Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αποδοτικός διαμοιρασμός πόρων Εισόδου / Εξόδου σε περιβάλλοντα εικονικών μηχανών

Διδακτορική Διατριβή

Αναστάσιου Α. Νάνου

Διπλωματούχου Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών

Αθήνα,
Δεκέμβριος 2013

Η παρούσα έρευνα έχει συγχρηματοδοτηθεί από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο - ΕΚΤ) και από εθνικούς πόρους μέσω του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» του Εθνικού Στρατηγικού Πλαισίου Αναφοράς (ΕΣΠΑ) – Ερευνητικό Χρηματοδοτούμενο Έργο: Ηράκλειτος ΙΙ. Επένδυση στην κοινωνία της γνώσης μέσω του Ευρωπαϊκού Κοινωνικού Ταμείου.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αποδοτικός διαμοιρασμός πόρων Εισόδου / Εξόδου σε περιβάλλοντα εικονικών μηχανών

Διδακτορική Διατριβή

Αναστάσιου Α. Νάνου

Διπλωματούχου Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών

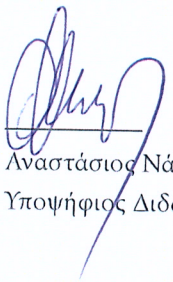
Συμβουλευτική Επιτροπή: Νεκτάριος Κοζύρης
Ανδρέας Σταφυλοπάτης
Ιωάννης Μαΐστρος

Εγκρίθηκε από την επταμελή επιτροπή την 27/12/2013

Νεκτάριος Κοζύρης Καθ. Ε.Μ.Π.	Ανδρέας Σταφυλοπάτης Καθ. Ε.Μ.Π.	Ιωάννης Μαΐστρος Επίκ. Καθ. Ε.Μ.Π.
Παναγιώτης Τσανάκας Καθηγ. Ε.Μ.Π.	Δημήτριος Τσουμάκος Επίκ. Καθ. Ιόνιου Παν.	Άγγελος Μπίλας Καθ. Παν. Κρήτης

Δημήτριος Νικολόπουλος
Καθ. Queen's University of Belfast

Αθήνα,
Νοέμβριος 2013



Αναστάσιος Νάνος
Υποψήφιος Διδάκτωρ Εθνικού Μετσοβίου Πολυτεχνείου

Στον αδερφό μου, Παναγιώτη,
στους γονείς μου, Αντώνη και Ελευθερία

Περίληψη

Οι υποδομές cloud computing προσφέρουν μεγάλη υπολογιστική ισχύ και φιλοξενούν ένα ευρύ φάσμα εφαρμογών που κυμαίνονται από υπηρεσιοστρεφείς εφαρμογές μέχρι επιστημονικές προσομοιώσεις υψηλών απαιτήσεων. Οι εφαρμογές υψηλών απαιτήσεων (HPC applications) συνήθως εκτελούνται κατανεμημένα, σε μεγάλο αριθμό κόμβων, με αποτέλεσμα η επικοινωνία μεταξύ των κόμβων να παίζει σημαντικό ρόλο στη συνολική επίδοση. Η εκτέλεσή τους σε εικονικά περιβάλλοντα προϋποθέτει την απαλοιφή των ενδιάμεσων επιπέδων του virtualization που προσδίδουν σημαντική επιβάρυνση τόσο στη ρυθμαπόδοση της επικοινωνίας όσο και στο χαμηλό χρόνο απόκρισης κατά την ανταλλαγή μηνυμάτων μεταξύ των κόμβων. Ταυτόχρονα, πρέπει να διατηρούνται τα πλεονεκτήματα του εικονικού περιβάλλοντος, όπως η ευελιξία στην εκτέλεση, το απομονωμένο περιβάλλον καθώς και η ευκολία στη διαχείριση των υποδομών. Οι σύγχρονες μέθοδοι για Είσοδο / Έξοδο σε εικονικά περιβάλλοντα είτε παρουσιάζουν μειωμένη επίδοση στην επικοινωνία, είτε απαιτούν εξειδικευμένο υλικό που πολυπλοκοποιεί τις υποδομές και μειώνει σημαντικά την ευελιξία στη διαχείρισή τους.

Στην παρούσα εργασία παρουσιάζεται μια εκτενής μελέτη των μεθόδων Ε/Ε σε εικονικά περιβάλλοντα με έμφαση στη δικτυακή επικοινωνία. Αρχικά περιγράφουμε τις βασικές αρχές των σύγχρονων δικτύων διασύνδεσης υψηλής επίδοσης. Στη συνέχεια αναλύουμε τα επίπεδα του λειτουργικού συστήματος που λαμβάνουν μέρος στην ανταλλαγή μηνυμάτων και περιγράφουμε με λεπτομέρεια τις επιλογές για Ε/Ε σε πλατφόρμες virtualization. Με βάση τη σχετική βιβλιογραφία, που προτείνει κυρίως λύσεις λογισμικού, σχεδιάζουμε και υλοποιούμε το Xen2MX: ένα δίκτυο διασύνδεσης σχεδιασμένο να παρέχει επικοινωνία υψηλής επίδοσης σε εικονικά περιβάλλοντα. Το Xen2MX είναι πλήρως συμβατό με το Myrinet/MX, χωρίς να απαιτεί την ύπαρξη εξειδικευμένων προσαρμογέων δικτύου. Συνδυάζει τα χαρακτηριστικά διαμοιρασμού μνήμης της πλατφόρμας virtualization Xen, με τεχνικές επικοινωνίας μηδενικών αντιγράφων (zero-copy) για να παρέχει στις εικονικές μηχανές απευθείας πρόσβαση στο δίκτυο με το χαμηλότερο δυνατό χρόνο απόκρισης.

Περίληψη

Cloud computing infrastructures provide vast processing power and host a diverse set of computing workloads, ranging from service-oriented deployments to High-Performance Computing (HPC) applications. As HPC applications scale to a large number of VMs, providing near-native network I/O performance to each peer VM is an important challenge. To deploy communication-intensive applications in the cloud, we have to fully exploit the underlying hardware, while at the same time retaining the benefits of virtualization: consolidation, flexibility, isolation, and ease of management. Current approaches present either limited performance or require specialized hardware that increases the complexity of the setup.

In this work, we present Xen2MX, a paravirtual interconnection framework, binary compatible with Myrinet/MX and wire compatible with MXoE. Its design is based on the Open-MX protocol, a port of the Myrinet/MX over generic Ethernet adapters. Xen2MX combines the zero-copy characteristics of Open-MX with Xen's memory sharing techniques; the objective is to construct the most efficient data path for high-performance communication in virtualized environments that can be achieved with software techniques.

Experimental evaluation of our prototype implementation shows that Xen2MX is able to achieve nearly the same raw performance as Open-MX running in a non-virtualized environment. On the latency front, Xen2MX reduces the RTT latency to less than 60% of the generic paravirtual setup with a software bridge and performs as close as 96% to the directly attached case (IOV). Regarding throughput, Xen2MX is able to nearly saturate a 10Gbps link, achieving 1159MB/s, compared to 1192MB/s of the directly-attached case. Xen2MX scales efficiently with the number of VMs, saturating the link for even smaller messages when 40 single-core VMs put pressure on the network adapters.

Περιεχόμενα

Κατάλογος σχημάτων

Αντί προλόγου

Η παρούσα διατριβή εκπονήθηκε στον τομέα Τεχνολογίας Πληροφορικής και Υπολογιστών, της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, του Εθνικού Μετσόβιου Πολυτεχνείου. Περιλαμβάνει την έρευνα και τα αποτελέσματα των μεταπτυχιακών μου σπουδών στη Σχολή Ηλεκτρολόγων Μηχανικών του ΕΜΠ και συγκεκριμένα στο Εργαστήριο Υπολογιστικών Συστημάτων. Μέσα από αυτό το κείμενο θα ήθελα να εκφράσω τις ευχαριστίες μου σε ένα πλήθος ανθρώπων που συνέβαλαν, ο καθένας με τον τρόπο του, στην ολοκλήρωση της εργασίας αυτής.

Πρώτον απ'όλους θα ήθελα να ευχαριστήσω τον επιβλέποντά μου, καθηγητή Νεκτάριο Κοζύρη για την επιστημονική καθοδήγηση κατά τη διάρκεια της έρευνάς μου, για τις πολύτιμες συμβουλές του καθώς και για την εμπιστοσύνη που μου έδειξε από την αρχή της συνεργασίας μας. Η δυνατότητα που μου έδωσε να συμμετέχω στο Εργαστήριο Υπολογιστικών Συστημάτων έπαιξε καθοριστικό ρόλο στη μετέπειτα πορεία μου. Με ενθάρυνε να ακολουθήσω τις επιλογές μου, με πίστη στην προσπάθειά μου, την οποία εξεδήλωνε κυρίως στις πιο δύσκολες στιγμές. Τον ευχαριστώ θερμά.

Ιδιαίτερα θερμές είναι οι ευχαριστίες μου και στον επίκουρο καθηγητή Γιάνη Μαΐστρο, η παρουσία του οποίου υπήρξε ηθική και επιστημονική έμπνευση για μένα, από τον πρώτο καιρό της φοίτησής μου στο μεταπτυχιακό πρόγραμμα. Πέρα από τις πολύτιμες και καθοριστικές συμβουλές του, μου έδωσε την ευκαιρία να συμμετάσχω στην ομάδα διαχείρισης του Υπολογιστικού Κεντρου ΣΗΜΜΥ. Θα ήθελα να τον ευχαριστήσω για τη δυνατότητα που μου έδωσε να καταπιαστώ με ενδιαφέροντα προβλήματα. Θα ήθελα επίσης να ευχαριστήσω και τον καθηγητή Ανδρέα-Γεώργιο Σταφυλοπάτη για την προθυμία του να συνδράμει με οποιονδήποτε τρόπο στην ολοκλήρωση της διατριβής αυτής.

Ιδιαίτερη αναφορά θα ήθελα επίσης να κάνω και στο Λέκτορα Γιώργο Γκούμα, για την ενθάρρυνσή του και την αξιοθαύμαστη ικανότητά του να εμπνέει αυτοπεποίθηση σε δύσκο-

λες στιγμές και να κάνει τα πράγματα να φαίνονται απλούστερα.

Η παρούσα διατριβή θα ήταν σίγουρα πολύ φτωχότερη χωρίς τις μακροσκελείς τεχνικές συζητήσεις με τους Δρ. Βαγγέλη Κούκη και Κορνήλιο Κούρτη. Η τεχνική τους κατάρτιση σε συνδυασμό με την προθυμία τους να συνδράμουν στην επίλυση προβλημάτων που προέκυπταν, συνέβαλαν αποφασιστικά στην ολοκλήρωση της δουλειάς αυτής. Θα ήθελα να τους ευχαριστήσω θερμά για την πολύτιμη βοήθεια τους και την ικανότητά τους να με προσγειώνουν απότομα στην πραγματικότητα με εύστοχη, καλοπροαίρετη κριτική.

Θα ήθελα να ευχαριστήσω τον επίκουρο καθηγητή και μέλος της επταμελούς επιτροπής Δημήτρη Τσουμάκο, που στα τελευταία βήματα της πορείας μου επέδειξε ενδιαφέρον για την έρευνά μου και με τις συμβουλές του, με βοήθησε σημαντικά στη διαμόρφωση και ολοκλήρωση της διατριβής.

Θα ήθελα επίσης να ευχαριστήσω ιδιαίτερα τους Δρ. Κωστή Νίκα, Νίκο Αναστόπουλο, και Βασίλη Καρακάση καθώς και τους Γιάννη Κωνσταντίνου, Νάσια Ασίκη και Κατερίνα Δόκα για τις πολύτιμες συμβουλές τους στην κατεύθυνση της έρευνάς μου. Οι Δρ. Άρης Σωτηρόπουλος, Αντώνης Χαζάπης και Αντώνης Ζήσιμος υπήρξαν πολύτιμοι αρωγοί στην πορεία μου στο εργαστήριο, τόσο κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας, όσο και της διατριβής μου.

