηνέα για να μετατραπεί ο αρχικός κώδικας σε μια ειδική μορφή (συχνά αναφέρεται ως bytecode), η οποία θα τρέξει στο εικονικό περιβάλλον που δημιουργεί η εικονική μηχανή. Το πιο χαρακτηριστικό παράδειγμα εικονικής μηχανής σε επίπεδο εφαρμογών χρήστη αποτελεί το Java Virtual Machine. Οι εικονικές μηχανές αυτού του τύπου τρέχουν ως κανονικές διεργασίες του λειτουργικού συστήματος, και στόχο έχουν να αποκρύψουν τις λεπτομέρειες για

το λειτουργικό σύστημα πάνω στο οποίο τρέχουν οι εφαρμογές, δημιουργώντας ένα καθορισμένο εικονικό περιβάλλον για τις εφαρμογές, και αναλμβάνοντας αυτές τις λεπτομέρειες για την επικοινωνία με το εκάστοτε λειτουργικό σύστημα, και αρχιτεκτονική πλατφόρμα, επιτρέποντας στις εφαρμογές να τρέξουν με διαφανή (transparent) τρόπο πάνω σε διαφορετικά λειτουργικά συστήματα, και αρχιτεκτονικές.

Εικονικές μηχανές σε επίπεδο λειτουργικού συστήματος δημιουργούν πολλαπλά απομονωμένα στιγμιότυπα του λειτουργικού συστήματος, σε επίπεδο χρήστη. Τέτοιου τύπου εικονικές μηχανές αποτελούν επέκταση κλασικών τεχνικών απομώνσης του λειτουργικού συστήματος Unix (chroot/jails), προσθέτοντας όμως περισσότερες δυνατότητες για λεπτομερή διαχείρηση των πόρων του συστήματος, και απομόνωση των διάφορων στιγμιοτύπων σε επίπεδο πόρων. Χαρακτηριστικά παραδείγματα τέτοιων εικονικών μηχανών αποτελούν το OpenVZ καθώς και τα Solaris Containers.

Τέλος, οι εικονικές μηχανές σε επίπεδο υλικού δημιουργούν ένα εικονικό περιβάλλον σε επίπεδο υλικού, το οποίο συμπεριφέρεται ακριβώς, ή σχεδόν ακριβώς, όπως ένας πραγματικός υπολογιστής σε επίπεδο υλικού.

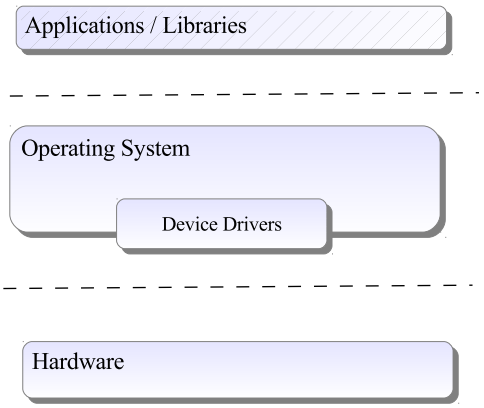
Με την τεχνική της εξομείωσης, μπορεί να δημιουργηθεί σε εικονικό επίπεδο και να εξομειωθεί ένας μεγάλος αριθμός αρχιτεκτονικών, και υπολογιστικών πόρων, ανεξάρτητα από το υλικό και την αρχιτεκτονική του συστήματος, και χρησιμοποιείται κυρίως όταν χρειάζεται, για λόγους πχ testing, να τρέξουν εφαρμογές σε διαφορετικό υλικό, από αυτό του συστήματος. Ένας από τους πιο γνωστούς εξομειωτές υλικού είναι το QEMU, το οποίο υποστηρίζει και εξομειώνει ένα μεγάλο αριθμό αρχιτεκτονικών, αλλά και συσκευών.

2.1.1 Hardware Virtualization

Μια πιο συνηθισμένη τεχνική, είναι η εικονικοποίηση μόνο του υλικού του συστήματος. Οι εικονικές μηχανές αυτού του τύπου, χρησιμοποιούν ένα πρόσθετο στρώμα λογισμικού (software layer), το οποίο συνήθως ονομάζεται επόπτης (hypervisor), και το οποίο έχει απόλυτη πρόσβαση στο υλικό και τους πόρους του συστήματος. Ο επόπτης δημιουργεί εικονικά αντίγραφα/στιγμιότυπα του υλικού του συστήματος, ενώ αναλμβάνει επίσης και τον διαμοιρασμό των πόρων του συστήματος μεταξύ των εικονικών στιγμιοτύπων, και την απόμονωση μεταξύ τους.

Η κύρια διαφορά με την τεχνική της εξομείωσης είναι ότι, αφού εικονικοποιείται μόνο η αρχιτεκτονική του συστήματος, ο κώδικας που εκτελείται στις εικονικές μηχανές, μπορει κατά κανόνα να εκτλεστεί natively πάνω στους φυσικούς επεξεργαστές του συστήματος.

Στις αρμοδιότητες του ελεγκτή εικονικών μηχανών εντάσσεται η χρονοδρομολόγηση των εικονικών μηχανών και τον εικονικών επεξεργαστών πάνω στους φυσικούς επεξεργαστές του συστήματος (VM/vCPU scheduling), η εικονικοποίηση και η ασφαλής και απομονωμένη πρόσβαση στη μνήμη, αλλά και στις συσκευές Εισόδου / Έξόδου του συστήματος (Memory / I/O multiplexing), αλλά και η διαχείρηση και η κατάλληλη δρομολόγηση των διακοπών στις εικονικές μηχανές (interrupt routing).



Σχήμα 2.1: Υπολογιστής χωρίς εικονικοποίηση

Οι ελεγκτές εικονικών μηχανών μπορούν να διαχωριστούν σε δύο κατηγορίες, με βάση τον τρόπο με τον οποίο έχουν υλοποιηθεί.

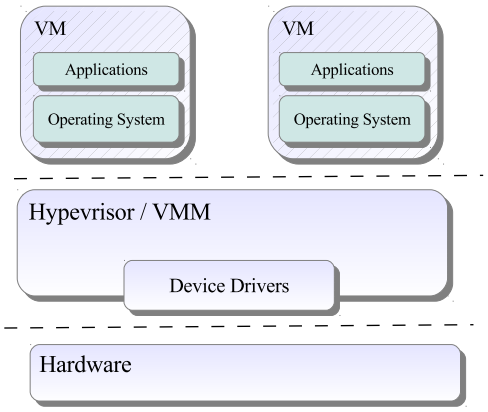
Μία τεχνική υλοποιεί έναν lightweight hypervisor (type I), ο οποίος τρέχει πάνω στο υλικό και αναλμβάνει βασικές λειτουργίες, όπως vm scheduling, memory management κλπ, ενώ παραχωρεί αυξημένα δικαιώματα σε ορισμένες εικονικές μηχανές, οι οποίες μπορούν να διαχεριίζονται τις συσκευές, να επικοινωνούν με τον hypervisor, και να παρέχουν την διεπαφή ελέγχου του hypervisor για τον έλεγχο του εικονικού περιβάλλοντος και των εικονικών μηχανών. Με αυτή την τεχνική υλοποιούνται οι ελεγκτές Xen και VMware. (Σχήμα 2.3)

Μια άλλη προσέγγιση είναι να χρησιμοποιηθεί ένα έτοιμο λειτουργικό σύστημα, στον πυρήνα του οποίου ενσωματώνεται ο ελεγκτής εικονικών μηχανών (type II). Με αυτόν τον τρόπο ο ελεγκτής μπορεί να εκμεταλλευτεί όλες τις λειτουργίες που είναι ήδη υλοποιημένες στο λειτουργικό σύστημα (scheduling, memory management, device drivers). Αυτή την τεχνική χρησιμοποιεί ο ελεγκτής εικονικών μηχανών KVM, ο οποίος ουσιαστικά μετατρέπει τον πυρήνα του Linux σε hypervisor. (Σχήμα 2.4)

Full Virtualization

Η δημιουργία πανομοιότυπων εικονικών στγμιοτύπων του υλικού ενός συστήματος (Full Virtualization), επιτρέπει σε εφαρμογές (κατά κανόνα ολόκληρα λειτουργικά συστήματα) να τρέξουν διαφανώς (trasparently) πάνω στο εικονικό στιγμιότυπο, χωρίς καμία αλλαγή στον κώδικά τους. Οι ελεγκτές εικονικών μηχανών KVM και VMware είναι χαρακτηριστικά παραδείγματα hypervisors που υποστηρίζουν full virtualization.

Ωστόσο, για είναι αποδοτική αυτή η τεχνική πρεέπει να ικανοποιούνται συγκεκριμένες προϋποθέσεις, τις οποίες περιέγραψαν οι Popek και Goldberg [1]. Οι προϋπο-



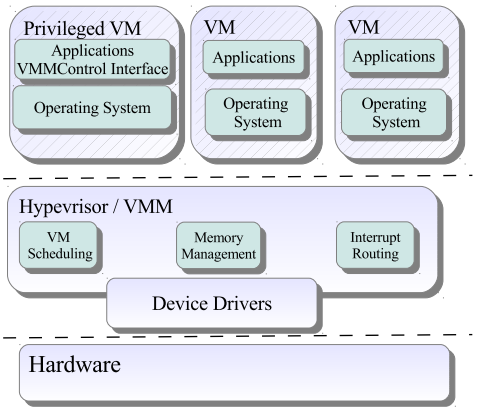
Σχήμα 2.2: Υπολογιστής με εικονικοποίηση

θέσεις αυτές έχουν να κάνουν με το γεγονός ότι οι πελάτες που τρέχουν πάνω στα εικονικά στιγμιότυπα μπορούν να τρέξουν εντολές με αυξημένα δικαιώματα (privileged instructions), και οι οποίες μπορεί να αλλάξουν την κατάσταση του συστήματος. Αυτές οι εντολές, σύμφωνα με τους Popek και Goldberg, θα πρέπει να ειδοποιούν με κάποιου είδους διακοπή (trap) τον επεξεργαστή, όταν εκτελούνται με λιγότερα δικαιώματα, έτσι ώστε ο επόπτης (hypervisor), να μπορεί να τις ελέγχει, και να εγγυάται την απομόνωση μεταξύ των διαφορετικών εικονικών στιγμιοτύπων. Σε αντίθεση περίπτωση ο επόπτης θα πρέπει να ελέγχει manually για αυτές τις εντολές, και να αλλάζει τον κώδικα on-the-fly ώστε αυτές να μην επηρεάσουν την κατάσταση ολόκληρου του συστήματος.

Δυστυχώς, αρκετές αρχιτεκτονικές δεν πληρούν αυτές τις προϋποθέσεις, συμπεριλαμβανομένης και μιας από της πιο ευρέως χρησιμοποιούμενες αρχιτεκτονικής, της Intel x86. Προκειμένου να επιτευχθεί αποδοτική εικονικοποίηση σε αυτές τις αρχιτεκτονικες, υλοποιήθηκαν επεκτάσεις σε επίπεδο υλικού (virtualization extensions), οι οποίες διευκολύνουν το έργο των ελεγκτών εικονικών μηχανών, όπως για παράδειγμα οι τεχνολογίες Intel-VTx [2] και AMD-V [3].

Οι τεχνικές πλήρους εικονικοποίησης παρουσιάζουν επίσης θέματα απόδοσης και σε ό,τι αφορά τις λειτουργίες εισόδου / εξόδου (I/O). Η συνηθέστερη πρακτική είναι να παρουσιάζονται στις εικονικές μηχανές εξομειωμένες συσκευές εισόδου / εξόδου. Οι εικονικές αυτές συσκευές υλοποιούνται συνήθως από τον hypervisor, ο οποίος αναλαμβάνει να πολυπλέξει (multiplexing) τις αιτήσεις των εικονικών μηχανών προς τις εικονικές συσκευές, και να τις μεταφράζει σε κατάλληλες αιτήσεις προς τις πραγματικές συσκεύες. (Σχήμα 2.5).

Η τεχνική αυτή όμως παρουσιάζει μεγάλη επιβάρυνση στην απόδοση των λειτουργιών εισόδου / εξόδου. Μια άλλη λύση είναι η απευθείας 'ανάθεση' μιας φυσικής συσκευής σε μια εικονική μηχανή. Το πρόβλημα με αυτή την λύση είναι ότι η συσκευή δεν



Σχήμα 2.3: Type I Hypervisor

μπορεί να διαμοιραστεί μεταξύ εικονικών μηχανών, ενώ ανακύπτουν και ζητήματα ασφάλειας και απομόνωσης, καθιστώντας απαραίτητη την ύπαρξη ένος IOMMU υλοποιημένου σε υλικό [4].