Η καθημερινή μου επαφή με το εργαστήριο συνέβαλε καθοριστικά στη διαμόρφωσή μου ως ερευνητή – η παρουσία μου στο χώρο μου έδωσε τη δυνατότητα να συμμετάσχω σε αναρίθμητες επιστημονικές συζητήσεις υψηλού επιπέδου που μου άνοιξαν πολλούς ορίζοντες. Συγκεκριμένα θα ήταν παράλειψη να μην ευχαριστήσω τους Υ.Δ. Τάσο Κατσιγιάννη, Στέφανο Γεράγγελο, Βαγγέλη Αγγέλου, Δημήτρη Σιακαβάρα, Αλέξανδρο Χαριτάτο, Χριστίνα Μπούμπουκα, Νίκο Παπαηλίου, Γιάγκο Μυτιλήνη, Νικέλα Παπαδοπούλου και Χρήστο Μαντά.

Κατά τη διάρκεια της παραμονής μου στο εργαστήριο, συνέβαλα στην επίβλεψη διπλωματικών εργασιών οι οποίες στο σύνολό τους αποτέλεσαν τόσο το έναυσμα όσο και την επιβεβαίωση της πορείας της διατριβής μου. Η συνεργασία μου με τους ανθρώπους αυτούς ήταν άψογη και οι χρήσιμες απορίες τους με βελτίωσαν τόσο σαν ερευνητή όσο και σαν άνθρωπο. Συγκεκριμένα θα ήθελα να ευχαριστήσω ιδιαίτερα τους Νίκο Νικολέρη, Στράτο Ψωμαδάκη και Δημήτρη Αραγιώργη, που πέρα από πολύτιμοι συνεργάτες κατά τη διάρκεια εκπόνησης της διπλωματικής τους, παραμένουν πολύ καλοί μου φίλοι. Θα ήθελα επίσης να ευχαριστήσω την Ελίζα Κοζύρη για την εκπληκτική δουλειά που έκανε στη διπλωματική της, καθώς και τους Γιώργο Ανδριανάκη, Κωνσταντίνο Μπαρμπαρή, Κωνσταντίνο Μουζακίτη και Ιωάννα Αλιφιεράκη.

Θα ήθελα να ευχαριστήσω το Ίδρυμα Κρατικών Υποτροφιών, για τη στήριξη που μου παρείχε στα 3 πρώτα χρόνια εκπόνησης της διατριβής. Επιπλέον θα ήθελα να ευχαριστήσω την Ελίζα Αγγελίδη, την Έμμα Αφεντή και την Κατερίνα Ταχριλτζίδου που με βοήθησαν ουσιαστικά στα δύσκολα γραφειοκρατικά θέματα που προέκυπταν κατά καιρούς.

Η παρούσα έρευνα έχει συγχρηματοδοτηθεί από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο - ΕΚΤ) και από εθνικούς πόρους μέσω του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» του Εθνικού Στρατηγικού Πλαισίου Αναφοράς (ΕΣΠΑ) – Ερευνητικό Χρηματοδοτούμενο Έργο: Ηράκλειτος ΙΙ. Επένδυση στην κοινωνία της γνώσης μέσω του Ευρωπαϊκού Κοινωνικού Ταμείου.

Τέλος, θα ήθελα να ευχαριστήσω τους κοντινούς μου ανθρώπους, τους φίλους και την οικογένειά μου, που στάθηκαν δίπλα μου σε κάθε δύσκολη στιγμή και με βοήθησαν τόσο ηθικά όσο και υλικά στην προσπάθεια περάτωσης της διατριβής. Τους ευχαριστώ θερμά και ελπίζω να μπορώ να τους το ανταποδώσω έστω και στο ελάχιστο.

Κλείνοντας, Θα ήθελα να αναφέρω πως η παρούσα διατριβή εκπονήθηκε στο πλαίσιο ενός ανοιχτού, δημόσιου εκπαιδευτικού ιδρύματος, γεγονός που επιδρά καθοριστικά στη μορφή και τον προσανατολισμό της έρευνας. Με αυτή την έννοια και στους καιρούς που ζούμε, όπου τα πάντα υποτάσσονται στην αγοραία λογική, θεωρώ σημαντικό να ευχαριστήσω όλους αυτούς που αγωνίζονται για να διατηρηθεί ο δημόσιος χαρακτήρας του πανεπιστημίου.

Δεκέμβριος 2013,
Α.Νάνος

1

Εισαγωγή

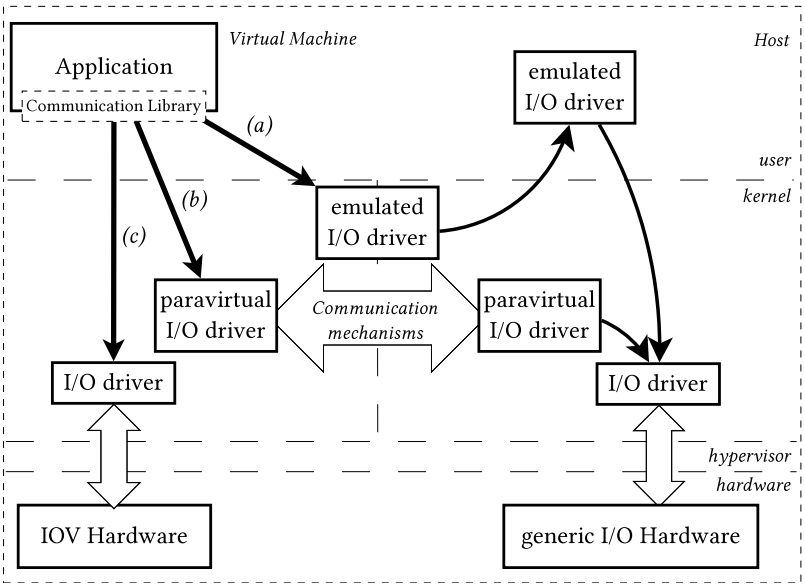
Τα σύγχρονα κέντρα δεδομένων προσφέρουν ευελιξία, προσαρμοσμένη εκτέλεση και απομόνωση σε ένα μεγάλο αριθμό υπηρεσιοστρεφών εφαρμογών όπως εξυπηρετητές ιστοσελίδων υψηλής επίδοσης, υπηρεσίες διαχείρισης δικτυακού κορμού κλπ. Αυτές οι υποδομές βασίζονται σε συστοιχίες υπολογιστικών συστημάτων πολλαπλών πυρήνων και προσφέρουν σημαντική υπολογιστική ισχύ· αυτό το χαρακτηριστικό τους τις καθιστά ιδανικές για εκτέλεση υπολογιστικά απαιτητικών εφαρμογών. Στο πλαίσιο του High-performance Computing (HPC), οι εφαρμογές συνήθως κλιμακώνουν σε ένα μεγάλο αριθμό από κόμβους, με αποτέλεσμα το δίκτυο διασύνδεσης που καθιστά δυνατή την επικοινωνία να παίζει πολύ σημαντικό ρόλο στην υψηλή ρυθμαπόδοση καθώς και το χαμηλό χρόνο απόκρισης.

Στο cloud computing, οι εφαρμογές που είναι απαιτητικές σε Είσοδο/Εξοδο παρουσιάζουν σημαντικά μειωμένη επίδοση [31, 42, 51] λόγω των επιμέρους επιπέδων που χρησιμοποιούνται για την πολύπλεξη των αιτήσεων πρόσβασης των εφαρμογών στο φυσικό μέσο. Το γεγονός αυτό είναι ένας από τους σημαντικότερους λόγους που οι εφαρμογές HPC δεν εκτελούνται ευρέως σε κέντρα δεδομένων cloud computing [44].

Πολλές εργασίες σε φυσικά [11, 24] και εικονικά [7, 8, 27, 29, 31, 32, 33, 34, 35, 51] περιβάλλοντα ερευνούν τη δυνατότητα παράκαμψης των μονοπατιών δεδομένων που δημιουργούν αυτή την καθυστέρηση με σκοπό την αύξηση της επίδοσης της Ε/Ε, βοηθώντας τις εφαρμογές να παρακάμψουν τη συμφόρηση στην ανάκτηση/τοποθέτηση δεδομένων από/προς συσκευές αποθήκευσης ή δικτύωσης. Παρόλα αυτά, σε εικονικό περιβάλλον, η επίδοση των συστημάτων Ε/Ε με συμβατικό υλικό, παρουσιάζεται σημαντικά μειωμένη σε σχέση με την αντίστοιχη σε φυσικό περιβάλλον. Η αιτία της μειωμένης επίδοσης βασίζεται στις διεπαφές που προσφέρουν οι ελεγκτές εικονικών μηχανών και στις συσκευές Ε/Ε: οι βελτιώσεις που προτείνονται στο λογισμικό είναι αρκετά ριζικές [5, 25, 27, 29], ενώ οι αντίστοιχες στο υλικό απαιτούν εξειδικευμένους προσαρμογείς [3, 9, 20, 37].

Οι λειτουργίες Ε/Ε σε εικονικά περιβάλλοντα εξυπηρετούνται είτε από επίπεδα λογι-

σμικού μέσα στον ελεγκτή εικονικών μηχανών, είτε από εξειδικευμένο υλικό (Σχήμα 1.1). Τα επίπεδα λογισμικού υλοποιούνται με: (a) *εξομοίωση* λειτουργιών συσκευών, (b) το *μοντέλο διαχωρισμένου οδηγού* που προσφέρει το paravirtualization [46]. Στο υλικό, περίπτωση (c), οι λειτουργίες υλοποιούνται με χρήση του μηχανισμού *αντιστοίχησης συσκευών (device assignment)*, όπου ο ελεγκτής εικονικών μηχανών επιτρέπει στις εικονικές μηχανές να αλληλεπιδρούν με το υλικό απευθείας. Αυτή η μέθοδος παρουσιάζει την καλύτερη απόδοση συγκριτικά με τις δύο προηγούμενες όσον αφορά στη ρυθμαπόδοση και στο χρόνο απόκρισης [31, 48].



Σχήμα 1.1: Πρόσβαση σε συσκευές Ε/Ε σε εικονικά περιβάλλοντα

Η περίπτωση της αντιστοίχησης συσκευών υπονοεί ότι ο προσαρμογέας θα είναι διαθέσιμος *μόνο* σε μία εικονική μηχανή, καθιστώντας αυτή την προσέγγιση ανεφάρμοστη σε περιβάλλοντα cloud, όπου, εξ' ορισμού, οι εικονικές μηχανές *μοιράζονται* υποδομές υλικού. Για να αντιμετωπιστεί αυτό το πρόβλημα, προτάθηκαν οι τεχνικές *I/O virtualization* [3, 35]. Οι τεχνικές αυτές συνδυάζουν τα πλεονεκτήματα της αντιστοίχησης συσκευών (υψηλή ρυθμαπόδοση, χαμηλός χρόνος απόκρισης) ενώ ταυτόχρονα επιτρέπουν πολλαπλές εικονικές μηχανές να μοιράζονται την ίδια συσκευή.

Η ερευνητική κοινότητα έχει προτείνει σημαντικές βελτιστοποιήσεις στις μεθόδους IOV · παρόλα αυτά, ένα σημαντικό ζήτημα παραμένει άλυτο: η ευελιξία. Η χρήση προσαρμογέων IOV καθιστά δυσκολότερη τη μεταφορά (migration) των εικονικών μηχανών, αφού ο βαθ-

μός ετερογένειας του υλικού σε υποδομές cloud computing αυξάνεται σημαντικά. Επιπρόσθετα, ο αριθμός των εικονικών μηχανών που μπορούν να απολαμβάνουν απευθείας πρόσβαση στη συσκευή περιορίζεται βάσει των ορίων του υλικού.

Αυτό είναι σοβαρό ζήτημα· οι πάροχοι υπηρεσιών cloud πρέπει να μπορούν να διαχειρίζονται την πρόσβαση των εικονικών μηχανών στο δίκτυο, για να μπορούν να χρησιμοποιούν αποδοτικότερα τον εξοπλισμό τους, ενώ ταυτόχρονα να παρέχουν Quality of Service (QoS) και Service Level Agreement (SLA)s. Από σχεδιασμού, οι τεχνικές IOV παρακάμπτουν τον ελεγκτή εικονικών μηχανών, αφού η πολύπλεξη των αιτήσεων πρόσβασης στη συσκευή γίνεται εξολοκλήρου στο υλικό. Κάποιες συγκεκριμένες συσκευές προσφέρουν μια υποτυπώδη διεπαφή διαχείρισης, η οποία όμως δεν αρκεί για ενιαία διαχείριση των δυνατοτήτων τους. Το γεγονός αυτό πολυπλοκοποιεί σημαντικά τη διαχείριση της πρόσβασης στο δίκτυο για ένα συγκεκριμένο εύρος εικονικών μηχανών.

Καθώς το Ethernet εγκαθιδρύεται στους κόσμους του Cloud computing και του HPC, χρειάζεται να μελετηθεί η επίδραση των πρωτοκόλλων ανταλλαγής μηνυμάτων στο cloud χωρίς την πολυπλοκότητα της στοίβας του TCP/IP. Οι σύγχρονες μέθοδοι δεν προσφέρουν μια λύση που να εκμεταλλεύεται πλήρως τις διεπαφές λογισμικού του ελεγκτή εικονικών μηχανών για την πρόσβαση στο υλικό. Στη συγκεκριμένη εργασία εξερευνούμε τις δυνατότητες που υπάρχουν για το διαμοιρασμό συσκευών, χρησιμοποιώντας προσαρμοσμένα πρωτόκολλα χαμηλού επιπέδου [31, 32, 34], για να καταλήξουμε σε μια καθολική σχεδίαση, αξιοποιώντας τη μελέτη των υποσυστημάτων Ε/Ε σε βάθος. Πρωταρχικός στόχος είναι η βελτιστοποίηση της επικοινωνίας των εικονικών μηχανών με το δίκτυο.

Στην παρούσα διατριβή, περιγράφουμε το σχεδιασμό και την υλοποίηση του Xen2MX, ενός πρωτοκόλλου ανταλλαγής μηνυμάτων υψηλής επίδοσης για εικονικά περιβάλλοντα. Τα χαρακτηριστικά του Xen2MX ελαχιστοποιούν και, σε περιπτώσεις, εξαλείφουν προβλήματα που σχετίζονται με τους κοινούς οδηγούς συσκευών που χρησιμοποιούνται σε υποδομές cloud στο πλαίσιο του HPC. Πιο συγκεκριμένα, ελαχιστοποιεί την επιβάρυνση του χειρισμού γεγονότων για ανταλλαγή μηνυμάτων και βελτιώνει σημαντικά τη ρυθμαπόδοση με: (a) χρήση τεχνικών μηδενικών αντιγραφών (zero-copy) για μεγάλα μηνύματα, (b) επαναχρησιμοποίηση αντιστοιχίσεων μνήμης μεταξύ εικονικών μηχανών και του ελεγκτή, (c) αποδέσμευση των μηνυμάτων ελέγχου από τη μεταφορά δεδομένων, χάρη στη βελτιστοποιημένη έκδοση του μηχανισμού καταναλωτή–παραγωγού για την επικοινωνία με στρώματα του λειτουργικού συστήματος και του ελεγκτή εικονικών μηχανών. Ο σχεδιασμός του Xen2MX μπορεί να εφαρμοστεί σε οποιοδήποτε ελεγκτή υποστηρίζει δυνατότητες paravirtualization.

Η συνεισφορά της διατριβής συνοψίζεται παρακάτω:

- Περιγράφουμε ενδελεχώς τις τεχνολογίες στις οποίες βασίζεται η παρούσα μελέτη. Συγκεκριμένα, αναλύουμε τη βιβλιογραφία για δίκτυα υψηλής επίδοσης σε συστοιχίες υπολογιστικών κόμβων, διερευνούμε τις δυνατότητες παράκαμψης επιπέδων του λειτουργικού συστήματος με στόχο την υψηλή επίδοση αποστολής / λήψης δεδομένων προς / από το δίκτυο, και μελετούμε τεχνολογίες virtualization με έμφαση στα κατώτερα στρώματα σχεδιασμού και υλοποίησης.
- Καταγράφουμε τις βασικές επιλογές σχεδίασης για ένα πρωτόκολλο ανταλλαγής μηνυμάτων υψηλής επίδοσης για εικονικά περιβάλλοντα και περιγράφουμε αναλυτικά τις δυνατότητες δικτύωσης σε σύγχρονες πλατφόρμες virtualization.
- Περιγράφουμε τη σχετική βιβλιογραφία, τονίζοντας τις διαφορετικές προσεγγίσεις πάνω στο ζήτημα της δικτύωσης υψηλής επίδοσης σε εικονικά περιβάλλοντα.
- Μελετάμε τους συμβιβασμούς που γίνονται μεταξύ της χρήσης βελτιστοποιήσεων στο λογισμικό και στο υλικό και αναλύουμε τους τρόπους αξιοποίησης των μεθόδων δικτύωσης για την απρόσκοπτη εκτέλεση εφαρμογών υψηλής επίδοσης στο cloud με τη μέγιστη δυνατή επίδοση.
- Παρουσιάζουμε το Xen2MX, ένα πρωτόκολλο υψηλής επίδοσης για εικονικά περιβάλλοντα, που βασίζεται στο Myrinet/MX. Το Xen2MX διαθέτει υπόστρωμα για το MPI χάρη στην πλήρη συμβατότητα με το Myrinet/MX.
- Εκτελούμε πειράματα για την αποτίμηση του Xen2MX καθώς και για να αναλύσουμε τη συμπεριφορά του. Στόχος μας είναι να αναδείξουμε συγκεκριμένα χαρακτηριστικά της προσέγγισής μας, σε σύγκριση με συμβατικούς τρόπους επικοινωνίας που χρησιμοποιούνται σήμερα σε εικονικά περιβάλλοντα. Δείχνουμε πως το Xen2MX επιφέρει σημαντική αύξηση του ρυθμού ανταλλαγής μηνυμάτων (ανεξαρτήτως μεγέθους) και, έτσι, επιτρέπει την απρόσκοπτη εκτέλεση εφαρμογών απαιτητικών σε Ε / Ε σε εικονικά περιβάλλοντα.

Το κείμενο δομείται ως εξής: πρώτα παρουσιάζουμε το θεωρητικό υπόβαθρο στο Κεφάλαιο 2, περιγράφοντας τις βασικές παραμέτρους των δικτύων υψηλής επίδοσης καθώς και του virtualization, και αναλύουμε τη σχετική βιβλιογραφία (Κεφάλαιο 3). Στη συνέχεια διερευνούμε ανορθόδοξα μονοπάτια μεταφοράς δεδομένων από / προς το δίκτυο και παρουσιάζομε τις βασικές επιλογές σχεδίασης ενός δικτύου υψηλής επίδοσης για εικονικά περιβάλλοντα (Κεφάλαιο 4). Στο Κεφάλαιο 5 περιγράφουμε αναλυτικά το σχεδιασμό και την υλοποίηση του Xen2MX ενώ στη συνέχεια, παρουσιάζουμε τα αποτελέσματα της πειραματικής αποτίμησης του πρωτοκόλλου (Κεφάλαιο 6). Τέλος, κλείνουμε με μια σύνοψη της δια-

τριβής, τις εφαρμογές της παρούσας υλοποίησης καθώς και μελλοντικές επεκτάσεις της (Κεφάλαιο 7).

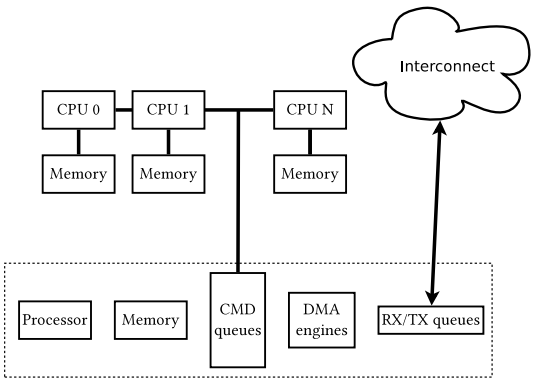
2

Θεωρητικό Υπόβαθρο

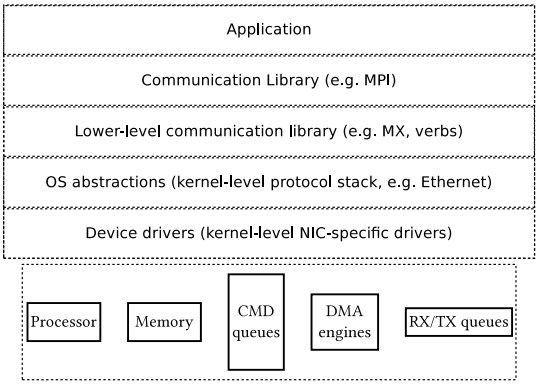
2.1 Δίκτυα Διασύνδεσης

Ένα σύγχρονο σύστημα υψηλής επίδοσης αποτελείται από κόμβους οι οποίοι περιέχουν επεξεργαστές, μνήμη και έναν προσαρμογέα δικτύου. Στις πιο διαδεδομένες αρχιτεκτονικές ο προσαρμογέας δικτύου βρίσκεται στο υποσύστημα Ε/Ε του κόμβου (πχ. το PCI-Express). Μια τυπική διάταξη προσαρμογέα φαίνεται στο σχήμα 2.1. Αυτή περιέχει έναν επεξεργαστή, ένα υποσύστημα μνήμης, μηχανές DMA, ουρές εντολών για την αλληλεπίδραση με το λογισμικό και ουρές για τη μετάδοση από και προς το δίκτυο.

Στο σχήμα 2.2 φαίνεται η αρχιτεκτονική του λογισμικού που τρέχει σε ένα σύστημα υψηλής επίδοσης. Στο σχήμα επίσης φαίνονται και διάφοροι πιθανοί τρόποι για την αλληλεπίδραση με τον προσαρμογέα δικτύου. Σε αυτούς συμπεριλαμβάνονται ο οδηγός του λειτουργικού συστήματος, η βιβλιοθήκη παράκαμψης του λειτουργικού ακόμα και εντολές φόρτωσης/αποθήκευσης σε συστήματα με υποστήριξη υλικού για Καθολικό Χώρο Διευθύνσεων (Global Address Space).



Σχήμα 2.1: Διάγραμμα του υλικού για έναν προσαρμογέα δικτύου υψηλής επίδοσης



Σχήμα 2.2: Η στοίβα του λογισμικού για ένα δίκτυο υψηλής επίδοσης

Η επίδοση ενός δικτύου διασύνδεσης καθορίζεται από ένα σύνολο παραγόντων εκ των οποίων οι πιο σημαντικοί είναι: το μέγιστο εύρος ζώνης και ο χρόνος αρχικής απόκρισης μικρών μηνυμάτων. Εκτός από την επίδοση των προσαρμογέων δικτύου και των μεταγωγέων στο δίκτυο, οι δύο παράμετροι επηρεάζονται σημαντικά από τη συνολική σχεδίαση του συστήματος. Η επίδοση της μνήμης, η χρησιμοποίηση της κρυφής μνήμης (cache), ακόμα και η αρχιτεκτονική του υλικού μπορεί να έχουν σοβαρό αντίκτυπο στην επίδοση του δικτύου διασύνδεσης.