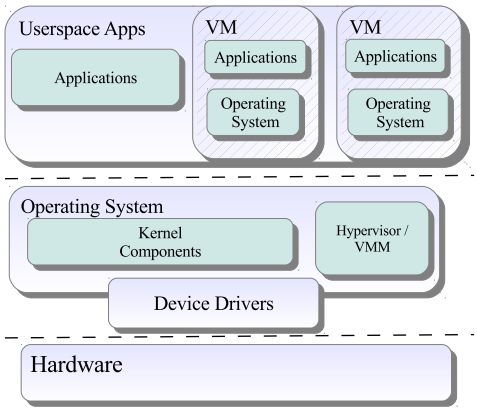
Για τον λόγο αυτό, αναπτύχθηκαν οι τεχνολογίες SR/MR-IOV [5], οι οποίες δίνουν τη δυνατότητα στις συσκευές να παρουσιάζουν στον ελεγκτή εικονικών μηχανών διακριτά και απομονωμένα στιγμιότυπα της συσκευής, τα οποία μπορούν να ανατεθούν σε διαφορετικές εικονικές μηχανές. Οι τεχνολογίες αυτές όμως προϋποθέτουν υποστήριξη από το υλικό των συσκευών, ενώ ανακύπτουν και ζητήματα κλιμάκωσης, αλλά και ασφάλειας, καθώς και σε αυτή την περίπτωση είναι αναγκαία η ύπαρξη ενός hardware IOMMU.

Para Virtualization

Σε αντίθεση με την πλήρη εικονικοποίηση, η τεχνική του Para Virtualization εικονικοποεί μερικώς το υλικό του υπολογιστή, και κατά συνέπεια απαιτεί τροποποιήσεις στα λειτουργικά συστήματα που θα εκτελεστούν στις εικονικές μηχανές (VM-aware OS). [6]

Με την τεχνική του Para Virtualization, εισάγεται η έννοια των υπερ-κλήσεων hypercalls. Τα λειτουργικά συστήματα που τρέχουν σε paravirtualized εικονικές μηχανές, τροποποιούνται έτσι ώστε όταν χρειάζεται να τρέξουν privileged εντολές, εκτελούν ένα hypercall, με το οποίο επικοινωνούν με τον hypervisor και του ζητούν να εκτελέσει την ενέργεια που επιθυμουν.

Έτσι, η διεπαφή των εικονικών μηχανών τροποποιείται, ώστε να διευκολύνεται η εικονικοποίηση (ειδικά για αρχιτεκτονικές όπως η x86), γεγονός το οποίο προσφέρει επιπλέον δυνατότητες βελτιστοποίησης λειτουργιών σημαντικών για την απόδοση



Σχήμα 2.4: Type II Hypervisor

του συστήματος (λειτουργίες εισόδου εξόδου, διαχείρηση μνήμης, κλήσεις συστήματος, διακοπες κλπ).

Καθώς, όμως, οι επεκτάσεις υλικού για την υποστήριξη της εικονικοποίησης βελτιώνονται με ταχείς ρυθμούς, έχει αρχίσει να γίνεται πιο διαδομένη μια υβριδική προσέγγιση, μεταξύ πλήρους εικονικοποιήσης και Para Virtualization, και περιλαμβάνει την χρήση πλήρους εικονικοποίησης σε ό,τι αφορά υπολογιστικούς πόρους όπως οι επεξεργαστές και η μνήμη, και τροποιποιημένους οδηγούς συσκευών (para virtualized device drivers) για λειτουργίες εισόδου / εξόδου, για να παρακαμθούν τα εγγενή προβλήματα των λειτουργιών αυτών σε τεχνικές πλήρους εικονικοποιήσης. (Σχήμα 2.6

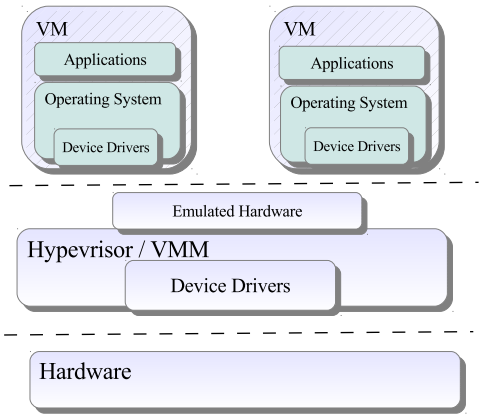
2.1.2 XEN

Ο ελεγκτής εικονικών μηχανών Xen [7] είναι ένας Type-I hypervisor, ο οποίος υλοποιεί την τεχνική του Para Virtualization.

Ο hypervisor τρέχει πάνω στο υλικό του υπολογιστή, και υλοποιεί μόνο τις απολύτως απαραίτητες λειτουργίες που απαιτούνται από έναν hypervisor (VM scheduling, memory management, interrupt routing κλπ),

Όλες οι υπόλοιπες λειτουργίες τις αναλαμβάνουν ειδικές εικονικές μηχανές με αυξημένα δικαιώματα (privileged guest και driver domains). Ο privileged guest είναι μια εικονική μηχανή με αυξημένα δικαιώματα, η οποία μπορεί να επικοινωνεί με τον Xen hypervisor, και υλοποιεί την διεπαφή ελέγχου με τον hypervisor, την οποία χρησιμοποιεί ο διαχειριστής κάθε συστήματος, για την ρύθμιση του hypervisor, την διαχείριση των εικονικών μηχανών κλπ.

Ο privileged guest έχει επίσης δικαιώματα να επικοινωνεί άμεσα με όλες τις συσκευές του υπολογιστή, και κατά κανόνα είναι αυτός που αναλαμβάνει να πολυπλέξει τις



Σχήμα 2.5: Full Virtualization I/O - Emulated Devices

αιτήσεις για λειτουργίες εισόδου / εξόδου από διαφορετικές εικονικές μηχανές και να τις περάσει στο πραγματικό υλικό. Την λειτουργία αυτή μπορούν επίσης να αναλάβουν ειδικές εικονικές μηχανές, που έχουν αποκλειστική άμεση επικοινωνία με το υλικό (driver domains).

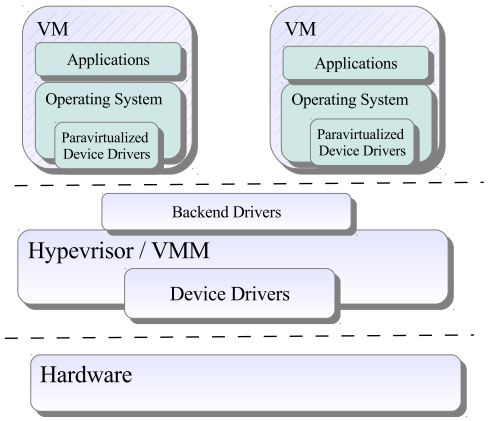
Split Driver Model

Το Xen υποστηρίζει και υλοποιεί ένα μοντέλο διαχωρισμένου οδηγού για την επικοινωνία των εικονικών μηχανών με συσκευές εισόδου / εξόδου. Σε αυτό το μοντέλο, ο οδηγός χωρίζεται σε δύο μέρη. Το πάνω μέρος του οδηγού (frontend) υλοποιείται στον πυρήνα του λειτουργικού που τρέχει στις unprivileged εικονικές μηχανές, και δίνει στις εικονικές μηχανές μια γενική διεπαφή για την λειτουργία που απαιτείται, και επικοινωνεί με το κάτω μέρος του οδηγού (backend).

Το backend υλοποιείται συνήθως στον πυρήνα του privileged guest του Xen (ή στον πυρήνα ενός driver domain), και αναλαμβάνει την επικοινωνία με το πραγματικό υλικό του υπολογιστή, μέσω των οδηγών συσκευών που υπάρχουν στον πυρήνα.

Η επικοινωνία μεταξύ των δύο οδηγών γίνεται με ειδικούς μηχανισμούς που παρέχει το περιβάλλον του Xen, και περιγράφονται στη συνέχεια.

Το εικονικό περιβάλλον του Xen, εκτός από Para Virtualization, υποστηρίζει επίσης και πλήρη εικονικοποίηση, με χρήση των επεκτάσεων υλικού VT-x / AMD-V. Σε αυτή την περίπτωση, δεν είναι δυνατή η χρήση του μοντέλου διαχωρισμένου οδηγού, και χρησιμοποιούνται τεχνικές πλήρους εικονικοποίησης που περιγράψαμε παραπάνω. (Σχήμα 2.7)



Σχήμα 2.6: Para-Virtualization I/O

Μηχανισμοί Επικοινωνίας

Προκειμένου να είναι εφικτή η επικοινωνία μεταξύ των διαχωρισμένων οδηγών του Xen, ο hypervisor υλοποιεί μηχανισμούς επικοινωνίας μεταξύ των εικονικών μηχανών.

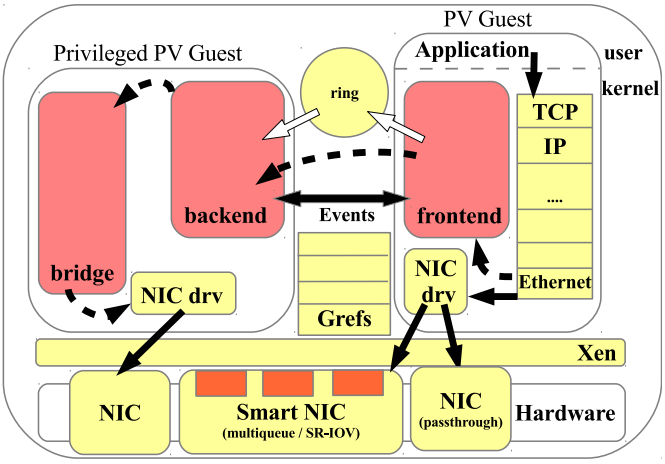
Το Xen υλοποιεί έναν μηχανισμό shared memory, τον οποίο ονομάζει Grant Mechanism, μέσω του οποίου διάφορες εικονικές μηχανές δίνουν δικαιώματα ανάγνωσης / εγγραφής σε περιοχές της μνήμης τους, σε άλλες εικονικές μηχανές. Τα grant references που περιγράφουν αυτά τα δικαιώματα αποθηκευόνται σε μία μοναδική ανά εικονική μηχανή δομή, το grant table.

Με τη βοήθεια των grants, το Xen προσφέρει τη δυνατότητα δημιουργίας ειδικών producer / consumer δομών, οι οποίες ονομάζονται ring buffers, για την ανταλλαγή δεδομένων μεταξύ των εικονικών μηχανών. Δημιουργείται έτσι ένα κανάλι επικοινωνίας μεταξύ των εικονικών μηχανών, όπου ο producer κάνει push requests / pull responses στον ring buffer, ενώ ο consumer push responses / pull requests. (Σχήμα 2.8)

Προκειμένου να γίνει εφικτή η μεταφορά δεδομένων μεταξύ των εικονικών μηχανών με τη χρήση των ring buffers, πρέπει να δημιουργηθεί ένα κανάλι επικοινωνίας, όπου ο κάθε guest να μπορεί να ειδοποιήσει τον άλλο για outstanding requests / responses.

Το Xen υλοποιεί έναν doorbell μηχανισμό επικοινωνίας, με τη χρήση γεγονότων (events). Τα events του Xen θυμίζουν εικονικές διακοπές (virtual interrupts), έχοντας 'δικτυακά' semantics. Ένας guest δεσμεύει ένα event channel, κάνει bind στο ένα virtual end, και εκχωρεί το δικαίωμα σε έναν άλλο guest να κάνει bind στο άλλο άκρο, σε συγκεκριμένη θύρα (port).

Για την αρχικοποιήση της επικοινωνίας μεταξύ των εικονικών μηχανών, καθώς και για το πέρασμα αρχικών ρυθμίσεων από τον hypervisor, το Xen υλοποιεί μια ιεραρχική δομή, το Xenstore. Το Xenstore έχει δενδρική δομή, και τα φύλλα του δέντρου αποτε-



Σχήμα 2.7: Xen I/O - Split Driver Model / Pass-through / SR-IOV

λούνται από ζευγάρια (key / value). Στα φύλλα αποθηκεύονται πληροφορίες για την αρχικοποίηση των εικονικών μηχανών, για νέα grant references από άλλες εικονικές μηχανές, ενώ χρησιμοποιείται επίσης και ως ένας μηχανισμός device discovery από τις εικονικές μηχανές.

Στον πυρήνα του Linux, υλοποιείται μια προγραμματιστική διεπαφή διαύλου, για την επικοινωνία των εικονικών μηχανών με τον Xenstore, η οποία ονομάζεται Xenbus.

Στο σχήμα 2.9 παρουσιάζεται ένα overview της λειτουργίας των paravirtualized οδηγών δικτύου που υλοποιεί το Xen, οι οποίοι χρησιμοποιούν τους παραπάνω μηχανισμούς επικοινωνίας, για την υλοποίηση του split driver.

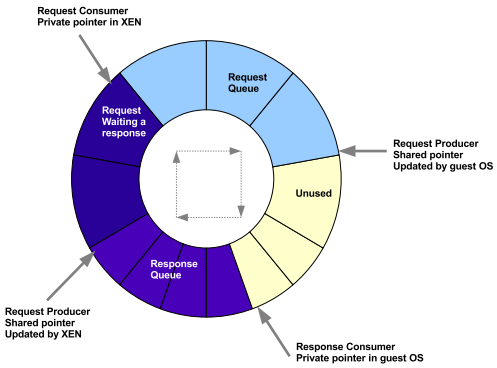
2.2 Δίκτυα Διασύνδεσης Υψηλής Επίδοσης

Εφαρμογές με πολύ μεγάλες υπολογιστικές απαιτήσεις, όπως επιστημονικές εφαρμογές που μοντελοποιούν και προσομειώνουν φυσικά φαινόμενα κλπ, οδήγησαν στην δημιουργία παράλληλων και πολυνηματικών αρχιτεκτονικών, λόγω του εγγενή παραλληλισμού στην επεξεργασία των δεδομένων που διαχειρίζονται.