- Το μέγιστο εύρος ζώνης αυξάνεται με σταθερό ρυθμό. Αν και δεν μπορεί να ακολουθήσει τους εκθετικούς ρυθμούς ανάπτυξης της επίδοσης των επεξεργαστών, έχει ξεπεράσει κατά πολύ την ανάπτυξη που έχει υποστεί η επίδοση της μνήμης. Το γεγονός αυτό, στο μέλλον, αναμένεται να αλλάξει – σήμερα υπάρχει μεγάλη αβεβαιότητα στα υλικά, όπως τα οπτικά καλώδια που υιοθετεί η βιομηχανία σε αντίθεση με το απαγορευτικό κόστος και τα φυσικά όρια των αυξανόμενων ρυθμών των σημάτων στα καλώδια χαλκού. Η υιοθέτηση των οπτικών καλωδίων και το αυξανόμενο κόστος της μεταγώγησης στο μέγιστο εύρος ζώνης, έχουν οδηγήσει στην ανάπτυξη ενδιαφέροντος για εναλλακτικές τοπολογίες δικτύων.
- Ο χρόνος αρχικής απόκρισης μειώνεται ασυμπτωτικά με αποτέλεσμα ο ελάχιστος χρόνος απόκρισης στη διεπαφή MPI να είναι ελαφρώς μικρότερος από το 1μsec στα μοντέρνα δίκτυα διασύνδεσης των κορυφαίων συστημάτων υψηλής επίδοσης. Μόνο ένα μικρό μέρος του χρόνου αρχικής απόκρισης οφείλεται στην καθυστέρηση μετάδοσης του μηνύματος στο καλώδιο. Για την ακρίβεια, μεγάλο μέρος του χρόνου αρχικής απόκρισης στη διεπαφή MPI καταναλώνεται στους κόμβους της πηγής και του προορισμού κατά τη μετακίνηση των δεδομένων από το δίκτυο στον επεξεργαστή του κόμ-

βου και στη στοίβα των πρωτοκόλλων επικοινωνίας. Αν και υπάρχει ενδιαφέρον για μείωση του χρόνου μετακίνησης των δεδομένων με την παράκαμψη του PCI-Express, η προσπάθεια να περιοριστεί η επιβάρυνση της επικοινωνίας από το λογισμικό έχει επικεντρωθεί κυρίαρχα στην επιτάχυνση των εργασιών επεξεργασίας μηνυμάτων με την υποστήριξη και από το υλικό.

Αυτό που έχει σημασία για τα σύγχρονα συστήματα υψηλής επίδοσης είναι όσο το δυνατόν απρόσκοπτη διοχέτευση της απόλυτης διαθέσιμης φυσικής υπολογιστικής ισχύος στους χρήστες με τις λιγότερες δυνατές απώλειες. Ωστόσο, τα ερωτήματα για το τι πρέπει να έχει ένα δίκτυο διασύνδεσης είναι πιο πολύπλοκα, ειδικά όταν εμπλέκεται το κριτήριο του υλικού (εξειδικευμένο ή μη) και του κόστους.

2.2 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 αντιγράφονται 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 κατάλ-

ληλη 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 [36]. Συνεπώς, προστίθεται καθυστέρηση λόγω των κλήσεων συστήματος, αφού αυτό συνεπάγεται αλλαγή της διεργασίας υπό εκτέλεση (context switch) και λόγω των πολλαπλών αντιγραφών των δεδομένων.

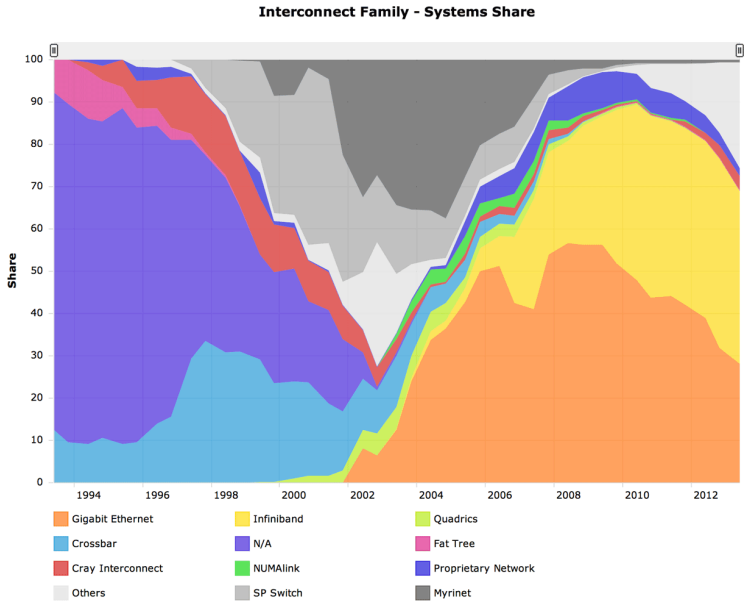
Για να αντιμετωπιστεί το πρόβλημα αυτό της απόδοσης έχουν αναπτυχθεί διάφορες αρχιτεκτονικές επικοινωνίας σε επίπεδο χρήστη (user-level communication), που αφαιρούν το λειτουργικό σύστημα από το μονοπάτι της επικοινωνίας.

Ένα από τα 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 Ε/Ε 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 με κλήσεις συστήματος.

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 υλικού 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 προσπελάσουν 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 ζωτικής σημασίας για τις εφαρμογές. Στα εμπορικά αυτά προϊόντα συγκαταλέγονται τα συστήματα διασύνδεσης υψηλής ταχύτητας όπως το InfiniBand και το Myrinet.

Στην ενότητα αυτή παρουσιάζουμε τα πιο διαδεδομένα δίκτυα διασύνδεσης τα τελευταία χρόνια, όπως άλλωστε αυτό αποδεικνύεται και από το μερίδιο τους στην αγορά συστημάτων υψηλής επίδοσης 2.3.



Σχήμα 2.3: Τα στατιστικά δικτύων διασύνδεσης στη λίστα των 500 πιο ισχυρών υπολογιστικών συστημάτων.

2.2.1 Myrinet

Εισαγωγή

Το Myrinet[6] είναι ένα δίκτυο υψηλής επίδοσης που βασίζεται σε τεχνολογίες διασύνδεσης πολλαπλών επεξεργαστών σε αρχιτεκτονικές MPP (Massivily Parallel Processors). Προσφέρει ταχύτητες σύνδεσης μέχρι και 10Gbps, δυνατότητα μετάδοσης δεδομένων και στις δύο κατευθύνσεις (full duplex) και μικρούς χρόνους αρχικής απόκρισης. Το Myrinet βρίσκει εφαρμογή σε συστοιχίες υπολογιστών και γενικότερα χρησιμοποιείται ως Δίκτυο Περιοχής Συστήματος (System Area Network – SAN).

Για το δίκτυο Myrinet υπάρχουν δύο διαθέσιμες εκδόσεις: το Myri–10G και το Myrinet – 2000. Στο φυσικό επίπεδο το Myrinet–2000 προσφέρει δυνατότητα μετάδοσης δεδομένων προς τις δύο κατευθύνσεις, σε οπτικές συνδέσεις σημείου–προς–σημείο 2+2Gbps. Το Myri–10G βασίζεται στο ίδιο φυσικό επίπεδο (PHY, layer – 1) όπως και το 10G Ethernet, αυξάνοντας έτσι το εύρος διαύλου στα 10Gbps και μπορεί να χρησιμοποιήσει ως επίπεδο συνδέσμου είτε το 10G Ethernet, είτε το Myrinet με δρομολόγηση από την πηγή.

Οι κόμβοι σε ένα δίκτυο Myrinet διασυνδέονται με μεταγωγείς τύπου Crossbar, σε μια τοπολογία Clos. Ένα από τα πιο σημαντικά χαρακτηριστικά του δικτύου είναι η δρομολό-

γηση από την πηγή: το δίκτυο χαρτογραφείται έτσι ώστε κάθε κόμβος που συμμετέχει να γνωρίζει τη διαδρομή μέχρι κάποιον άλλο, χρησιμοποιώντας δρομολόγηση up/down. Κάθε πακέτο περιέχει όλη την πληροφορία για τη διαδρομή μέχρι τον προορισμό του, ως μια σειρά από πόρτες που πρέπει να περάσει μέσα σε κάθε μεταγωγέα. Για την επικοινωνία των κόμβων χρησιμοποιούνται πακέτα μεταβλητού μήκους.

Για να επιτευχθεί η ελάχιστη, κατά το δυνατό, επιβάρυνση του επεξεργαστή από τη χρήση του δικτύου, το Myrinet χρησιμοποιεί τεχνικές υλοποίησης του πρωτοκόλλου στο χώρο χρήστη. Με τον τρόπο αυτό εξασφαλίζεται η απεμπλοκή του πυρήνα από το κρίσιμο μονοπάτι. Σε αυτό το μοντέλο, μια εφαρμογή μπορεί να ελέγχει τη διεπαφή του δικτύου απευθείας. Καθώς ο πυρήνας δεν εμπλέκεται στην επικοινωνία, το ρόλο του αναλαμβάνουν μια βιβλιοθήκη στο χώρο χρήστη, καθώς και το υλικολογισμικό (firmware) στη διεπαφή δικτύου. Η ανταλλαγή δεδομένων μεταξύ της εφαρμογής και του προσαρμογέα δικτύου γίνεται με ένα μηχανισμό που προετοιμάζεται από κώδικα με αυξημένα δικαιώματα. Ο κώδικας αυτός υλοποιείται μέσα σε ένα module του πυρήνα του λειτουργικού.

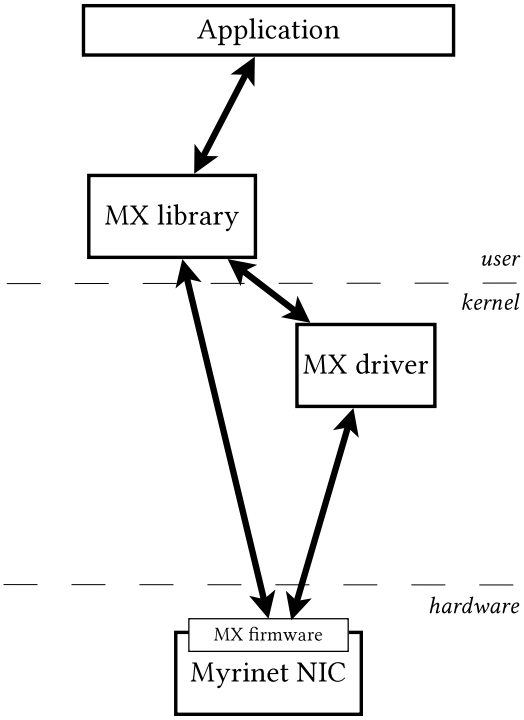
Μια εφαρμογή μπορεί να έχει τον έλεγχο του προσαρμογέα δικτύου με την αντιστοίχηση στο χώρο μνήμης της, μέρους της φυσικής του μνήμης. Χρησιμοποιεί εντολές φόρτωσης/αποθήκευσης σε συγκεκριμένες θέσεις μνήμης, πραγματοποιώντας έτσι την επικοινωνία.

Myrinet eXpress: Το σύστημα διαβίβασης μηνυμάτων του Myrinet

Το MX [30] (Myrinet Express) είναι ένα λογισμικό διαβίβασης μηνυμάτων χαμηλού επιπέδου προσαρμοσμένο στο Myrinet. Το MX υποστηρίζει μοντέρνες διεπαφές middleware όπως το MPI και το VI, εκμεταλλευόμενο τις ικανότητες επεξεργασίας στον προσαρμογέα δικτύου. Επιτρέπει την εξομοίωση του Ethernet σε ταχύτητες συνδέσμου με μικρή επιβάρυνση και προσφέρει μια απλή προγραμματιστική διεπαφή για την ανάπτυξη εφαρμογών που κάνουν χρήση του Myrinet.

Τα κυρίαρχα χαρακτηριστικά του MX συνοψίζονται στα εξής:

- Προστατευόμενη και ανεξάρτητη πρόσβαση στις δικτυακές λειτουργίες για εφαρμογές στο χώρο χρήστη.
- Διαφανής δήλωση της μνήμης.
- Χαμηλός χρόνος πρώτης απόκρισης για μικρά μηνύματα (της τάξης του 1usec).
- Ασύγχρονες λειτουργίες επικοινωνίας.
- Υποστήριξη αποστολής και λήψης μηνυμάτων από και προς διάσπαρτες θέσεις στη μνήμη.



Σχήμα 2.4: Η αρχιτεκτονική του MX.

- Μηχανισμός γενικευμένης ταυτοποίησης μηνυμάτων.
- Επικάλυψη της επικοινωνίας και των υπολογισμών ακόμα και για μεγάλα μηνύματα.
- Αξιόπιστη ταυτοποίηση σειράς μηνυμάτων.
- Αποδοτική υποστήριξη για μη αναμενόμενα μηνύματα.
- Αποκατάσταση λαθών στο δίκτυο και υψηλή διαθεσιμότητα.
- Βασικός μηχανισμός ταυτοποίησης για κάθε μήνυμα.
- Για κάθε μήνυμα ή για κάθε άκρο συναρτήσεις που κάνουν δοκιμές με τακτική σάρωση (polling) ή σταματούν τη ροή ελέγχου μέχρι την ολοκλήρωσή τους (blocking)
- Υποστήριξη για δρομολόγηση διασποράς.
- Ενσωματωμένη υποστήριξη χαρτογράφησης του δικτύου
- Υποστήριξη ακύρωσης των αιτήσεων σε αναμονή.
- Βιβλιοθήκες τόσο ενός νήματος αλλά και ασφαλών νημάτων.

Στη συνέχεια περιγράφονται σημαντικές ενέργειες που εκτελούνται στο πλαίσιο της επικοινωνίας με το MX. Η τυπική σειρά την οποία ακολουθεί μια εφαρμογή που κάνει χρήση του λογισμικού είναι: αρχικοποίηση της βιβλιοθήκης, αρχικοποίηση ενός άκρου (endpoint), σύνδεση με τους κόμβους προορισμού, έναρξη αποστολής και λήψης μηνυμάτων που ακολουθούνται από κλήσεις σε συναρτήσεις που παρακολουθούν την εξέλιξη των αιτημάτων, κλείσιμο του άκρου και απελευθέρωση των δομών της βιβλιοθήκης.

Τα άκρα της επικοινωνίας (Endpoints) Ένα άκρο στο MX είναι ένας εικονικός προσαρμογέας δικτύου στο επίπεδο της διεργασίας, ο οποίος παρέχει πρόσβαση στο υλικό διασύνδεσης. Η πρόσβαση αυτή είναι προστατευμένη από άλλες διεργασίες. Το άκρο είναι επίσης ένα στιγμιότυπο της διεπαφής του λογισμικού. Η αναφορά σε αυτό γίνεται με μια μεταβλητή τύπου mx_endpoint_t, και χρησιμοποιείται στις περισσότερες από τις λειτουργίες του MX. Όλες οι λειτουργίες σε ένα αρχικοποιημένο άκρο είναι περιορισμένες μέσα σε αυτό. Τα αντικείμενα του MX, όπως είναι ένας χειριστής μιας αίτησης για αποστολή ή λήψη, σχετίζονται με ένα συγκεκριμένο άκρο και δεν έχουν νόημα έξω από αυτό ακόμα και μέσα στην ίδια διεργασία.

Ένα άκρο μπορεί να δημιουργηθεί από την κλήση της mx_open_endpoint(), που επιστρέφει ένα δείκτη στη δομή χειρισμού του, που έχει τύπο mx_endpoint_t. Αν η mx_open¬ _endpoint δεν επιστρέψει MX_SUCCESS, τότε η μεταβλητή τύπου mx_endpoint_t που πέρασε ως παράμετρος παραμένει αμετάβλητη.

Διευθυνσιοδότηση των άκρων Για την επικοινωνία με ένα απομακρυσμένο άκρο μια εφαρμογή πρέπει να έχει τη διεύθυνση του, που αναπαρίσταται από μια μεταβλητή τύπου mx_¬ endpoint_addr_t. Η διεύθυνση αυτή κατασκευάζεται από τρεις πληροφορίες: το αναγνωριστικό του προσαρμογέα δικτύου (NIC ID), το αναγνωριστικό του άκρου και μια τιμή φίλτρου. Μια μεταβλητή αυτού του τύπου δημιουργείται από μια κλήση στη συνάρτηση mx_¬ connect() και μπορεί να χρησιμοποιηθεί μόνο από το άκρο που πέρασε ως παράμετρος κατά την κλήση της.

Το αναγνωριστικό του προσαρμογέα είναι μεγέθους 64bit και μπορεί να ανακτηθεί μέσα σε έναν κόμβο με την κλήση της συνάρτησης mx_board_number_to_nic_id(). Το αναγνωριστικό του άκρου είναι ένας ακέραιος που υπάρχει για κάθε αρχικοποιημένο άκρο. Μπορεί είτε να οριστεί από την εφαρμογή που το αρχικοποιεί, είτε από τη βιβλιοθήκη του MX. Αυτή είναι μια τιμή ανάμεσα από το 0 και το MX_MAX_ENDPOINTS - 1. Το φίλτρο είναι ένας ακέραιος, που παίρνει τιμή από την εφαρμογή, για να διαχωρίσει ξεχωριστές λήψης της ίδιας

εφαρμογής.

Η αντιστοίχιση Η αντιστοίχιση αφορά στη διαδικασία συσχετισμού ενός εισερχόμενου μηνύματος με μια εκκρεμούσα λήψη. Κάθε διεπαφή διαβίβασης μηνυμάτων ορίζει τους δικούς της κανόνες με βάση στοιχεία που παρέχονται από το άκρο αποστολής και/ή το άκρο λήψης. Ο ισχυρός μηχανισμός αντιστοίχησης είναι απαραίτητος για την εγκατάσταση μιας σύνθετης διεπαφής διαβίβασης μηνυμάτων πάνω από μια χαμηλού επιπέδου διεπαφή ή για την κατευθείαν υλοποίηση εφαρμογών πάνω από αυτή.

Το MX παρέχει μια ευέλικτη αλλά και πολύ ισχυρή διεπαφή αντιστοίχησης Κάθε μήνυμα στο MX περιέχει 64bit πληροφορίας που χρησιμοποιείται στην αντιστοίχιση. Ο αποστολέας καθορίζει την πληροφορία αυτή, match_send ως μέρος της διαδικασίας αποστολής και ο παραλήπτης παρέχει το match_recv και μια μάσκα match_mask όταν κάνει μια αίτηση λήψης. Ένα εισερχόμενο μήνυμα θα συσχετιστεί με μια εκκρεμούσα λήψη όταν και μόνο όταν, το εισερχόμενο match_send με τη μάσκα match_mask ταιριάζει με την πληροφορία match_recv στην πλευρά της λήψης.

Αιτήσεις Οι αιτήσεις είναι μεταβλητές – χειριστές που ξεχωρίζουν συγκεκριμένα στιγμιότυπα από το σύνολο των ασύγχρονων λειτουργιών που εκκρεμούν. Για όλες τις ασύγχρονες λειτουργίες στο MX, πρέπει να δημιουργηθεί ένα αντικείμενο mx_request_t. Το αντικείμενο αυτό χρησιμοποιείται για να καθορίσει τη λειτουργία που εκκρεμεί σε μια οποιαδήποτε επόμενη λειτουργία. Χειριστές αιτήσεων επιστρέφονται από τις συναρτήσεις mx_isend(), mx¬ _issend() και irecv() και μπορούν να περαστούν ως παράμετροι στις συναρτήσεις mx_test¬ (), mx_wait(), mx_ibuffered() και mx_cancel(). Αν κάποια από τις ασύγχρονες συναρτήσεις δεν επιστρέψει MX_SUCCESS τότε ο χειριστής της άιτησης, τύπου mx_request_t παραμένει αμετάβλητος.

Κάθε αίτηση για λήψη πρέπει να έχει μια επιτυχή αντιστοίχιση σε μια κλήση της mx_¬ test() ή της mx_wait(). Με τον τρόπο αυτό μπορεί να ελευθερώσει και να ανακυκλώσει τους πόρους που σχετίζονταν με την αίτηση, όταν αυτό καταστεί εφικτό.

2.2.2 InfiniBand

Εισαγωγή

Το InfiniBand[1] ορίζει ένα πρότυπο αρχιτεκτονικής Ε/Ε (InfiniBand Architecture – IBA) που χρησιμοποιείται για να διασυνδεθούν εξυπηρετητές, εξοπλισμός υποδομής επικοινω-

νιών, συσκευές αποθήκευσης και ενσωματωμένα συστήματα. Ένα σύστημα αρχιτεκτονικής InfiniBand μπορεί να είναι ένας μικρός εξυπηρετητής με έναν επεξεργαστή και μερικές συσκευές E/E, μέχρι και ένα σύστημα MPP με εκατοντάδες επεξεργαστές και χιλιάδες συσκευές E/E. Η διασύνδεση στο InfiniBand πραγματοποιείται μέσω μιας υποδομής επικοινωνίας με μεταγωγείς που προσφέρουν μεταφορά δεδομένων με ταχύτητες μέχρι και 120Gbps. Έχει εφαρμογές τόσο μέσα στα πλαίσια ενός κόμβου, όσο και έξω από αυτόν, μέσω εξωτερικών συνδέσεων από χαλκό ή οπτικές ίνες.

Το InfiniBand επιτυγχάνει χαμηλό χρόνο πρώτης απόκρισης, απαιτεί μικρή επιβάρυνση επεξεργασίας και έχει σχεδιαστεί για τη μεταφορά διαφορετικών τύπων κίνησης πάνω από μια μόνο σύνδεση. Πάνω από την ίδια υποδομή διακινούνται με τρόπο αποδοτικό τόσο δεδομένα που αφορούν στη επικοινωνία επεξεργαστών, όσο και αυτά που αφορούν σε συστήματα αποθήκευσης. Το InfiniBand χρησιμοποιείται σε κέντρα δεδομένων (data centers), συστοιχίες υπολογιστών υψηλής απόδοσης (HPC clusters) και ενσωματωμένα συστήματα προσδίδοντας τους δυνατότητες κλιμάκωσης, από δύο μέχρι εκατοντάδες κόμβους.

Η αρχιτεκτονική του InfiniBand προέρχεται από τη συνένωση δυο προτύπων E/E, το Future I/O (Compaq, IBM και Hewlett – Packard) και το Next Generation I/O (Intel, Microsoft και Sun). Την ανάπτυξη του έχει αναλάβει το InfiniBand Trade Association (IBTA). Η ραγδαία ανάπτυξη των συστοιχιών υπολογιστών και η κοινή προσέγγιση του από τις εταιρίες, καθιστούν το InfiniBand, ένα ανοιχτό και υλοποιημένο από πολλές εταιρίες πρότυπο. Για τη λειτουργία του διατίθεται υποδομή σε ταχύτητες των 10Gbps (SDR – Single Data Rate), των 20Gbps (DDR – Double Data Rate) ενώ από το 2008 παρουσιάστηκε εξοπλισμός για ταχύτητες μέχρι και 40Gbps (QDR – Quad Data Rate).

Έχει σχεδιαστεί με γνώμονα να προσφέρει Αξιοπιστία, Διαθεσιμότητα, και Λειτουργικότητα (Reliability, Availability, Serviceability – RAS). Η αρχιτεκτονική υποστηρίζει το πρωτόκολλο IP για τη σύνδεση με το Διαδίκτυο και τους μακρινούς εξυπηρετητές. Επιπλέον καθορίζει ένα υλικό επικοινωνίας με υποστήριξη μεταγωγέων με υψηλό εύρος ζώνης και χαμηλό χρόνο πρώτης απόκρισης. Το μεγαλύτερο μέρος του υπολογιστικού φόρτου για την επικοινωνία μεταβιβάζεται στο υλικό του InfiniBand.

Τα χαρακτηριστικά της αρχιτεκτονικής InfiniBand

Υψηλή επίδοση Το InfiniBand παρέχει συγκριτικά μικρούς χρόνους πρώτης απόκρισης (της τάξης του 1μsec) σε σχέση με τα πιο διαδεδομένα δίκτυα διασύνδεσης. Η υποδομή που το υλοποιεί έχει εύρος ζώνης μέχρι και 120Gbps.