Το γεγονός αυτό, λόγω και του κόστους και της δυσκολίας κλιμάκωσης SMP αρχιτεκτονικών, με πολλαπλούς επεξεργαστές ενσωματωμένους στον ίδιο κόμβο, οδήγησε στην δημιουργία συστοιχιών υπολογιστών, με δεκάδες ή και εκατοναδες κόμβους.

Η επικοινωνία μεταξύ των κόμβων επιτυγχάνεται με δίκτυα διασύνδεσης, και βιβλιοθήκες επικοινωνίας (MPI), οι οποίες παρουσιάζουν στις εφαρμογές μια καλά ορισμένη προγραμματιστκή διεπαφή, αποκρύπτωντας τις λεπτομέρειες του δικτύου διασύνδεσης και των πρωτοκόλλων επικοινωνίας.

Η απόδοση του δικτύου διασύνδνεσης, αλλά και των πρωτοκόλλων επικοινωνίας που χρησιμοποιούνται, επηρεάζει σημαντικά την γενικότερη απόδοση του συστήματος. Η



Σχήμα 2.8: Xen Ring Buffers

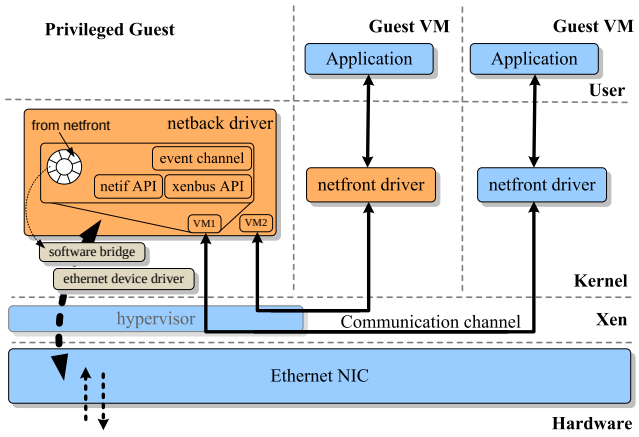
δυνατότητα για κλιμάκωση χωρίς να επηρεάζεται η απόδοση, καθώς και η απαίτηση για υψηλούς ρυθμούς μεταφοράς, και χαμηλούς χρόνους απόκρισης, καθιστά τα κλασικά δίκτυα και δικτυακά πρωτόκολλα ακατάλληλα για χρήση σε τέτοιες συστοιχίες.

Τα δίκτυα διασύνδεσης υψηλής επίδοσης, που αναπτύχθηκαν για χρήση σε συστοιχίες υπολογιστών, χρησιμοποιούν lightweight πρωτόκολλα επικοινωνίας, με semantics που ταιριάζουν καλύτερα στην επικοινωνία μεταξύ των κόμβων μιας συστοιχίας. Επίσης, αναπτύχθηκαν μηχανισμοί και υλικό για το offloading του κόστους επεξεργασίας και υλοποίησης τους πρωτοκόλλου, από τους επεξεργαστές των κόμβων, σε έξυπνους προσαρμογείς δικτύου, με στόχο την όσο το δυνατόν μικρότερη επιβάρυνση των επεξεργαστών των κόμβων κατά την μεταξύ τους επικοινωνία. Οι έξυπνοι αυτοί προσαρμογείς, διαθέτουν μηχανές DMA, για την μεταφορά δεδομένων χωρίς την παρέμβαση των επεξεργαστών των κόμβων, αλλά και αρκετούς υπολογιστικούς πόρους (volatile memory, I/O processors), ώστε να αναλάβουν αυτοί την πολυπλεξία των αιτήσεων μεταξύ πολλαπλών εφαρμογών που χρησιμοποιούν τον προσαρμογέα, καθώς και την αποστολή των δεδομένων.

Προκειμένου να πετύχουν χαμηλό latency αναπτύχθηκαν μηχανισμοί παράκαμψης του πυρήνα του λειτουργικού συστήματος (OS-bypass) [8], όπως zero copy, page-flipping, pagecache bypass, γεγονός το οποίο οδήγησε στην ιδέα του user-level networking (Σχήμα 2.10). Οι προσαρμογείς δικτύου δίνουν τη δυαντότητα στις εφαρμογές να επικοινωνούν με 'εικονικά στιγμιότυπα' του προσαρμογέα (endpoints), ενώ αναλαμβάνουν αυτοί την απομόνωση μεταξύ των εφαρμογών.

Στα πιο διαδεδομένα δίκτυα υψηλής διασύνδεσης περιλαμβάνονται το Myrinet [9], και το πρωτόκολλο MX [10], το οποίο υλοποιεί ένα κατά βάση σύγχρονο send receive interface, και το Infiniband [11], το οποίο υλοποιεί ένα RDMA Interface.

Και τα δύο δίκτυα διασύνδεσης που αναφέρθηκαν, επιτυγχάνουν υψηλούς ρυθμούς μεταφοράς (10 Gbps), και χαμηλούς χρόνους απόρκισης (<5usec), απαιτούν όμως εξι-



Σχήμα 2.9: Xen Split Network Drivers

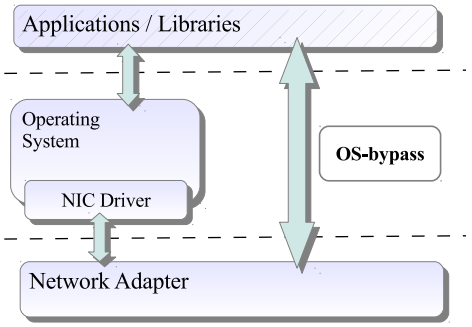
δεικευμένο υλικό, ενώ ανακύπτουν και θέματα compatibility, με πιο διαδεομένα physical και data link layer πρωτόκολλα.

2.2.1 10G Ethernet

Το Ethernet αποτελεί την πλέον διαδεομένη τεχνολογία διασύνδεσης υπολογιστών σε τοπικά δίκτυα. Με την προτυποποίηση του 10Gbps Ethernet, το Ethernet μπορεί να αποτελέσει ρεαλιστική λύση για να υποστηρίξει ένα δίκτυο διασύνδεσης υψηλής επίδοσης, λόγω τόσο της αύξηση του ρυθμού μεταφοράς, και της δημιουργίας μεταγωγών (switches) πολύ χαμηλού latency, όσο και της ευρείας χρήσης του Ethernet σε τοπικά δίκτυα και του χαμηλού κόστους υποδομής που απαιτεί, σε σχέση με εξιδεικευμένα δίκτυα υψηλής επίδοσης.

Ένας περιοριστικός παράγοντας στη χρήση του 10G Ethernet σε ένα δίκτυο διασύνδεσης υψηλής επίδοσης, είναι η στενή σύνδεσή του με πολύπλοκες στοίβες πρωτοκόλλων, όπως το TCP/IP. Λόγω της φύσης των δικτύων διασύνδεσης, πολλά από τα χαρακτηριστικά του TCP/IP, δεν είναι πλέον αναγκαία, όπως για παράδειγμα το byte-stream abstraction, ο έλεγχος συμφόρησης, και όλες οι δυνατότητες που προσφέρουν τα πρωτόκολλα δρομολόγησης του IP. Αντιθέτως, αυτά τα χαρακτηριστικά επιβαρύνουν την υλοποίηση του πρωτοκόλλου, και οδηγούν σε επιβάρυνση της γενικότερης απόδοσης του συστήματος.

Αντικαθιστώντας τα πρωτόκολλα αυτά, με lightweight πρωτόκολλα, είτε υλοποιώντας τα σε λογισμικό, είτε κάνοντας offload την υλοποίηση του πρωτοκόλλου σε έξυπνους προσαρμογείς δικτύου, έτσι ώστε να καλυφθεί η απαραίτητη λειτουργικότητα σε επίπεδο data link layer, δίνεται η δυνατότητα χρησιμοποίησης του 10G Ethernet για την δημιουργία ενός δικτύου διασύνδεσης υψηλής διασύνδεσης, επιτυγχάνοντας υψηλό throughput και χαμηλό latency.



Σχήμα 2.10: OS-bypass

2.2.2 Δίκτυα Διασύνδεσης Υψηλής Επίδοσης σε Εικονικά Περιβάλλοντα

Η αλματώδης ανάπτυξη τεχνικών εικονικοποίησης, ειδικά για την πλέον διαδεδομένη αρχιτεκτονική x86, σε συνδυασμό με την διάδοση των πολυπήρυνων επεξεργαστών (mutlicore CPUs), οδήγησε στην σταδιακή μετατροπή των κλασικών συστοιχιών υπολογιστών σε συστοιχίες εικονικών μηχανών.

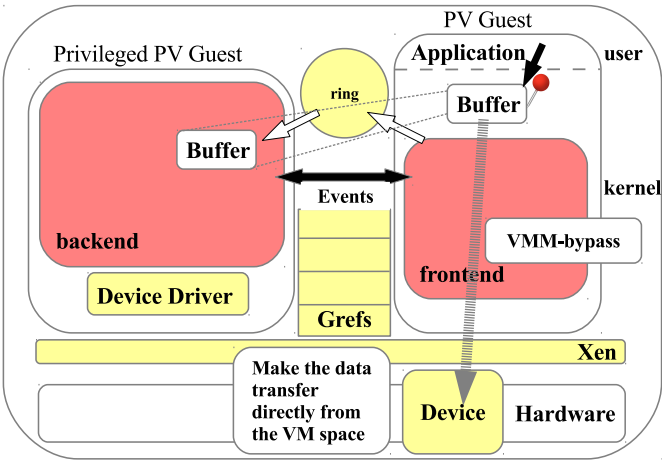
Η εικονικοποίηση προσφέρει μεγαλύτερη ευελιξία και απομόνωση στην εκτέλεση εφαρμογών σε τέτοιες συστοιχίες. Πολλαπλές εφαρμογές μπορούν να εκτελούνται σε απομονωμένες εικονικές μηχανές στον ίδιο φυσικό υπολογιστικό κόμβο, δίνοντας την δυανατότητα στους διαχειριστές των συστοιχιών για πιο ακριβή και λεπτομερή διαχείριση των πόρων των κόμβων. Σε αυτό συμβάλλει και η δυνατότητα της διαφανής μετακίνησης των εικονικών μηχανών μεταξύ των κόμβων, διευκολύνοντας επίσης την ανάκαμψη από αποτυχία, είτε υλικού είτε λογισμικού (failure recovery).

Δυστυχώς, όμως, η εικονικοποίηση επιβάλλει επιπλέον στρώματα αφαίρεσης (abstraction layers), τα οποία επιβαρύνουν την γενικότερη απόδοση του συστήματος. Σε ό,τι αφορά τους υπολογιστικούς πόρους (επεξεργαστής, μνήμη), έχουν αναπτυχθεί μηχανισμοί που αντιμετωπίζουν με επιτυχία την επιβάρυνση αυτή, και επιτυγχάνουν απόδοση παρόμοια με αυτή ενός μη-εικοινικοποιημένου συστήματος, και οι οποίοι αναφέρθηκαν νωρίτερα.

Σε αντίθεση με την διαχείρηση των υπολογιστικών πόρων, η λειτουργία Εισόδου / Εξόδου παραμένει προβληματική σε εικονικά περιβάλλοντα. Η ανάγκη για απομόνωση και ασφάλεια μεταξύ των εικοινικών μηχανών δεν επιτρέπει την απευθείας πρόσβαση των εικονικών μηχανών, και κατ' επέκταση των εφαρμογών στις συεκευές Εισόδου / Εξόδου, μειώνοντας έτσι κατά πολύ την απόδοση των λειτουργιών Εισόδου / Εξόδου, το οποίο τελικά επηρεάζει την συνολική απόδοση του συστήματος.

Ειδικότερα για τα δίκτυα διασύνδεσης υψηλής επίδοσης, όπως αναφέρθηκε, προκειμένουν να επιτύχουν υψηλούς ρυθμούς μεταφοράς (throughput), και χαμηλούς χρόνους πρώτης απόκρισης (latency), χρησιμοποιούν μηχανισμούς που παρακάμπτουν το λειτουργικό σύστημα (OS-bypass), και μηχανισμούς δικτύωσης στο χώρο χρήστη (user-level networking), οι οποίο όμως δεν μπορούν να λειτουργήσουν σε εικονικά περιβάλλοντα.

Προκύπτει λοιπόν η ανάγκη για ανάπτυξη νέων μηχανισμών, και ενσωμάτωσή τους στην υλοποίηση των πρωτοκόλλων και των δικτύων διασύνδεσης υψηλής επίδοσης, ώστε να παρακαμφθούν τα στρώματα αφαίρεσης που επιβάλλει το εικονικό περιβάλλον (VMM-bypass). [12] (Σχήμα 2.11)



Σχήμα 2.11: VMM-bypass

Έχουν προταθεί δύο βασικοί μηχανισμοί για την παράκαμψη των layers και του overhead, που επιβάλει η εικονικοποίηση, έτσι ώστε να υλοποιηθεί επιτυχώς ένα δίκτυο διασύνδεσης υψηλής επίδοσης σε εικονικό περιβάλλον.

Ο ένας μηχανισμός περιλαμβάνει τεχνικές υλοποιημένες σε υλικό, οι οποίες αναφέρθηκαν νωρίτερα (SR/MR-IOV). Στην περίπτωση αυτή, τίθενται θέματα κλιμάκωσης, λόγω περιορισμών του υλικού, καθώς και επιπλέον απαιτήσεις από το υλικό, έτσι ώστε να επιτυγχάνεται η απομόνωση μεταξύ των εικονικών μηχανών (IOMMU). Οι μηχανισμοί αυτοί δεν εγγυώνται όμως και την παράκμψη του πυρήνα του guest, γεγονός το οποίο επηρεάζει την απόδοση.

Η δεύτερη λύση χρησιμοποιεί το Para Virtualization, και το μοντέλο διαχωρισμένου οδηγού. Το πάνω μέρος του οδηγού παρουσιάζει στις εφαρμογές HPC interconnect semantics, και επικοινωνεί με το backend για την αρχικοποίηση της επικοινωνίας των εφαρμογών με τους προσαρμογείς δικτύου. Οι μεταφορές δεδοεμένων μεταξύ των εφαρμογών και τον προσαρμογέων δικτύου παρακάμπτουν τόσο τον πυρήνα του guest, όσο και τον hypervisor, προσφέροντας ένα άμεσο μονοπάτι δεδομένων των εφαρμογών προς το δίκτυο.

Κεφάλαιο 3

Previous Work

3.1 SLURPoE

Στα πλαίσια της διπλωματική εργασίας "Σχεδίαση και Υλοποίηση μηχανισμού απευθείας απομακρυσμένης πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου προσαρμογέα δικτύου 10GbE"[13] υλοποιήθηκε ένα δίκτυο διασύνδεσης υψηλής επίδοσης βασισμένο στο 10G Ethernet.

Για τις ανάγκες του δικτύου διασύνδεσης, σχεδιάστηκε και υλοποιήθηκε το πρωτόκολλο SLURPoE (Simple User-Level RDMA Protocol over Ethernet), ένα lightweight πρωτόκολλο απομακρυσμένης απευθείας πρόσβασης στη μνήμη.

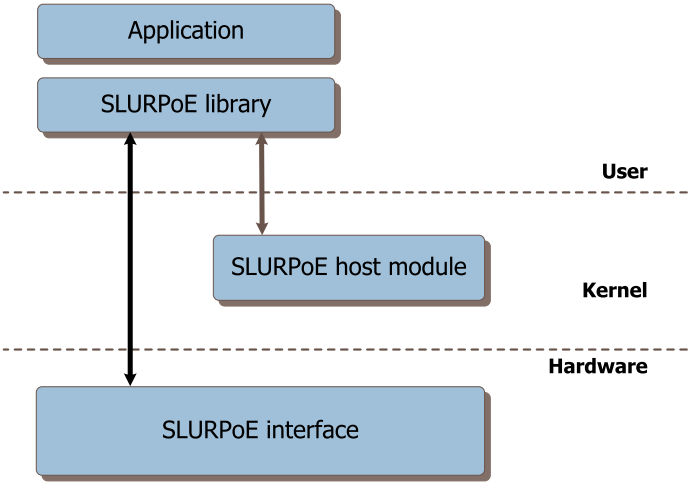
Το SLURPoE υλοποιεί μηχανισμούς παράκαμψης του λειτουργικού συστήματος, και των επιβαρύνσεων που αυτό επιφέρει στην επίδοση του δικτύου. Συγκεκριμένα, χρησιμοποιεί μηχανισμούς μηδενικών αντιγράφων, όπου οι μεταφορές δεδομένων γίνονται με λειτουργίες απευθείας πρόσβασης στη μνήμη (DMA), από περιοχές μνήμης των εφαρμογών, οι οποίες έχουν καθηλωθεί στη μνήμη (pinned-down buffers).

Προκειμένου να αποφευχθεί η επιβάρυνση των επεξεργαστών των κόμβων από το overhead της υλοποιησης και επεξεργασίας του πρωτοκόλλου, χρησιμοποιείται ένας έξυπνος προσαρμογέας δικτύου, ο οποίος υλοποιεί το πρωτόκολλο (SLURPoE firmware, και αναλμβάνει τον προγραμματισμό των μηχανών DMA (μεταφορά δεδομένων), την απομώνση και την ασφαλή πρόσβαση των εφαρμογών στον προσαρμογέα, και την αποστολή των πλαισίων Ethernet στο δίκτυο.

Η σχεδίαση του SLURPoE (σχήμα 3.1) προσφέρει ένα άμεσο μονποπάτι δεδομένων από τις εφαρμογές στον προσαρμογέα δικτύου, χρησιμοποιώντας τον πυρήνα του λειτουργικού συστήματος των κόμβων μόνο για λειτουργίες ελέγχου και αρχικοποίησης της επικοινωνίας μεταξύ των εφαρμογών και του προσαρμογέα δικτύου (SLURPoE host module).

Μια βιβλιοθήκη χώρου χρήστη (SLURPoE library) ορίζει την διεπαφή την οποία χρησιμοποιούν οι εφαρμογές για την μεταξύ τους επικοινωνία, αποκρύπτωντας τις λεπτο-

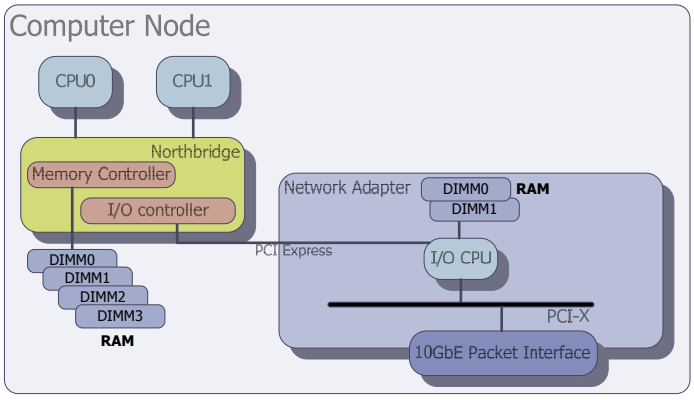
μέρειες της επικονωνίας της βιβλιοθήκης χώρου χρήστη με τον έξυπνο προσαρμογέα δικτύου.



Σχήμα 3.1: SLURPoE Design

Ο έξυπνος προσαρμογέας δικτύου, που χρησιμοποιήθηκε για την υλοποίηση του πρωτοκόλλου (σχήμα 3.2, αποτελείται από ένα I/O ARM XScale board της Intel (81342). Το board επικοινωνεί με τον host μέσω του διαύλου PCI-E, ενώ στον δίαυλο PCI-X τοποθετείται ένας 10G Ethernet προσαρμογές δικτόου.

Το board διαθέτει 3 μηχανές DMA, για την μεταφορά δεδομένων από και πρός τον host, ενώ διαθέτει και έναν μηχανισμό για εκατέρωθεν ειδοποιήσεις με διακοπές (Messaging Unit), και ο οποίος χρησιμοποιείται για την ανταλλαγή μηνυμάτων ελέγχου. Τα μηνύματα που ανταλάσσονται χρησιμοποιούν δομές με μορφή κυκλικών ουρών, μία για κάθε κατεύθυνση.



Σχήμα 3.2: SLURPoE Architecture

Στην φάση της αρχικοποίησης, η βιβλιοθήκη χώρου χρήστη, και ο πυρήνας του λειτουργικού συστήματος του host, συνεργάζονται, προκειμένου να αρχικοποιηθεί ο έξυπνος προσαρμογέα δικτύου, και να γίνουν map οι κυκλικές ουρές του Messaging Unit στον χώρο διευθύνσεων της βιβλιοθήκης χρήστη. Από κει και πέρα, η βιβλιοθήκη, και κατ' επέκταση οι εφαρμογές, μπορούν να επικοινωνούν απευθείας με τον προσαρμογέα, χωρίς παρέμβαση του λειτουργικού συστήματος του host.

Οι βασικές λειτουργίες που ορίζει το πρωτόκολλο SLURPoE είναι:

- Open / Close Endpoint

  Ακολουθώντας μια προσέγγιση user-level networking, παρουσιάζεται σε κάθε εφαρμογή ένα εικονικό στιγμιότυπο endpoint του προσαρμογέα δικτύου, το οποίο χρησιμοποιούν άμεσα οι εφαρμογές. Στα πλαίσια της αρχικοποίησης της επικοινωνίας, κάθε εφαρμογή πρέπει να καλέσει την συνάρτηση $\mathrm{slurpoe\_open\_endpoint}()$, προκειμένου να ενημερωθεί το firmware του προσαρμογέα και να δεσμευτούν και να αρχικοποιηθούν οι κατάλληλες δομές (struct slurpoe_endpoint).

  Κατά τον τερματισμό της επικονωνίας, καλείται η συνάρτηση $\mathrm{slurpoe\_close\_endpoint}()$, για την αποδέσμευση του endpoint.

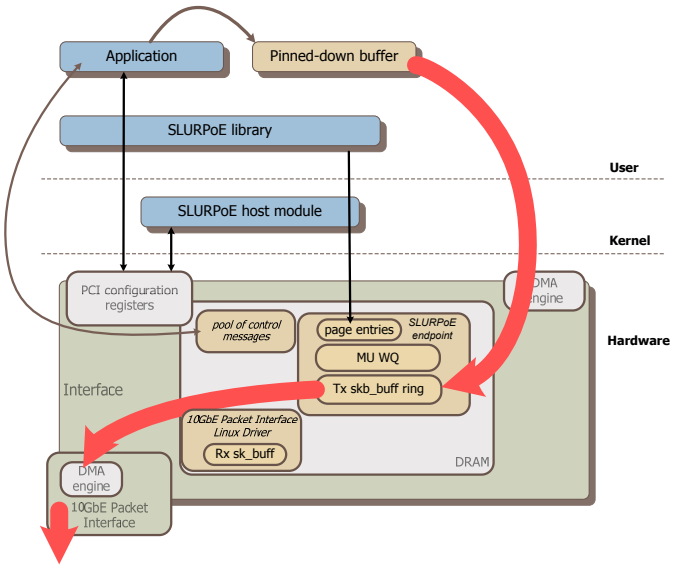
- Register / Deregister Region

  Προτού χρησιμοποιηθούν λειτουργίες RDMA, οι εφαρμογές πρέπει να δεσμεύσουν περιοχές μνήμες οι οποίες θα χρησιμοποιηθούν για τη μεταφορά των δεδομένων. Οι περιοχές αυτές της μνήμης θα πρέπει να καθηλωθούν στην μνήμη του κόμβου (pinned-down buffers), καθώς η μεταφορά των δεδομέων γίνεται με λειτουργίες DMA, με ευθύνη του προσραμογέα. Προκειμένου να αποφευχθεί η μεταφορά των σελίδων αυτων της μνήμης (pages) στον δίσκο (swap), οι περιοχές αυτές της μνήμης πρέπει να δηλωθούν με συγκεκριμένο τρόπο (να καθηλωθούν). Επίσης, οι διευθύνσεις των περιοχών αυτών, μαζί με το αντίστοιχο endpoint, στένονται στον έξυπνο προσαρμογέα, ο οποίος διατηρεί ένα software IOTLB, υλοποιώντας με αυτόν τον τρόπο έναν μηχανισμό μετάφρασης διευθύνσεων αλλά και απομόνωσης μεταξύ των διαφορετικών εφαρμογών και endpoints.

- RDMA Read / Write

  Αφού αρχικοποιηθούν τα endpoints, και δεσμευτούν οι περιοχές μνήμης, οι εφαρμογές μπορούν να επικοινωνήσουν μεταξύ τους με λειτουργίες RDMA. Η βιβλιοθήκη ειδοποιεί τον προσαρμογέα μέσω των κυκλικων ουρών του Messaging Unit, και ο προσαρμογέας με τη σειρά του, προγραμματίζει κατάλληλα τις μηχανές DMA. Μόλις ολοκληρωθεί η μεταφορά των δεδομένων από τον κόμβο σε buffers του προσαρμογέα, δημιουργείται ο κατάλληλος socket buffer (skb), και το πλαίσιο Ethernet, τα οποία στη συνέχεια αποστέλονται στο δίκτυο.

Το μονοπάτι δεδομένων σε μια λειτουργία RDMA Write φαίνεται στα σχήματα 3.3 και 3.4, για την αποστολή και λήψη δεδομένων αντίστοιχα.