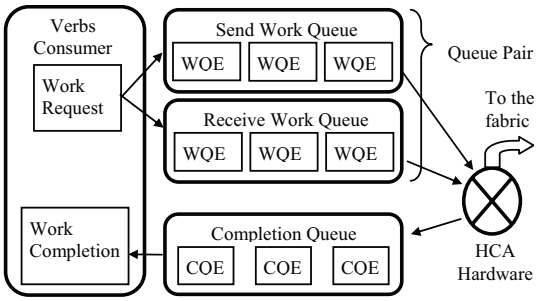
Μειωμένη πολυπλοκότητα Το InfiniBand επιτρέπει τη συγχώνευση πολλαπλών λειτουρ-

γιών Ε/Ε σε ένα ενιαίο καλώδιο ή μια εσωτερική διασύνδεση, χαρακτηριστικό κρίσιμο για τους υπολογιστές Blade, τα κέντρα δεδομένων, τα συστήματα αποθήκευσης και τα ενσωματωμένα συστήματα. Το InfiniBand συγχωνεύει τη διαβίβαση σημάτων που αφορούν στις συστοιχίες συστημάτων, στις επικοινωνίες, στα συστήματα αποθήκευσης και στα συστήματα διαχείρισης πάνω από μια μόνο σύνδεση. Η συγχώνευση αυτή των Ε/Ε σε ένα ενιαίο υλικό του InfiniBand μειώνει σημαντικά τις ανάγκες σε χώρο, κόστος αλλά και σε ηλεκτρική ενέργεια, στοιχεία πολύ κρίσιμα για τις συστοιχίες εξυπηρετητών. Οι υπόλοιπες τεχνολογίες διασύνδεσης ταιριάζουν λιγότερο σε ενιαία υλικά, γιατί ο αρχικός τους σχεδιασμός δεν έχει γίνει για να υποστηρίζει τη μεταφορά δεδομένων διαφορετικού τύπου.

Υψηλή επίδοση διασύνδεσης Το InfiniBand αναπτύχθηκε για να παρέχει καλή κλιμάκωση. Για τις ανάγκες της επεξεργασίας κατά την επικοινωνία, παρέχει το κατάλληλο υλικό – αποφορτίζοντας το CPU από το ρόλο αυτό – και επιτρέπει την πλήρη χρησιμοποίηση των πόρων κάθε κόμβου που προστίθεται στη συστοιχία. Επιπλέον υποστηρίζει την Απομακρυσμένη Απευθείας Πρόσβαση στη Μνήμη (RDMA) που αποτελεί ένα βελτιστοποιημένο πρωτόκολλο μεταφοράς δεδομένων και αφήνει τον εξυπηρετητή να επικεντρωθεί στην επεξεργασία των προγραμμάτων που τρέχουν. Το RDMA συμβάλει στην καλή απόδοση τόσο σε εφαρμογές για συστοιχίες εξυπηρετητών, όσο και συστημάτων αποθήκευσης.

Αξιόπιστες και σταθερές συνδέσεις Το InfiniBand παρέχει αξιόπιστα δίπλα στοιχεία σύνδεσης. Αυτή η ικανότητα υλοποιείται στο υλικό. Επιπλέον, διευκολύνει εικονικές (virtualization) λύσεις διασύνδεσης, οι οποίες επιτρέπουν πολλαπλές εφαρμογές να χρησιμοποιούν απομονωμένα το ίδιο δίκτυο. Κατά συνέπεια, πολλαπλές εφαρμογές τρέχουν ταυτόχρονα σε σταθερές συνδέσεις, ελαχιστοποιώντας το χρόνο που δε λειτουργούν. Τα υλικά κατασκευάζονται με λογικές πλεονασμού σχεδόν σε όλα τα επίπεδα. Στόχος είναι αν μια σύνδεση εμφανίσει κάποια βλάβη, όχι μόνο το σφάλμα να περιορίζεται στη μια αυτή σύνδεση, αλλά και μια πρόσθετη σύνδεση να μπορεί αυτόματα να εξασφαλίσει τη σύνδεση σε όλο το υλικό. Ο πλεονασμός μέσα στο υλικό έχει ως αποτέλεσμα την υψηλή αξιοπιστία του.

Πολλές λειτουργίες του InfiniBand βασίζονται σε ένα μηχανισμό από δύο ουρές με εντολές προς εκτέλεση από το υλικό. Όπως φαίνεται και στο σχήμα 2.5, υπάρχει μια ουρά για διαδικασίες αποστολής, μια ουρά για διαδικασίες παραλαβής καθώς και μια ουρά ολοκληρωμένων αιτημάτων. Η ουρά αποστολής κρατά εντολές οι οποίες καθορίζουν πώς θα μετα-



Σχήμα 2.5: Ο μηχανισμός ουρών στο InfiniBand

φερθούν τα δεδομένα από τη μνήμη του αποστολέα προς τη μνήμη του παραλήπτη. Η ουρά λήψης κρατά εντολές οι οποίες καθορίζουν πού θα αποθηκευθούν δεδομένα που έχουν παραληφθεί.

Όταν μια αίτηση κατατεθεί, η εντολή της τοποθετείται στην κατάλληλη ουρά. Έπειτα ο προσαρμογέας του καναλιού εκτελεί την εντολή όταν έρθει η σειρά της. Μετά το τέλος της εκτέλεσης την τοποθετεί στην ουρά των ολοκληρωμένων αιτημάτων.

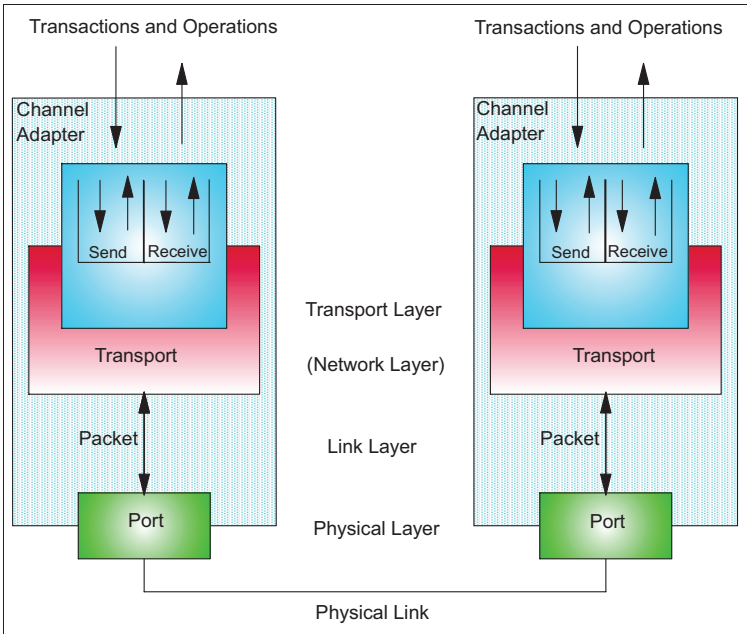
Το InfiniBand μπορεί να παρέχει υπηρεσίες τόσο συνδεσιοστρεφείς (connection – oriented) όσο και δεδομενογραμμάτων (datagram ή connection-less). Στις συνδεσιοστρεφείς υπηρεσίες, υπάρχει μια συσχέτιση μεταξύ ενός προορισμού (ζεύγη ουρών αποστολής και λήψης) στην πλευρά του αποστολέα και του παραλήπτη. Αυτή η συσχέτιση είναι απότοκο της αρχιτεκτονικής διαύλου, στην οποία όλη η επικοινωνία μοιράζεται ένα κοινό κανάλι.

Στη διαδικασία μιας αποστολής, ο προσαρμογέας του καναλιού διερμηνεύει τον τύπο της ενέργειας, δημιουργεί ένα μήνυμα, το τεμαχίζει (αν χρειάζεται) σε πολλαπλά πακέτα, προσθέτει την πληροφορία δρομολόγησης και στέλνει τα πακέτα σε μία πόρτα. Το λογικό κύκλωμα της πόρτας είναι πλέον υπεύθυνο για την αποστολή του πακέτου στο άλλο άκρο. Όταν φθάσουν τα πακέτα στο κύκλωμα της πόρτας του προορισμού, αυτό πιστοποιεί το πακέτο, ο προσαρμογέας του καναλιού το τοποθετεί στην ουρά και έπειτα εκτελεί την αντίστοιχη διαδικασία. Αν έχει ζητηθεί θα δημιουργηθεί και ένα σήμα επιβεβαίωσης το οποίο και θα στείλει πίσω στον αποστολέα του πακέτου.

2.2.3 Ethernet

Εισαγωγή

Το Ethernet είναι μια οικογένεια τεχνολογιών δικτύωσης Υπολογιστών για Τοπικά Δίκτυα (LANs) βασισμένη σε πακέτα. Το όνομα του προέρχεται από τις φυσικές ιδιότητες του



Σχήμα 2.6: Η στοίβα των πρωτοκόλλων του InfiniBand.

αιθέρα (από την αγγλική λέξη “ether” ή “aether”). Μαζί με το Ethernet ορίζεται και ένας αριθμός από πρότυπα καλωδίωσης και σημάτων για το Φυσικό Επίπεδο (Physical Layer) του δικτυακού μοντέλου OSI, μέσα από τη δικτυακή πρόσβαση στο Επίπεδο Συνδέσμου Μετάδοσης Δεδομένων (Data Link Layer) καθώς και μια κοινή μορφή διευθυνσιοδότησης.

Το Ethernet έγινε το πρότυπο IEEE 802.3. Οι εκδόσεις του, που κάνουν χρήση καλωδίων συνεστραμμένων ζευγών (twisted pair) για τη σύνδεση συστημάτων στο δίκτυο και αυτές που κάνουν χρήση διαφόρων τύπων οπτικών για το δίκτυο κορμού, είναι οι πιο διαδεδομένες τεχνολογίες δικτύωσης για τοπικά δίκτυα. Χρησιμοποιείται από το 1980 έως σήμερα και έχει επικρατήσει ανάμεσα σε άλλα σημαντικά ανταγωνιστικά πρότυπα τοπικών δικτύων όπως το Token Ring, το FDDI και το ARCNET.

Γενική Περιγραφή

Το Ethernet αρχικά βασίστηκε στην ιδέα της επικοινωνίας κόμβων πάνω από ένα μοιραζόμενο ομοαξονικό καλώδιο που λειτουργούσε ως μέσο μεταφοράς εκπεμπόμενων μεταδόσεων. Οι μέθοδοι που χρησιμοποιούνταν εμφανίζουν ομοιότητες με τα ραδιοσυστήματα, αν και υπάρχουν θεμελιώδεις διαφορές όπως το γεγονός ότι είναι πολύ πιο εύκολο να ανιχνευθούν συγκρούσεις πακέτων σε μια εκπομπή στο καλώδιο από ότι σε μια ραδιοεκπομπή. Το

κοινό καλώδιο που παρέχει το κανάλι επικοινωνίας παρομοιάζεται με τον αιθέρα.

Από αυτή την πρώιμη και συγκριτικά απλή ιδέα εξελίχθηκε το Ethernet σε μια αρκετά πολύπλοκη τεχνολογία δικτύωσης που σήμερα αποτελεί βάση για τα περισσότερα τοπικά δίκτυα. Το ομοαξονικό καλώδιο αντικαταστάθηκε από σημείο–προς–σημείο συνδέσεις, οι οποίες κατέληγαν σε διανομείς (hubs) και/ή μεταγωγείς (switches) για να μειωθεί το κόστος εγκατάστασης, να αυξηθεί η αξιοπιστία, και να επιτραπεί η καλύτερη διαχείριση και αποσφαλμάτωση του δικτύου. Το StarLan ήταν το πρώτο βήμα στην εξέλιξη του Ethernet από τον ομοαξονικό δίαυλο σε ένα δίκτυο ελεγχόμενο από διανομείς, με καλώδια συνεστραμμένων ζευγών. Η έλευση του καλωδίου συνεστραμμένων ζευγών έριξε δραματικά το κόστος της εγκατάστασης σε σύγκριση με αυτό ανταγωνιστικών τεχνολογιών συμπεριλαμβανομένων και παλαιότερων εκδόσεων του ίδιου του Ethernet.

Πάνω από το φυσικό επίπεδο, οι κόμβοι του Ethernet επικοινωνούν με την ανταλλαγή πακέτων δεδομένων. Όπως και με τα υπόλοιπα IEEE 802 τοπικά δίκτυα, ο κάθε κόμβος του Ethernet έχει μια διεύθυνση MAC των 48bit, που χρησιμοποιείται για να καθοριστεί τόσο ο προορισμός όσο και η πηγή σε κάθε πακέτο δεδομένων. Οι προσαρμογείς και τα ολοκληρωμένα κυκλώματα δικτύου γενικά δεν αποδέχονται πακέτα τα οποία απευθύνονται προς άλλους κόμβους Ethernet. Οι προσαρμογείς προγραμματίζονται με μια καθολικά μοναδική διεύθυνση κατά την κατασκευή τους, οι περισσότεροι από αυτούς ωστόσο μπορούν να υποστηρίξουν τη διαχείριση της διεύθυνσης από το χρήστη.

Παρά τις σημαντικές αλλαγές στο Ethernet, από ένα χοντρό ομοαξονικό καλώδιο, δίαυλο που προσφέρει ταχύτητες μέχρι 10Mbit/s σε σημείο–προς–σημείο συνδέσεις που προσφέρουν ταχύτητες 10Gbit/s και ακόμα μεγαλύτερες, όλες οι εκδόσεις (εκτός από τις πρώτες πειραματικές) μοιράζονται την ίδια διαμόρφωση πακέτου και μπορούν εύκολα να διασυνδεθούν.

Η μορφή του πλαισίου Ethernet Το πρότυπο 802.3 ορίζει μια βασική δομή του πλαισίου που απαιτείται από όλες τις υλοποιήσεις MAC, με αρκετές επιπρόσθετες και προαιρετικές μορφές που χρησιμοποιούνται για να επεκτείνουν τις βασικές δυνατότητες του πρωτοκόλλου. Στη βασική αυτή μορφή περιλαμβάνονται επτά πεδία:

Προοίμιο (Preamble – Pre)

Αποτελείται από 7bytes. Το Pre είναι μια εναλλασσόμενη αλληλουχία από 1 και 0 που ενημερώνουν τους παραλήπτες για ένα εισερχόμενο πακέτο.

Ένδειξη Έναρξης Πακέτου (Start–of–frame delimiter – SOF)

Αποτελείται από 1byte. Το SOF είναι μια εναλλασσόμενη αλληλουχία από 1 και 0 που

τελειώνει σε 1 και καθορίζει ότι επόμενο bit είναι το πρώτο της διεύθυνσης προορισμού.

Διεύθυνση Προορισμού (Destination Address – DA)

Αποτελείται από 6bytes. Το πεδίο DA υποδεικνύει ποιοι σταθμοί πρέπει να παραλάβουν το πακέτο. Το πρώτο bit στο πεδίο DA καθορίζει αν η διεύθυνση είναι για ένα μοναδικό προορισμό (καθορίζεται με το 0) ή μια ομάδα προορισμών (καθορίζεται με το 1). Το δεύτερο bit καθορίζει αν το DA διαχειρίζεται καθολικά (καθορίζεται με το 0) ή τοπικά (καθορίζεται με το 1). Τα υπόλοιπα 46bits είναι μια μοναδική τιμή που καθορίζει ένα μοναδικό κόμβο ή μια ορισμένη ομάδα από κόμβους ή όλους τους κόμβους του δικτύου.

Διεύθυνση Πηγής (Source – SA)

Αποτελείται από 6bytes. Το πεδίο SA είναι αναγνωριστικό του κόμβου που κάνει την αποστολή. Το SA είναι πάντα διεύθυνση ενός μοναδικού κόμβου και για αυτό το πρώτο του bit είναι πάντα 0.

Μέγεθος/Τύπος (Length/Type)

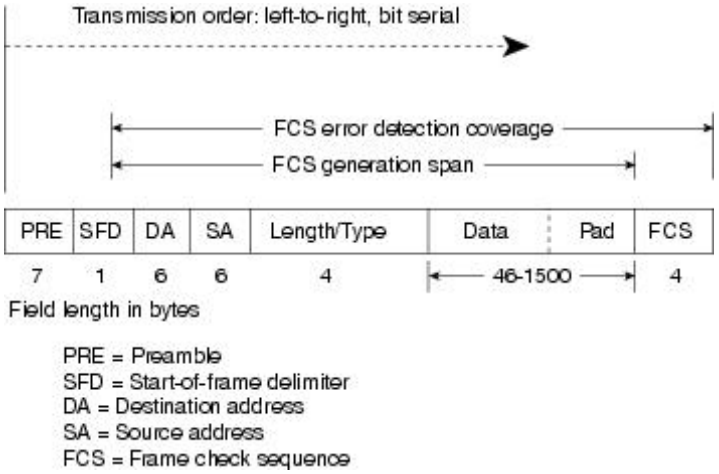
Αποτελείται από 2bytes. Αυτό το πεδίο αποτελεί είτε τον αριθμό των bytes που περιέχονται στο πεδίο των δεδομένων του πακέτου, είτε το αναγνωριστικό του τύπου του πακέτου αν αυτό ήταν φτιαγμένο με βάση μια από τις καθορισμένες προαιρετικές μορφές. Αν το πεδίο Length/Type είναι σε μια τιμή μικρότερη ή ίση με το 1500, ο αριθμός των LLC bytes στο πεδίο Δεδομένων είναι ίσος με το πεδίο Length/Type. Αν το πεδίο Length/Type είναι μεγαλύτερο από 1536 το πακέτο είναι μια προαιρετική μορφή και το πεδίο Length/Type καθορίζει ποια από τις μορφές πακέτου παραλαμβάνεται ή αποστέλλεται.

Δεδομένα (Data)

Είναι μια ακολουθία από n bytes, όπου το n είναι μικρότερο ή ίσο με το 1500. Αν το μέγεθος του πεδίου Δεδομένων είναι μικρότερο από το 46, το πεδίο Δεδομένων πρέπει να επεκταθεί προσθέτοντας κάποια byte ώστε το μήκος του να φτάσει τα 46bytes.

Ακολουθία Ελέγχου Πακέτου (Frame Check Sequence – FCS)

Αποτελείται από 4bytes. Αυτό το πεδίο περιλαμβάνει την τιμή του έλεγχου CRC μεγέθους 32bit, που υπολογίζεται από το Στρώμα Ελέγχου του Μέσου που αποστέλλει και υπολογίζεται ξανά από το αντίστοιχο που παραλαμβάνει. Το FCS δημιουργείται από τα πεδία DA, SA, Length/Type και Data.



Σχήμα 2.7: Η βασική μορφή του πακέτου Ethernet.

Το 10G Ethernet

Το 10Gbit/s Ethernet εγκρίθηκε επίσημα ως μέρος του πρότυπου IEEE 802.3 τον Ιούνιο του 2002. 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, με συνδέσεις που κυμαίνονται από τα 10Mbps μέχρι και τα 10Gbps. Αυτή η τεχνολογία είναι πολύ σημαντική όχι μόνο γιατί το Ethernet που πλέον έχει μεγάλο εύρος ζώνης, 10Gbps όχι μόνο γιατί θα εξυπηρετεί τη σύνδεση των τοπικών δικτύων σε μεγάλες ταχύτητες, αλλά πολύ περισσότερο επειδή ευνοεί την περαιτέρω υιοθέτηση του στην αγορά των συστημάτων υψηλής επίδοσης.

Η έλευση του 10GbE φέρνει μαζί της όλα τα πλεονεκτήματα του Ethernet όπως η συμβατότητα και ευκολία στη χρήση, συνδυασμένα με ένα εύρος ζώνης που μέχρι σήμερα ικανοποιεί το μεγαλύτερο μέρος της αγοράς συστημάτων υψηλής επίδοσης. Τα επόμενα χρόνια η πτώση της τιμής των υλικών του, αναμένεται να επεκτείνει με μεγαλύτερους ρυθμούς τη διάδοσή του.

Παρά τα πολλά πλεονεκτήματα του 10GbE, που τελικά αναμένεται να είναι ένα φτηνό υψηλής επίδοσης δίκτυο, όπως και οι παλιότερες του εκδόσεις έτσι και αυτό αναδεικνύει το χάσμα ανάμεσα στην υπολογιστική ισχύ που απαιτεί και στην ταχύτητά του. Ένας κανόνας αναφέρει ότι για κάθε 1Mbps δεδομένων στο καλώδιο απαιτούνται από το σύστημα 1MHz επεξεργασίας. Οι ταχύτητες στα 10Gbps θα απαιτούσαν στη λογική αυτή 10GHz επεξεργα-

σίας και για να γίνεται η επικοινωνία προς τις δύο κατευθύνσεις τελικά η απαίτηση ανεβαίνει στα 20GHz. Ακόμα περισσότεροι περιορισμοί ανακύπτουν αν κανείς λάβει υπόψη του τις επιπτώσεις στη χρήση του περιφερειακού διαύλου και στην κεντρική μνήμη σε τέτοιες ταχύτητες.

Η ραγδαία αυτή αύξηση του χάσματος της ταχύτητας μιας συσκευής E/E 10GbE σε σύγκριση με τους πόρους που απαιτούνται για τους σκοπούς της επικοινωνίας από το υπόλοιπο σύστημα, κάνει αναγκαίο τον επαναπροσδιορισμό του τρόπου που οι συσκευές χρησιμοποιούνται σε τέτοιας τάξης μεγέθους ταχύτητες.

2.3 Ο πυρήνας Λ/Σ Linux

Το Linux είναι ένας πυρήνας λειτουργικού συστήματος που μοιάζει με τον πυρήνα του AT&T UNIX. Είναι μία πρωτότυπη υλοποίηση πυρήνα λειτουργικού συστήματος και δε χρησιμοποιεί κώδικα του UNIX. Μπορεί να θεωρηθεί κλώνος του UNIX, αφού διαθέτει τις περισσότερες εντολές του, ενώ η φιλοσοφία της σχεδίασής του πλησιάζει περισσότερο το UNIX από οποιοδήποτε άλλο λειτουργικό σύστημα. Το Linux αναπτύσσεται με βάση το πρότυπο POSIX, το οποίο είναι μία προσπάθεια τυποποίησης όλων των κλώνων του UNIX.

2.3.1 Οι οδηγοί συσκευών χαρακτήρων

Ο χειρισμός των συσκευών χαρακτήρων (character devices) γίνεται μέσω ονομάτων στο σύστημα αρχείων. Αυτά τα ονόματα καλούνται ειδικά αρχεία ή αρχεία συσκευών και βρίσκονται παραδοσιακά στον κατάλογο /dev.

Οι οδηγοί χαρακτήρων αναγνωρίζονται από το μείζονα και τον ελάσσονα αριθμό (major, minor number) τους. Οι καινούργιες εκδόσεις του πυρήνα επιτρέπουν περισσότερες από μια συσκευές να μοιράζονται τον ίδιο μείζονα αριθμό, αλλά η συνηθισμένη οργάνωση είναι κάθε κατηγορία οδηγού να έχει το δικό του μείζονα αριθμό.

Ο ελάσσων αριθμός χρησιμοποιείται από τον πυρήνα για να ξεχωρίζει σε ποια συσκευή γίνεται η αναφορά. Συνηθισμένο είναι επίσης ένας οδηγός να κρατά ένα δείκτη στη συσκευή χαρακτήρων που δημιουργεί ή έναν πίνακα συσκευών, όπου ο ελάσσων αριθμός αντιστοιχεί στη θέση της συσκευής μέσα στον πίνακα.

Μέσα στον πυρήνα ορίζεται ο τύπος dev_t που μπορεί να κρατά το μείζονα και τον ελάσσονα αριθμό. Επίσης από τον πυρήνα γίνονται διαθέσιμες και οι συναρτήσεις χειρισμού μιας μεταβλητής αυτού του τύπου. Οι πιο σημαντικές λειτουργίες σε μια συσκευή χαρακτήρων

συμπεριλαμβάνονται από δομές του πυρήνα, κυρίαρχα τη stuct file_operations, την stuct file και την stuct node.

Η δομή struct file_operations

Με τη δομή struct file_operations μια συσκευή χαρακτήρων δημιουργεί τη σύνδεση μεταξύ των αριθμών και των υποστηριζόμενων λειτουργιών της συσκευής. Η δομή αυτή ορίζεται στο αρχείο κεφαλίδας <linux/fs.h> και είναι μια συλλογή από δείκτες σε συναρτήσεις. Κάθε ανοιχτό αρχείο (που αναπαρίσταται εσωτερικά με μια δομή τύπου struct file) σχετίζεται με το δικό του σύνολο συναρτήσεων. Οι λειτουργίες αυτές είναι υπεύθυνες κατά κύριο λόγο για την υλοποίηση κλήσεων συστήματος και για αυτό ονομάζονται open, close, κ.λ.π.. Το αρχείο αυτό μπορεί να θεωρηθεί ως ένα "αντικείμενο" και οι συναρτήσεις που σχετίζονται με αυτό ως οι "μέθοδοί" του, στην ορολογία του αντικειμενοστραφούς προγραμματισμού.