Σχήμα 3.3: SLURPoE Send Path

3.2 SLURPoE-XEN

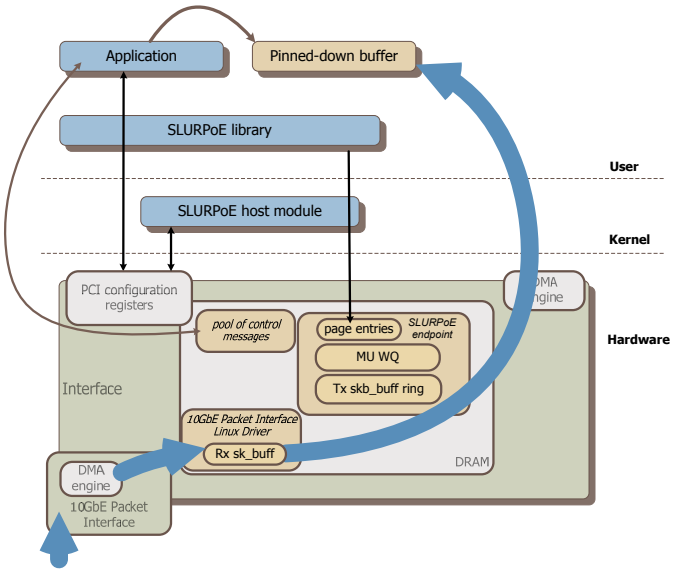
Στην διπλωματική εργασία ``Ένταξη Σημασιολογίας Δικτύων Διασύνδεσης Υψηλής Επίδοσης σε Εικονικά Περιβάλλοντα''[14], τροποποιήθηκε η υλοποίηση του πρωτοκόλλου SLURPoE, έτσι ώστε να είναι δυνατή η χρησιμοποίηση του δικτύου διασύνδεσης από εφαρμογές που εκτελούνται σε εικονικά περιβάλλοντα.

Η υλοποίηση έγινε για το εικονικό περιβάλλον του ελεγκτή εικονικών μηχανών Xen, χρησιμοποιώντας το μοντέλο διαχωρισμένου οδηγού split driver model. Η βιβλιοθήκη χώρου χρήστη, καθώς και το firmware του έξυπνου προσαρμογέα δικτύου διατηρήθηκαν χωρίς αλλαγές. Το SLURPoE host module, το οποίο εκτελούνταν στον πυρήνα του λειτουργικού συστήματος του κόμβου, χωρίστηκε σε δύο μέρη.

Το πάνω μέρος του οδηγού (frontend), το οποίο εκτελείται στον πυρήνα των DomUs, αναλαμβάνει την επικοινωνία με την βιβλιοθήκη και τις εφαρμογές χώρου χρήστη, και μεταφέρει τις αιτήσεις των εφαρμογών στο κάτω μέρος του οδηγού. Επίσης, κρατάει στη μνήμη δομές μοναδικές για κάθε εφαρμογή, όπως για παράδειγμα τις δομές που περιγράφουν τα endpoints.

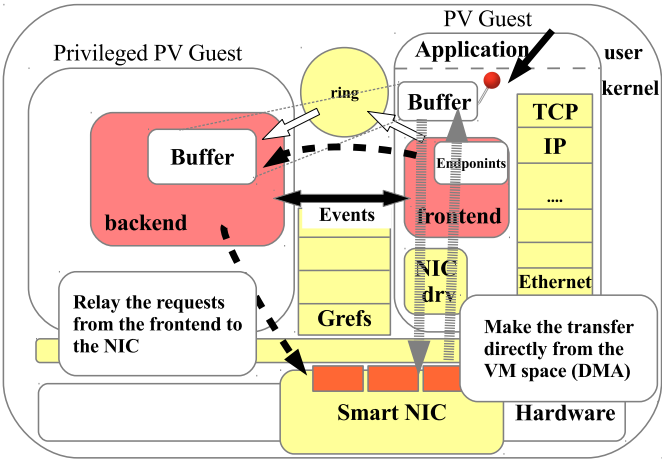
Το κάτω μέρος του οδηγού (backend), το οποίο εκτελείται στον πυρήνα του privileged guest (Dom0) του Xen, αναλαμβάνει την πολύπλεξη των αιτήσεων από τους frontend drivers των εικονικών μηχανών, και την μεταφορά τους στον έξυπνο προσαρμογέα δικτύου.

Καθώς οι unprivileged guests δεν έχουν άμεση πρόσβαση στο υλικό του προσαρμογέα (και συγκερκιμένα στο Messaging Unit, όλες οι λειτουργίες ελέγχου πρέπει να



Σχήμα 3.4: SLURPoE Receive Path

περάσουν από το Dom0, ακόμα και στη περίπτωση αίτησης για RDMA λειτουργίες. Η μεταφορά δεδομένων παρακάμπτει τόσο τον Xen hypervisor (VMM-bypass), όσο και τον πυρήνα του λειτουργικού των εικονικών μηχανών (σχήμα 3.5). Η μεταφορά δεδεομένων γίνεται μέσω των μηχανών DMA του προσαρμογέα, όμως το γεγονός ότι η αίτηση για εγγραφή ή ανάγνωση RDMA περνάει μέσα από το Dom0, επιφέρει μια σταθερή επιβάρυνση στην απόδοση σε σχέση με την native υλοποίηση.

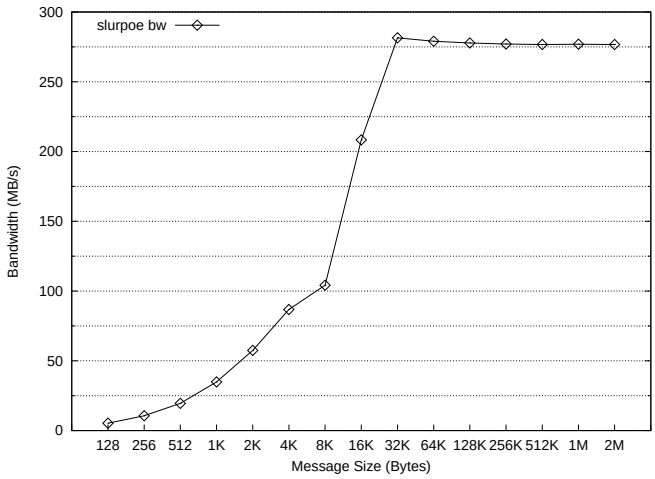


Σχήμα 3.5: SLURPoE-XEN Datapath

Κεφάλαιο 4

SLURPoE Perfomance Evaluation

Όπως φάνηκε και από τα διαγράμματα και τις μετρήσεις που παρουσιάστηκαν στην προηγούμενη ενότητα, το δίκτυο διασύνδεσης και η υλοποίηση του πρωτοκόλλου SLURPoE εμφανίζει suboptimal απόδοση.

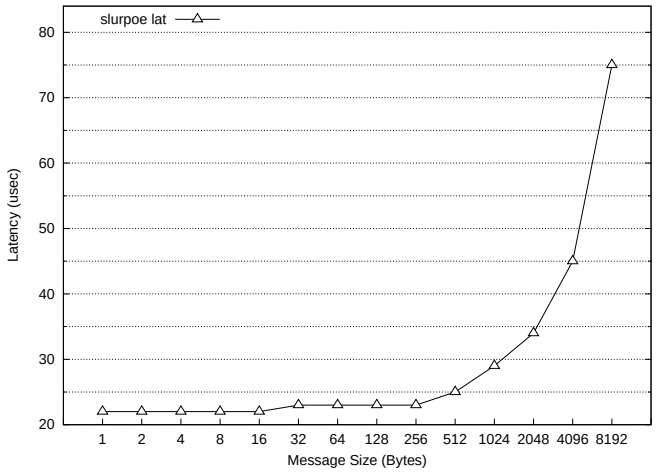


Σχήμα 4.1: SLURPoE End-to-end bandwidth

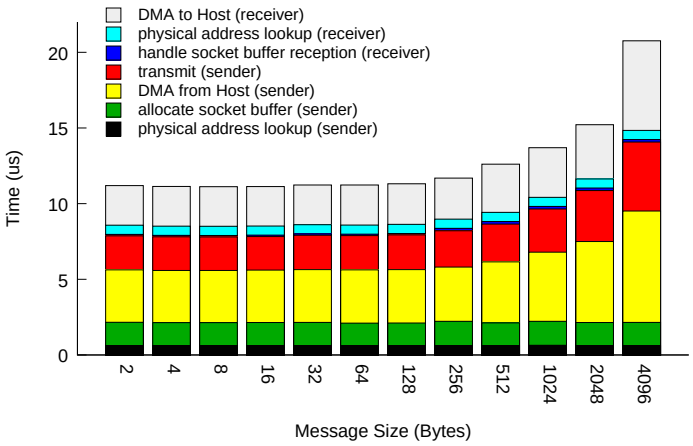
To end-to-end throughput (σχήμα 4.1) περιορίζεται στο 22% του 10G Ethernet line rate, για μεγάλα μηνύματα (>32KB), ενώ το latency παραμένει στα 22usecs (σχήμα 4.2), ακόμα και για μικρά μηνύματα (<512 Bytes).

Οι περιορισμοί αυτοί στην απόδοση του δικτύου διασύνδεσης έχουν άμεση επίδραση και στην απόδοση του εικονικοποιημένου πρωτοκόλλου SLURPoE-XEN, το οποίο παρουσιάζει μεγαλύτερη επιβάρυνση στην απόδοσή του λόγω του virtualization overhead (σχήματα 4.4 και 4.5).

Στην ενότητα αυτή παρουσιάζεται η προσπάθεια που έγινε να εντοπιστούν σημεία συμφόρησης και περιορισμού της απόδοσης (bottlenecks), του δικτύου διασύνδεσης,



Σχήμα 4.2: SLURPoE One-way latency



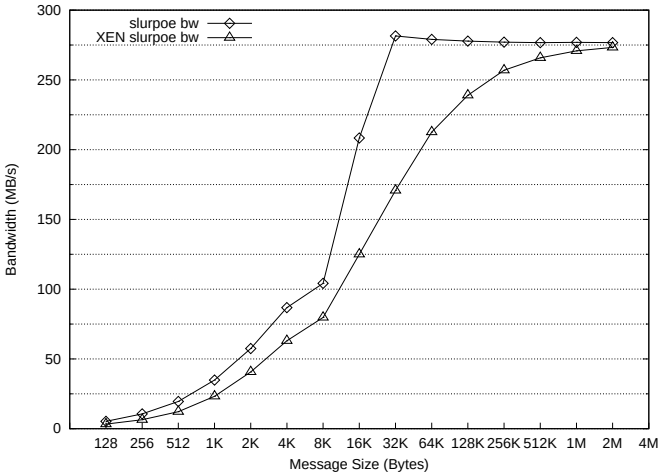
Σχήμα 4.3: SLURPoE latency breakdown

τόσο σε επίπεδο υλικού, όσο και επίπεδο υλοποίησης πρωτοκόλλου.

Τα πιθανά bottlenecks μπορούν να χωριστούν σε δύο βασικές κατηγορίες:

- Περιορισμοί λόγω υλικού του έξυπνου προσαρμογέα δικτύου (network adapter, memory bus / PCI-E / PCI-X throughput, CPU perfomance)
- Καλύτερο pipelining, και επικάλυψη των διάφορων λειτουργιών, ώστε να αποκρυφθεί το κόστος χρονοβόρων λειτουργιών

Από τις αρχικές μετρήσεις, φάνηκε ότι κατά τη διάρκεια πολλαπλών αποστολών και λήψεων, η συνολική χρησιμοποίηση του επεξεργαστή του έξυπνου προσαρμογέα δικτύου δνε ξεπέρασε το 40%.



Σχήμα 4.4: SLURPoE-XEN End-to-end bandwidth

Οι μετρήσεις και η ανάλυση της απόδοσης εστιάζεται στον 10G Ethernet adapter, και στο δίαυλο PCI-X του board, καθώς και στις λειτουργίες DMA, τον προγραμματισμό των αντίστοιχων μηχανών, και το throughput του memory bus του προσαρμογέα.

4.1 DMA

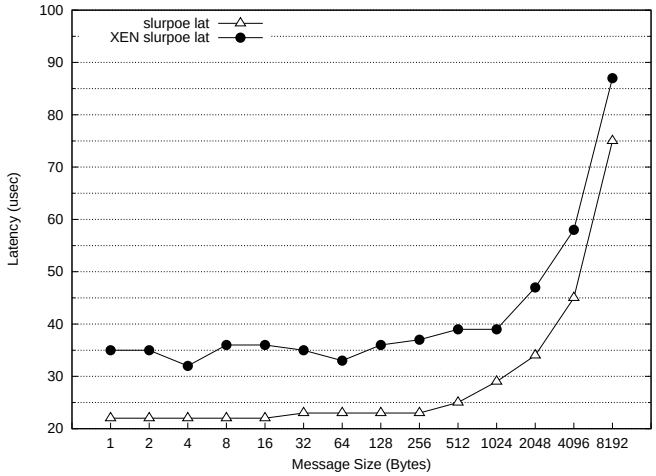
Σημαντικό ρόλο στην απόδοση του δικτύου διασύνδεσης παίζουν οι λειτουργίες DMA καθώς και η ικανότητα του memory bus του board να ανταπεξέλθει στο traffic που δημιουργείται.

Συγκεκριμένα, ο προγραμματισμός των μηχανών DMA είναι μια χρονοβόρα λειτουργία, η οποία αν δεν υλοποιηθεί σωστά, οδηγεί σε αναποτελεσματικό pipeline των μεταφορών δεδομένων.

Το board διαθέτει 3 μηχανές DMA, για τις οποίες υπάρχει οδηγός στον πυρήνα του Linux, ο οποίος τρέχει στον έξυπνο προσαρμογέα δικτύου (drivers/dma/iop-adma¬ .c). Για τον προγραμματισμό τους όμως, χρησιμοποιείται το πιο γενικό dmaengine API drivers/dma/dmaengine.c.

Το πρόβλημα με τη χρήση του dmaengine ήταν το γεγονός ότι για κάθε μεταφορά, έπρεπε να γίνει επαναπρογραμματισμός της μηχανής, γεγονός που επιβάρυνε την απόδοση.

Στην παρούσα διπλωματική, τροποποιήσαμε τον προγραμματισμό των μηχανών DMA, έτσι ώστε να χρησιμοποιεί το async-tx API. Η προγραμματιστική αυτή διεπαφή επιτρέπει την δημιουργία αλυσίδων με ασύγγχρονες μεταφορές δεδομένων με χρήση των μηχανών DMA. Η τροποποίηση αυτή οδηγεί σε καλύτερο pipelining των μεταφορών δεδομένων, και αύξηση του throughput σε ό,τι αφορά τις μεταφορές δεδομένων.



Σχήμα 4.5: SLURPoE-XEN One-way latency

Το πλεονέκτημα της προγραμματιστκής αυτής διεπαφής είναι το γεγονός ότι επιτρέπει να γίνουν chain πολλαπλές μεταφορές δεδομένων, ενώ ο προγραμματισμός της μηχανής DMA συμβαίνει μόνο μία φορά, όταν καλείται η συνάρτηση $\mathrm{dma\_issue\_pending\_all()}$. Το board μπορεί να κάνει poll στην συνάρτηση $\mathrm{dma\_wait\_for\_async\_tx()}$ για να ενημερωθεί για την ολκλήρωση των μεταφορών. Εναλλακτικα, δίνεται η δυνατότητα να οριστεί callback function, το οποίο καλείται μόλις ολοκληρωθεί η μεταφορά, ασύγχρονα, με τη βοήθεια διακοπών.

Με απλά benchmarks, τα οποία χρησιμοποιούσαν τις μηχανές DMA για να μεταφέρουν δεδομένα από και προς το board, μετρήθηκε το throughput του PCI-E host-to-bus interface, αλλά και το memory bus του board, καθώς και η αποδοτικότητα του προγραμματισμού των μηχανών DMA.

Το host-to-board throughput έφτασε τα 800Mib/sec, χρησιμοποιώντας DMA chained transcations, ενώ το συνολικό throughput του δικτύου διασύνδεσης δεν παρουσίασε σημαντική βελτίωση.

4.2 10GbE NIC

Αφού αποδείχθηκε, ότι ούτε το throughput του memory bus, ούτε ο προγραμματισμός των μηχανών DMA αποτελεί περιοριστικό παράγοντα στην απόδοση του δικτύου διασύνδεσης, στραφήκαμε στον 10G Ethernet adapter, ο οποίος συνδέεται στον PCI-X δίαυλο του board.

Για την μελέτη της απόδοσης του 10G Ethernet adapter χρησιμοποιούμε jumbo Ethernet frames (9000 bytes), και κατάλληλες ρυθμίσεις στον οδηγό του προσαρμογέα.

Στη συνέχεια, με τη βοήθεια μετροπρογραμμάτων (benchmakrs) αναλύουμε το μέγιστο throughput που μπορούμε να πετύχουμε με τον συγκεριμένο 10GbE προσαρμο-

γέα, πάνω σε δίαυλο PCI-X.

Αρχικά, εκτελούμε το δικτυακό benchmark netperf. Το netperf δίνει την δυνατότητα εκτέλεσης διαφορετικών δικτυακών benchmarks. Εμείς χρησιμοποιήσαμε το UDP_STREAM benchmark το οποίο μετράει το end-to-end throughput, χρησιμοποιώντας conetcionless UDP traffic μεταξύ δύο hosts.

Τα αποτελέσματα του UDP STREAM έδειξαν ότι το throughput ήταν περιορισμένα στα 280MiB/sec, με 100% χρησιμοποίηση του επεξεργαστή του board, δηλαδή το throughput ήταν περιορισμένο λόγω ανικανότητα του ARM I/O επεξεργαστή να ανταπεξέλθει στη δημιουργία και αποστολή UDP datagrams.

Στη συνέχεια, υλοποιήθηκε σε επίπεδο πυρήνα ένα netperf-like benchmark, το slurpoe_netperf.

Το slurpoe_netperf δεσμέυει αρχικά ένα pool από skbs (socket buffers), έτσι ώστε κατά την διάρκεια του benchmark ο επεξεργαστής να μην επιφορτίζεται με την διαδικασία δέσμευσης και δημιουργίας socket buffers, ώστε το throughput να μην περιορίζεται από την επεξεργαστική ισχύ του board.

Στη συνέχεια, χρησιμοποιώντας την συνάρτηση πυρήνα $\mathrm{dev\_queue\_xmit()}$, στέλνουμε τους socket buffers / Ethernet frames, στο δίκτυο, και μετράμε το end-to-end throughput. Τα αποτελέσματα των μετρήσεων έδειξαν ότι το μέγιστο συνολικό throughput που μπορούμε να πετύχουμε με τον συγκεκριμένο PCI-X 10G Ethernet adapter, είναι 550Mib/sec, 46% του 10GbE line rate, ενώ το CPU utilization στο board δεν ξερπέρασε το 1% κατά τη διάρκεια του benchmark.

4.3 Σύνοψη

Από τις παραπάνω μετρήσεις, συμπεραίνουμε ότι το αδύναμο σημείο είναι ο προσαρμογέας δικτύο, ο οποίος δεν μπορεί να υποστηρίξει 10G traffic. Παρόλα αυτά, το throughput που μπορεί να πετύχει ο προσαρμογέας, είναι περίπου δύο φορές μεγαλύτερο από το throughput του δικτύου διασύνδεσης.

Ένας άλλος περιοριστικός παράγοντας, ο οποίος δεν εξετάζεται στην παρούσα διπλωματική, είναι το latency και το κόστος της επικοινωνίας των δύο πλευρών μέσω του Messaging Unit, καθώς και ζητήματα γενικότερης υλοποίησης και βελτιστοποίησης του πρωτοκόλλου, κυρίως σε επίπεδο pipelining λειτουργιών εκτός των μεταφορών DMA.

Συμπερασματικά, ενώ το υλικό του προσαρμογέα θέτει κάποια άνω όρια στην απόδοση του πρωτοκόλλου, δεν είναι ο καταλυτικός παράγοντας για την μειωμένη απόδοση του δικτύου διασύνδεσης. Στην επόμενη ενότητα, θα εξετάσουμε μια διαφορετική προσέγγιση, η οποία παρακάμπτει τόσο τους περιορισμούς σε επίπεδο υλικού, αλλά και την ενδεχόμενη προβληματική και αργή επικοινωνία μεταξύ του host και του board.

Κεφάλαιο 5

Kernel-level SLURPoE-XEN

Έχοντας υπόψιν τα αποτελέσματα του προηγούμενου κεφαλαίου, ακολουθούμε μια διαφορετική προσέγγιση, και υλοποιούμε το εικονικοποιημένο πρωτόκολλο SLURPoE-XEN σε επίπεδο πυρήνα.

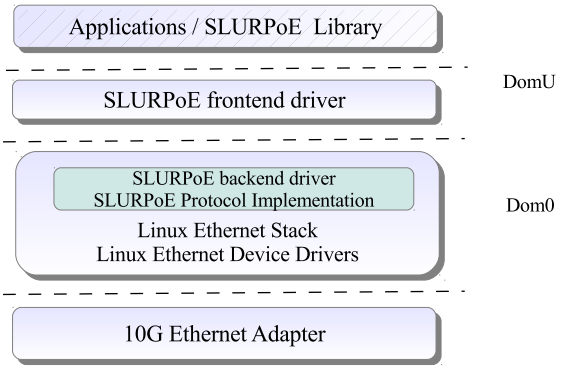
Στόχος είναι η μελέτη της συπμεριφοράς μια υλοποίησης σε επίπεδο πυρήνα, σε σχέση με την υλοποίηση του πρωτοκόλλου σε έναν έξυπνο προσαρμογέα δικτύου. Με αυτό τον τρόπο, θα μπορέσουμε απο τη μία να προσδιορίσουμε με μεγαλύτερη σαφήνεια τους περιοριστικούς παράγοντες της απόδοσης του δικτύου διασύνδεσης, δηλαδή σε ποιο βαθμό ευθύνεται το υλικό του προσαρμογέα 10G Ethernet, η επικοινωνία με τον έξυπνο προσαμογέα δικτύου, και η υλοποίηση του πρωτοκόλλου γενικότερα, και από την άλλη να συγκρίνουμε τις δύο υλοποιήσεις τόσο σε επίπεδο throughput και latency όσο και σε ό,τι αφορά το συνολικό CPU utilization στους κόμβους.

Η υλοποίηση του πρωτοκόλλου σε έναν έξυπνο προσαρμογέα δικτύου έχει στόχο την όσο τον δυνατόν μικρότερη επιβάρυνση των επεξεργαστών των κόμβων, αλλά με tradeoffs σε ό,τι αφορά το latency, λόγω του κόστους της επικονιωνίας μεταξύ host και board.

Από την άλλη, μια υλοποίηση σε επίπεδο πυρήνα, περιλαμβάνει λιγότερα layers, και η επικοινωνία της εφαρμογής με τον προσαρμογεά είναι πιο άμεση, οδηγώντας σε μικρότερο latency, και καλύτερο throughput. Το tradeoff σε αυτή την περίπτωση, είναι η επιβάρυνση των επεξεργαστών των κόμβων με το κόστος υλοποίησης και επεξεργασίας του πρωτοκόλλου.

5.1 Σχεδιασμός και Υλοποίηση

Η υλοποίηση του πρωτοκόλλου σε επίπεδο πυρήνα ουσιαστικά περιλαμβάνει την μεταφορά (port) του firmware, το οποίο μέχρι πρότινος, εκτελούνταν στον έξυπνο προσαρμογέα δικτύου (Intel Xscale ARM board), στον πυρήνα του privileged guest του Xen, και συγκεκριμένα στον ήδη υπάρχοντα backend driver. (σχήμα 5.1)



Σχήμα 5.1: Kernel-level SLURPoE-XEN Design

Παρόλο που η υλοποίηση του firmware του board είχε γίνει ως module στον πυρήνα του Linux (Linux ARM port), η μεταφορά του στον πυρήνα του Dom0 περιελάμβανε αρκετές αλλαγές, και λεπτά σημεία που έπρεπε να προσεχθούν ιδιαίτερα.

Καθώς αφαιρείται ο έξυπνος προσαρμογέας, ο κώδικας που αναλάμβανε την αρχικοποίηση του έξυπνου προσαρμογέα μέσω ενός PCI driver, καθώς και ο κώδικας χειρισμού του Messaging Unit αφαιρούνται τελειώς από τον backend, και πλέον, χρησιμοποιείται απευθείας το Ethernet stack και οι 10G Ethernet drivers του Linux.

Ο κώδικας αρχικοποίησης του έξυπνου προσαρμογέα δικτύου (του PCI interface / Messaging Unit αντικαθίσταται από κώδικα του networking stack του Linux, ο οποίος αναλαμβάνει να κάνει bind στους υπάρχοντες Ethernet adapters, και αναλαμβάνει την δέσμευση και δημιουργία των socket buffers / Ethernet frames, καθώς και όλες τις απαραίτητες λειτουργίες για την αποστολή και λήψη πακέτων SLURPoE (slurpoe_¬ pkt).

Στο σημείο αυτό, αξίζει να σημειωθεί ότι σε αντίθεση με τον κώδικα του firmware, υλοποιήθηκε υποστήριξη για περισσότερους από έναν προσαρμογείς δικτύου, με ξεχωριστά endpoints για κάθε προσαρμογέα.

Η μεταφορά της υλοποίησης του πρωτοκόλλου στον πυρήνα του host, οδήγησε σε αποπλοίηση του χειρισμού της κατάστασης των endpoints, καθώς πλέον ο backend driver του host έχει άμεση γνώση της κατάστασης των endpoints, και δεν χρειάζεται να περιμένει επιβεβαίωση από τον προσαρμογέα δικτύου για την κατάσταση του endpoint.

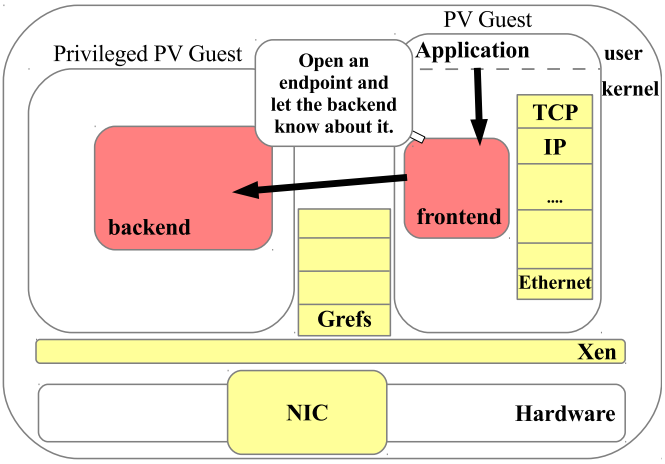
Όπως αναφέρθηκε προηγουμένως, ο έξυπνος προσαρμογέας δικτύου διατηρούσε μια δομή IOTLB, προκειμένου να επιτευχθεί η μετάφραση των διευθύνσεων και η απομόνωση μεταξύ των διαφορετικών εφαρμογών / endpoints. Η δομή αυτή μεταφέρεται πλέον στον πυρήνα του host. Το ζήτημα που προέκυψε κατά τη μεταφορά του IOTLB είχε να κάνει με αρχιτεκτονικές διαφορές μεταξύ του host και του board. Συγκεκρι-

μένα, το ARM Xscale board είχε 32bit physical addresses, ενώ ο AMD64 host 64bit physical addresses. Το γεγονός αυτό, εμπόδιζε ορισμένες από τις συναρτήσεις χειρισμού του IOTLB (lookup) να λειτουργήσουν σωστά, και έπρεπε να τροποποιηθούν ώστε να λειτουργούν σωστά σε 64bit αρχιτεκτονικές.

Τόσο στο send path όσο και στο receive path, προέκυψαν ορισμένα ζητήματα, σε ό,τι αφορά κυρίως τις μεταφορές δεδομένων. Καθώς πλέον δεν υπάρχει ο έξυπνος προσαρμογέας δικτύου, ο οποίος διέθετε μηχανές DMA, τις οποίες προγραμμάτιζε για την μεταφορά δεδομένων, έπρεπε να βρεθεί αποδοτικό τρόπος, χωρίς τη χρήση αντιγράφων (zero copy), για να γίνεται η αποστολή και η λήψη δεδομένων.

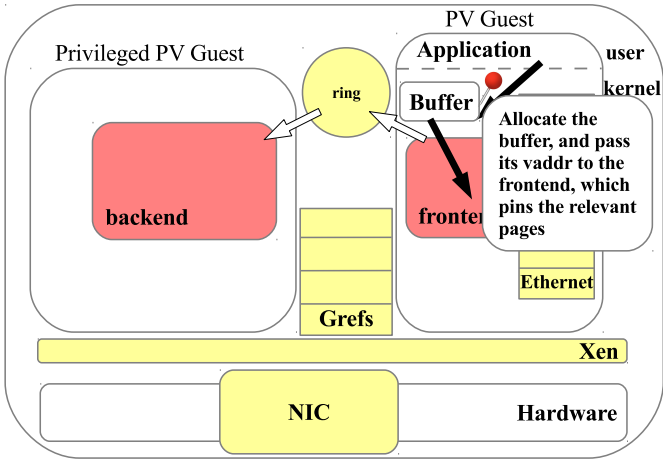
Στο send path, η αποστολή δεδομένων γίνεται χωρίς τη δημιουργία αντιγράφων. Χρησιμοποιώντας τις δυνατότητες που προσφέρει το networking API του Linux, ο Dom0 δημιουργεί τον socket buffer, και σε αυτόν ενσωματώνει άμεσα τις σελίδες που περιέχουν τα δεδομένα που πρέπει να αποσταλούν. Έτσι υλοποιείται ένα zero-copy send path.

Στο recive path, καθώς πλέον δεν υπάρχει ένας έξυπνος προσαρμογέας δικτύου, ήταν δύσκολο να αποφύγουμε τουλάχιστον ένα αντίγραφο. Τα δεδομένα πρέπει να αντιγραφούν στον staging χώρο του networking stack του Linux, και από κει να μεταφερθούν στους buffers που έχει δεσμεύσει η εφαρμογή. Αυτό μπορεί να γίνει είτε με memcpy, επιβαρύνοντας τους επεξεργαστές τον κόμβων, είτε χρησιμοποιώντας τις μηχανές DMA που προσφέρει η Intel στις μητρικές της. (I/OAT DMA)



Σχήμα 5.2: Kernel-level SLURPoE-XEN - open endpoint

Η λειτουργία του πρωτοκόλλου, με την υλοποίηση σε χώρο πυρήνα, φαίνεται στα σχήματα 5.2 5.3 5.4 5.5 5.6



Σχήμα 5.3: Kernel-level SLURPoE-XEN - register memory

5.2 Αξιολόγηση

Για την αξιολόγηση της απόδοσης της υλοποίησης του πρωτοκόλλου σε επίπεδο πυρήνα χρησιμοποιήθηκε ουσιαστικά το ίδιο micro-benchmark με αυτό που χρησιμοποιήθηκε για την αξιολόγηση του SLURPoE-XEN.

Για το benchmark, χρησιμοποιήθηκαν 2 Quad Core Intel Xen 2.4GHz, με Intel 5500 Chipset, 4GB μνήμη, και 2 PCIe-4x Myricom 10G-PCIE-8A 10GbE NICs συνδεδεμένες back to back.

Επίσης χρησιμοποιήθηκε η έκδοση 4.1-unstable του Xen και ο πυρήνας 2.6.32-24-pvops του Linux. Σε κάθε VM παραχωρήθηκε 1GB μνήμη και 1 physical core, και 2GB και 2 cores στον privileged guest,

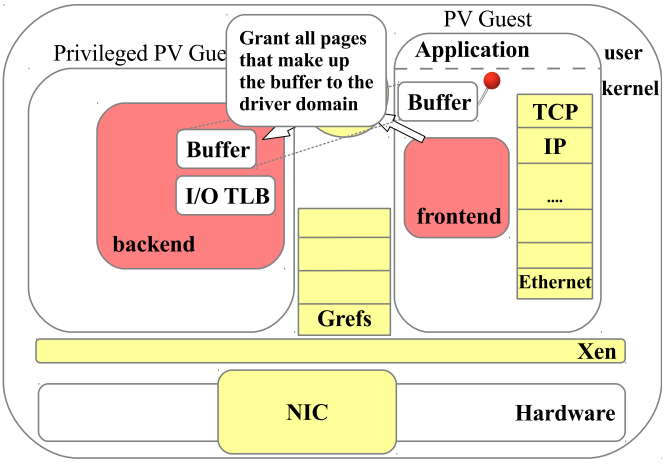
Τέλος, το networking stack του Linux ρυθμίστηκε ώστε να στέλνει Jumbo Ethernet frames (9000 bytes), ενώ προκειμένου να γίνει αποδοτικότερος ο χειρισμός των interrupts από τους προσαρμογείς δικτύου, ρυθμίστηκε το affinity των interrupts ώστε να παραδίδονται μόνο στα 2 physical cores του privileged guest.

Τα στατιστικά που αφορούν στο CPU utilization προήλθαν από τις πληροφορίες που περιέχονται στο αρχείο /proc/stats.

Αρχικά, χρησιμοποιώντας το kernel module pktgen, μετρήθηκε το baseline throughput των προσαρμογέων δικτύου για native Linux, για το Xen Driver Domain, και για το Xen VM.

Όπως φαίνεται στο σχήμα 5.7, ο privileged guest μπορεί να φτάσει μόλις στο 59% του thoughput του native Linux, ενώ ο unprivileged guest, χρησιμοποιώντας τον netfront paravirt driver, περιορίζεται στο 56% της απόδοσης του privileged guest.

Για την πιο αναλυτική αξιολόγηση της απόδοσης της υλοποίησης σε χώρο πυρήνα, χρησιμοποιούνται ως μέτρο σύγκρισης τόσο το pktgen, όσο και η χρήση του netperf



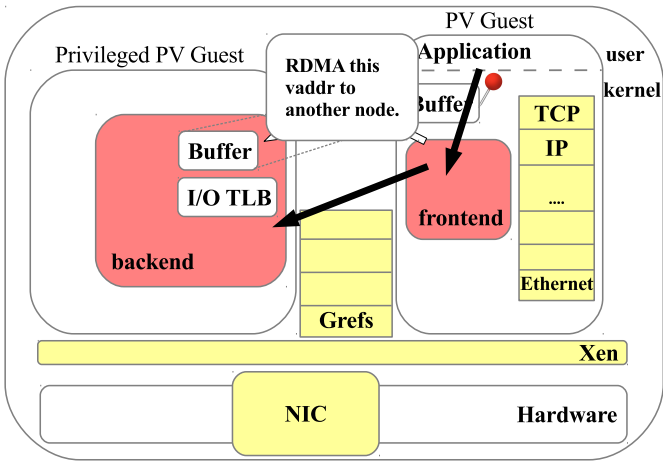
Σχήμα 5.4: Kernel-level SLURPoE-XEN - grant memory to dom0

TCP STREAM benchmark.

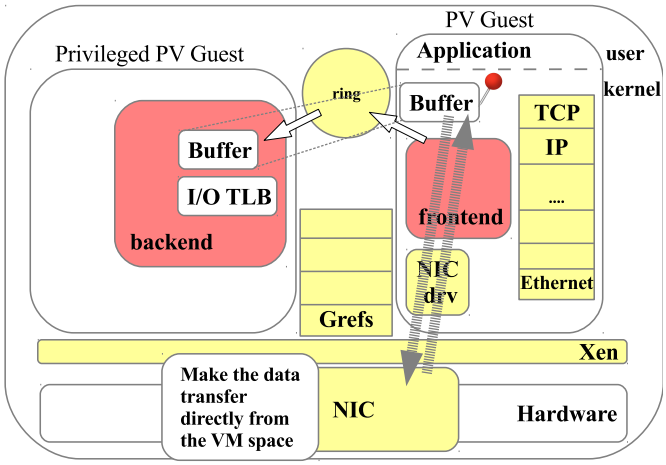
Για την αξιολόγηση του latency, υλοποιήσαμε ένα απλό RDMA write, με το οποίο μετρήσαμε το χρόνο για να γίνει transmit 1 byte από την εικονική μηχανή στον priviliged guest. Τα αποτελέσματα έδειξαν ότι χρειάζονται περίπου 14usecs, ώστε 1byte να περάσει τα ενδιάμεσα virtualization layers, και να φτάσει από το DomU στο Dom0.

Η υλοποίηση σε επίπεδο πυρήνα εμφανίζει σημαντική βελτίωση σε σχέση με το TCP STREAM benchmark, τόσο σε ό,τι αφορά το throughput και το latency, όσο και σε ό,τι αφορά το CPU utilization. Παρόλα αυτά, το CPU utilization στον priviliged guest, δεν είναι αμελητέο, και φτάνει μέχρι και το 34%.

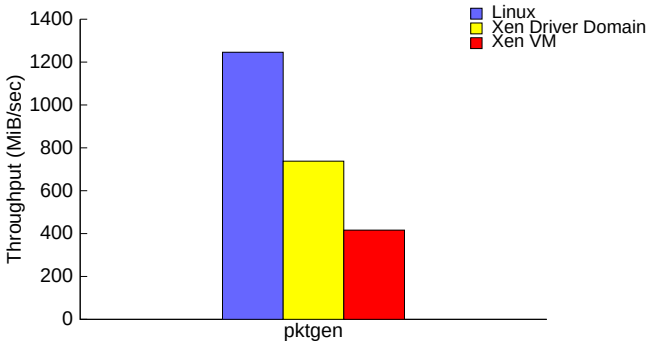
Συγκρίνοντας, την υλοποίηση σε επίπεδο πυρήνα, με την αρχική υλοποίηση του SLURPoE-XEN, με την χρήση έξυπνου προσαρμογέα δικτύου, παρατηρείται πολύ σημαντική βελτίωση του throughput, με μικρή όμως βελτίωση στο latency, και αυξημένο CPU utilization.



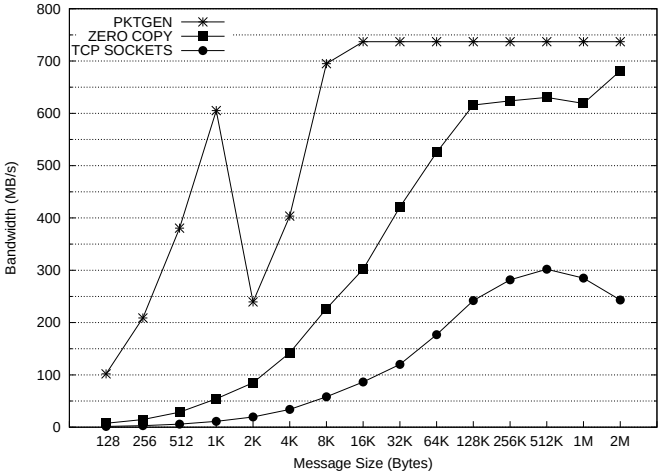
Σχήμα 5.5: Kernel-level SLURPoE-XEN - RDMA request



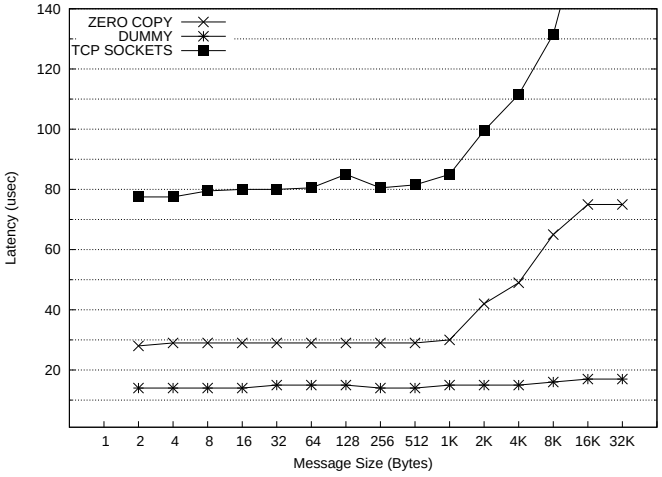
Σχήμα 5.6: Kernel-level SLURPoE-XEN - DMA data transfer



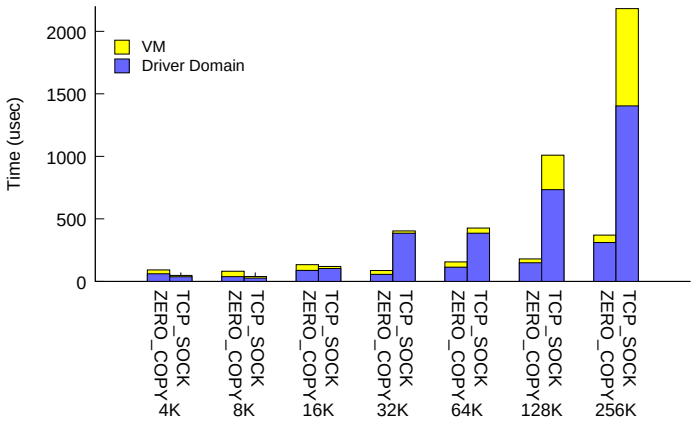
Σχήμα 5.7: Baseline throughput



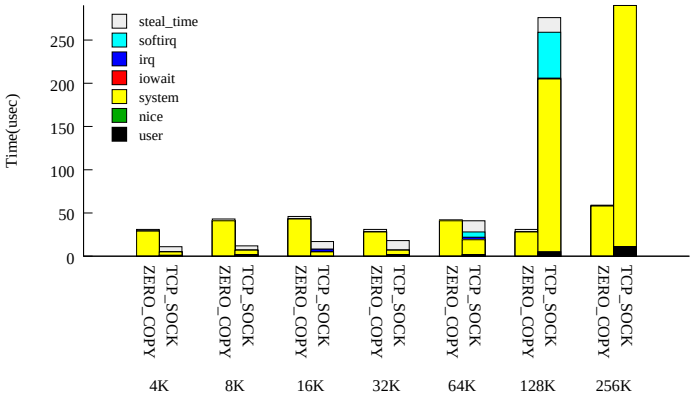
Σχήμα 5.8: Kernel-level SLURPoE-XEN end-to-end throughput



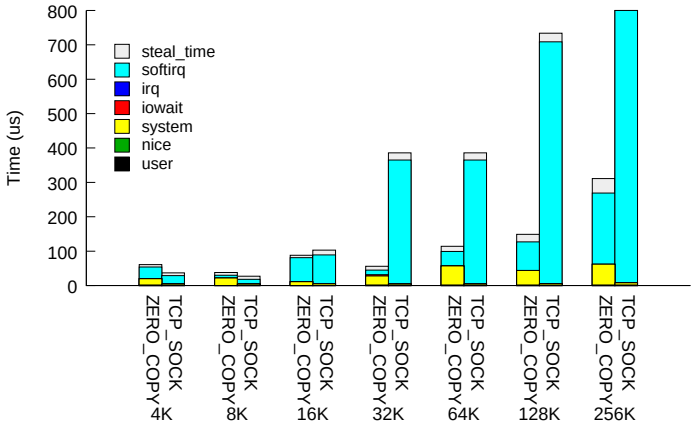
Σχήμα 5.9: Kernel-level SLURPoE-XEN one-way latency



Σχήμα 5.10: Kernel-level SLURPoE-XEN aggregate CPU utilization



Σχήμα 5.11: Kernel-level SLURPoE-XEN aggregate domU CPU utilization



Σχήμα 5.12: Kernel-level SLURPoE-XEN aggregate dom0 CPU utilization

Κεφάλαιο 6

Επίλογος

6.1 Σύνοψη

Σκοπός της παρούσας διπλωματικής εργασίας ήταν η μελέτη της συμπεριφοράς και της απόδοσης των δικτύων διασύνδεσης υψηλής επίδοσης σε εικονικά περιβάλλοντα.

Χρησιμοποιήσαμε το δίκτυο διασύνδεσης που αναπτύχθηκε κατά την διπλωματική εργασία "Σχεδίαση και Υλοποίηση μηχανισμού απομακρυσμένης απευθείας πρόσβασης στη μνήμη με χρήση προγραμματιζόμενου ελεγκτή 10GbE", και της υλοποίησης του πρωτοκόλλου SLURPoE για χρήση σε εικονικά περιβάλλοντα, που αναπτύχθηκε κατά τη διάρκεια της διπλμωατικής εργασίας "Ενταξη σημασιολογίας δικτύων υψηλής διασύνδεσης σε εικονικά περιβάλλοντα", στο Εργαστήριο Υπολογιστικών Συστημάτων, και κάναμε μια μελέτη για την συμπεριφορά και την απόδοσή τους, ενώ στην συνέχεια ακολουθώντας μια διαφορετική προσέγγιση υλοποιούμε το εικονικοποιημένο πρωτόκολλο σε επίπεδο πυρήνα, και αναλύουμε τα πλεονεκτήματα και τα μειονεκτήματα αυτής της προσέγγισης.

Αρχικά παρουσιάστηκε στον αναγνώστη το θεωρητικό υπόβαθρο που ήταν απαραίτητο για την κατανόηση των αποφάσεων που ελήφθησαν τόσο κατά την αξιολόγησης της απόδοσης των προηγούμενων υλοποιήσεων, όσο και για τους λόγους που οδήγησαν στην υλοποίηση σε επίπεδο πυρήνα.

Στη συνέχεια, έγινε μια σύντομη περιγραφή του δικτύου διασύνδεσης και του πρωτοκόλλου SLURPoE, και της υλοποίησής τους για χρήση σε εικονικά περιβάλλοντα. Παρουσιάστηκε η σχεδίαση και αρχιτεκτονική τους, καθώς και οι αρχικές μετρήσεις που διεξήχθησαν πάνω σε αυτό το δίκτυο διασύνδεσης.

Έχοντας μια γενική εικόνα της σχεδίασης και υλοποίησης του δικτύου διασύνδεσης, και των πρωτοκόλλων SLURPoE και SLURPoE-XEN, προχωρήσαμε σε μια ανάλυση της απόδοσής τους, τόσο σε επίπεδο υλικό, όσο και σε επίπεδο υλοποίησης, και προτάθηκαν και υλοποιήθηκαν ορισμένες βελτιστοποιήσεις (DMA Chaining).

Τέλος, παρουσιάστηκε η υλοποίηση του πρωτοκόλλου SLURPoE-XEN σε επίπεδο πυρήνα. Αναλύθηκαν οι αλλαγές που απαιτήθηκαν προκειμένου να ενσωματωθεί η υλο-

ποίηση του πρωτοκόλλου στον backend driver του Dom0, και παρατέθηκαν αποτελέσματα μετρήσεων, με βάση τα οποία αξιολογήθηκε η απόδοση της νέας υλοποίησης.

6.2 Συμπεράσματα

Η μελέτη των δικύτων διασύνδεσης υψηλής επίδοσης σε εικονικά περιβάλλοντα μας οδήγησε στα εξής συμπεράσματα:

- Η χρησιμοποίηση έξυπνων προσαρμογέων δικτύου για την υλοποίηση και εκτέλεση του πρωτοκόλλου ενός δικτύου διασύνδεσης, ενώ προσφέρει πολλές δυνατότητες, και δεν επιβαρύνει τους επεξεργαστές των κόμβων, επιβάλλει επιπλέον επίπεδα και overhead τα οποία μπορεί να οδηγήσουν σε suboptimal απόδοση. Χρειάζεται λεπτομερής ανάλυση των αναγκών του πρωτοκόλλου, ώστε να επιλεχθεί έτσι το υλικό του έξυπνου προσαρμογέα δικτύου, που από τη μία να μην περιορίζει την απόδοση, και από την άλλη όμως να είναι ρεαλιστικό σε ό,τι αφορά το κόστος του.

- Η υλοποίηση του πρωτοκόλλου σε επίπεδο πυρήνα από την άλλη δίνει πιο σύντομα και γρήγορα μονοπάτια ελέγχου και δεδομένων στις εφραμογές, γεγονός που οδηγεί σε καλύτερη απόδοση, αλλά επιβαρύνει τους επεξεργαστές των κόμβων, στερώντας τους υπολογιστική ισχύ.

- Για να επιτευχθεί η μέγιστη δυνατή απόδοση, πρέπει να μελετηθούν σε βάθος οι μηχανισμοί που προσφέρει τόσο ο πυρήνας του λειτουργικού, σε ό,τι αφορά το interface με το υλικό, όσο και οι μηχανισμοί που προσφέρει το εικονικό περιβάλλον, προκειμένου να βελτιστοποιηθούν τα κρίσιμα μονοπάτια, και να μειωθεί το overhead κρίσιμων λειτουργιών.

6.3 Μελλοντικές Επεκτάσεις

Σε καμία περίπτωση, η παρούσα διπλωματική εργασία, δεν μελετά όλους τους παράγοντες που επηρεάζουν την απόδοση των δικτύων υψηλής διασύνδεσης σε εικονικά περιβάλλοντα. Συγκεκριμένα, υπάρχουν αρκετές προσεγγίσεις, και βελτιώσεις οι οποίες δεν εξετάστηκαν, όπως:

- Η χρησιμοποιήση ενός ξεχωριστού driver domain στο Xen, το οποίο θα αναλαμβάνει αποκλειστικά την υλοποίηση του πρωτοκόλλου. Επεκτείνοντας την προσέγγιση αυτή, θα μπορούσε κανείς να υλοποιήσει έναν απόλυτα minimal kernel, ο οποίος θα περιέχει μόνο τον οδηγό για την υλοποίηση του πρωτοκόλλου, και για την επικονωνία με τους προσαρμογείς δικτύου, χωρίς το overhead ενός πλήρους λειτουργικού συστήματος.

- Η επίδραση του VM scheduling του Xen στην απόδοση του πρωτοκόλου και του δικτύου διασύνδεσης, και συγκεκριμένα, η ενσωμάτωση των αποτελεσμάτων της εργασίας Coexisting scheduling policies boosting I/O Virtual Machines [15].
- Η χρησιμοποίηση έξυπνων προσαρμογέων δικτύων, οι οποίοι υλοποιούν πολλαπλές ουρές (multiqueues), ή υποστηρίζουν SR-IOV τεχνικές, ώστε να δημιουργηθεί ένα άμεσο NIC-to-app datapath χωρίς καμία μεσολάβηση του Dom0.

Βιβλιογραφία

[1] G. J. Popek and R. P. Goldberg, ``Formal requirements for virtualizable third generation architectures,'' *Commun. ACM*, vol. 17, pp. 412--421, July 1974.

[2] Intel, ``Intel vt-x technology.''

[3] AMD, ``Amd virtualization.''

[4] Intel, ``Intel vt-d technology.''

[5] PCI SIG, ``SR-IOV,'' 2007. http://www.pcisig.com/specifications/iov/single_root/.

[6] A. Whitaker, M. Shaw, and S. D. Gribble, ``Denali: Lightweight virtual machines for distributed and networked applications,'' in *In Proceedings of the USENIX Annual Technical Conference*, 2002.

[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. A. Pratt, and A. Warfie ld, ``Xen and the Art of Virtualization,'' in *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, (New York, NY, USA), pp. 164--177, ACM, 2003.

[8] H. B. R.A.F Bhoedjang, T. Ruhl, ``User-level network interface protocols,'' *IEEE Computer*, pp. 53--60, Nov. 1998.

[9] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. king Su, ``Myrinet - A Gigabit-per-Second Local-Area Network,'' *IEEE Micro*, vol. 15, pp. 29--36, 1995.

[10] Myricom Inc., ``Myrinet EXpress (MX): A High Performance, Low-level, Message-Passing Interface for Myrinet,'' tech. rep., 2006.

[11] S. Toor, ``Introduction to infiniband white paper,'' 2003.

[12] J. Liu, W. Huang, B. Abali, and D. K. Panda, ``High performance VMM-bypass I/O in virtual machines,'' in *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, (Berkeley, CA, USA), pp. 3--3, USENIX Association, 2006.

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

[14] Ε. Κοζύρη, ``Ένταξη σημασιολογίας δικτύων διασύνδεσης υψηλής επίδοσης σε εικονικά περιβάλλοντα,'' 2010.

[15] D. Aragiorgis, A. Nanos, and N. Koziris, ``Coexisting scheduling policies boosting i/o virtual machines,'' in *Proceedings of the 6th Workshop on Virtualization in High-Performance Cloud computing (VHPC 2011), held in conjunction with Euro-par 2011*, 2011.