Παραδοσιακά μια μεταβλητή τύπου struct file_operations ή ένας δείκτης σε μια τέτοια μεταβλητή ονομάζεται fops. Κάθε πεδίο της υλοποιεί μια συγκεκριμένη λειτουργία ή μπορεί να έχει τιμή NULL, για λειτουργίες που δεν υποστηρίζονται.

Η δομή struct file

Η δομή struct file ορίζεται στο αρχείο κεφαλίδας <linux/fs.h>, και είναι η δεύτερη πιο σημαντική δομή σε μια συσκευή χαρακτήρων. Με τη δομή αυτή αναπαρίσταται ένα ανοιχτό αρχείο στο Linux και δεν αποτελεί ειδική δομή των συσκευών χαρακτήρων. Δημιουργείται από τον πυρήνα σε μία κλήση της open και περνά ως όρισμα σε κάθε συνάρτηση που κάνει κάποια λειτουργία στο αρχείο, μέχρι την κλήση της close. Όταν όλα τα στιγμιότυπα του αρχείου κλείσουν τότε ο πυρήνας ελευθερώνει το χώρο της δομής struct file.

2.3.2 Το υποσύστημα διαχείρισης της μνήμης

Οι διευθύνσεις

Το Linux είναι ένα λειτουργικό σύστημα που χρησιμοποιεί ένα εικονικό σύστημα διαχείρισης της μνήμης, υπό την έννοια ότι οι διευθύνσεις που μπορούν να δουν οι εφαρμογές δεν αντιστοιχούν στις φυσικές διευθύνσεις που χρησιμοποιούνται από το υλικό. Η εικονική μνήμη εισάγει ένα επίπεδο αναφοράς, που επιτρέπει μια σειρά από πλεονεκτήματα. Με την εικονική διαχείριση, προγράμματα τα οποία τρέχουν στο σύστημα μπορούν να δεσμεύσουν και να χρησιμοποιήσουν περισσότερη μνήμη από αυτή που είναι φυσικά διαθέσιμη. Στην

πραγματικότητα ακόμα και μια μεμονωμένη εφαρμογή μπορεί να έχει εικονικό χώρο διευθύνσεων μεγαλύτερο από τη φυσική μνήμη του συστήματος. Επίσης η εικονική μνήμη επιτρέπει κάποιους χρήσιμους χειρισμούς με το χώρο διευθύνσεων μια εφαρμογής, όπως είναι η πρόσβαση στη μνήμη μιας συσκευής.

Το Linux χρησιμοποιεί διάφορους τύπους διευθύνσεων, καθένας με διαφορετική σημασιολογική ερμηνεία. Δυστυχώς στον κώδικα του πυρήνα δεν είναι πάντοτε ξεκάθαρο ποιος τύπος διεύθυνσης χρησιμοποιείται σε κάθε περίπτωση για αυτό και ο προγραμματιστής πρέπει να έχει τον πλήρη έλεγχο. Η αδυναμία αυτή στο προγραμματιστικό περιβάλλον του Linux οφείλεται στη μη ύπαρξη διαφορετικών τύπων που να υποδεικνύουν σε τι χώρο της μνήμης γίνεται η κάθε αναφορά.

Οι τύποι των διευθύνσεων που ξεχωρίζουν στο Linux είναι οι εξής:

Εικονικές διευθύνσεις του χρήστη (User virtual addresses) Αυτές είναι οι κανονικές διευθύνσεις που χρησιμοποιούνται από εφαρμογές σε χώρο χρήστη. Μπορούν να έχουν μέγεθος 32 ή 64bit, ανάλογα με την αρχιτεκτονική του υλικού που χρησιμοποιείται. Η κάθε εφαρμογή έχει το δικό της εικονικό χώρο διευθύνσεων.

Φυσικές διευθύνσεις (Physical addresses) Οι διευθύνεις που χρησιμοποιούνται από τον επεξεργαστή και από τη μνήμη του συστήματος. Οι φυσικές διευθύνσεις είναι και αυτές τιμές των 32 ή 64bit. Ακόμα και στα συστήματα των 32bit μπορεί να χρησιμοποιούνται φυσικές διευθύνσεις των 64bit, όπως για παράδειγμα σε συστήματα που είναι 32bit αλλά οι επεξεργαστές έχουν την ικανότητα που στη βιβλιογραφία απαντάται ως Φυσική Επέκταση Διευθύνσεων (Physical Address Extension – PAE)

Διευθύνσεις Διαύλου (Bus addresses) Οι διευθύνσεις αυτές χρησιμοποιούνται μεταξύ των περιφερειακών διαύλων και της μνήμης. Πολύ συχνά ταυτίζονται με τις φυσικές διευθύνσεις που χρησιμοποιούνται από το CPU, αλλά αυτό είναι μια υπόθεση που επιβαρύνει τη φορητότητα του κώδικα. Οι διευθύνσεις διαύλου εξαρτώνται πολύ από την αρχιτεκτονική.

Λογικές διευθύνσεις του πυρήνα (Kernel logical addresses) Αυτός είναι ο κανονικός χώρος διευθύνσεων του πυρήνα. Αυτές οι διευθύνσεις αντιστοιχούν στο μεγαλύτερο μέρος αν όχι σε όλη την κεντρική μνήμη και συχνά αντιμετωπίζονται σαν να ήταν φυσικές διευθύνσεις. Στις περισσότερες αρχιτεκτονικές, οι λογικές διευθύνσεις και οι αντίστοιχες φυσικές διαφέρουν μόνο κατά μια σταθερά. Οι λογικές διευθύνσεις έχουν το μέγεθος του δείκτη του υλικού και για αυτό δεν μπορούν να διευθυνσιοδοτήσουν όλη τη φυσική μνήμη στα συστήματα 32bit. Ο χώρος που μπορεί να δεσμευτεί με την κλήση

της συνάρτησης kmalloc είναι στο λογικό χώρο διευθύνσεων του πυρήνα.

Οι εικονικές διευθύνσεις του πυρήνα (Kernel virtual addresses) Αυτές οι διευθύνσεις διαφέρουν από τις λογικές στο ότι δεν έχουν απαραίτητα μια απευθείας αντιστοίχιση σε φυσικές διευθύνσεις και μπορεί να εμφανίζουν ως συνεχόμενο ένα χώρο διάσπαρτο στη φυσική μνήμη. Όλες οι λογικές διευθύνσεις είναι και εικονικές. Η κλήση της συνάρτησης ioremap που αντιστοιχεί μνήμη στις συσκευές, καθώς και η συνάρτηση vmalloc, που δεσμεύει χώρο στην κεντρική μνήμη, επιστρέφουν εικονικές διευθύνσεις του πυρήνα.

Οι πίνακες σελιδοποίησης

Όταν ένα πρόγραμμα αναφέρεται σε μια εικονική διεύθυνση, το CPU πρέπει να τη μετατρέψει σε φυσική για να πραγματοποιήσει την πρόσβαση στη φυσική μνήμη. Η διαδικασία αυτή συνήθως γίνεται με το διαχωρισμό της διεύθυνσης σε πεδία. Το κάθε πεδίο λειτουργεί ως ένας δείκτης πίνακα, που καλείται πίνακας σελίδων και περιέχει είτε τη διεύθυνση του επόμενου πίνακα είτε τη φυσική σελίδα που αντιστοιχεί στην εικονική διεύθυνση που γίνεται η αναφορά.

Ο πυρήνας Linux διαχειρίζεται ένα σύστημα από πίνακες σελιδοποίησης τεσσάρων επιπέδων για να αντιστοιχίσει μια εικονική σε μια φυσική διεύθυνση. Τα πολλαπλά επίπεδα οργάνωσης κάνουν εφικτή τη διασπορά των φυσικών διευθύνσεων που συμπεριλαμβάνονται μέσα στο εικονικά διαθέσιμο εύρος διευθύνσεων που φαίνεται συνεχές.

Ακόμα και σε υλικό το οποίο υποστηρίζει δύο επίπεδα οργάνωσης στους πίνακες σελιδοποίησης, είτε σε υλικό το οποίο χρησιμοποιεί ένα διαφορετικό τρόπο να κάνει την αντιστοίχιση, το Linux χρησιμοποιεί τον ίδιο τρόπο οργάνωσης τεσσάρων επιπέδων. Η χρήση της οργάνωσης αυτής που υλοποιείται με έναν ανεξάρτητο τρόπο από την αρχιτεκτονική του CPU επιτρέπει τη χρήση του Linux σε κάθε CPU και τον προγραμματισμό των διάφορων μερών του με μια ενιαία αντιμετώπιση του υποσυστήματος της μνήμης. Το σημαντικό στοιχείο που χαρακτηρίζει την επιλογή αυτή είναι ότι δε συνοδεύεται από επιπλέον επιβάρυνση στα συστήματα που υποστηρίζουν λιγότερα επίπεδα οργάνωσης. Αυτό το πολύ σημαντικό χαρακτηριστικό του σχεδιασμού εξασφαλίζεται από το μεταγλωττιστή που απομακρύνει το ένα ή τα δύο επίπεδα που δεν χρειάζονται μέσω βελτιστοποιήσεων.

Η διαχείριση των εφαρμογών από το Linux βασίζεται πολύ στη σελιδοποίηση. Για την ακρίβεια, η αυτόματη μετάφραση των διευθύνσεων σε φυσικές σελίδες επιτυγχάνει τα εξής χαρακτηριστικά:

- Αποδίδονται διαφορετικοί φυσικοί χώροι διευθύνσεων σε κάθε εφαρμογή, γεγονός που εξασφαλίζει έναν αποδοτικό τρόπο προστασίας έναντι σε σφάλματα διευθύνσεων.
- Διαφοροποιούνται οι εικονικές σελίδες από τις φυσικές. Ο διαχωρισμός αυτός επιτρέπει σε μια εικονική σελίδα να αποθηκευτεί σε μια φυσική σελίδα, έπειτα όταν αυτό καταστεί αναγκαίο να αποθηκευτεί στο δίσκο και μετά να μεταφερθεί πάλι στη μνήμη σε μια διαφορετική φυσική σελίδα. Αυτό είναι και το βασικό χαρακτηριστικό ενός εικονικού συστήματος διαχείρισης της μνήμης.

2.3.3 Ο μηχανισμός των διακοπών

Οι διακοπές παρέχουν έναν τρόπο για να εκτρέπουν τον επεξεργαστή σε κώδικα έξω από την κανονική ροή ελέγχου. Όταν ένα σήμα διακοπής φτάνει, το CPU πρέπει να σταματήσει την τρέχουσα επεξεργασία και να τρέξει μία καινούργια ενέργεια. Αυτό πραγματοποιείται σώζοντας την τρέχουσα τιμή του μετρητή προγράμματος (program counter) στη στοίβα του πυρήνα και θέτοντάς του μια νέα τιμή που εξαρτάται από τον τύπο της διακοπής.

Υπάρχει μια ειδοποιός διαφορά μεταξύ του χειρισμού μιας διακοπής και της αλλαγής ελέγχου ανάμεσα σε διεργασίες: ο κώδικας που εκτελείται από μια συνάρτηση χειρισμού μιας διακοπής (Interrupt Service Routine – ISR) δεν είναι μια διεργασία. Αντίθετα είναι ένα μονοπάτι ελέγχου μέσα στον πυρήνα, που τρέχει με κόστος της ίδιας διεργασίας που έτρεχε όταν εμφανίστηκε η διακοπή. Σαν ένα μονοπάτι ελέγχου του πυρήνα, η ISR είναι πολύ ελαφρύτερη από μια διεργασία (έχει λιγότερες εντολές ενώ και η ροή ελέγχου χρειάζεται μικρότερο χρόνο για να εισέλθει καθώς και να εξέλθει από αυτή).

Ο χειρισμός μιας διακοπής εξαρτάται από τον τύπο της. Για τους σκοπούς της περιγραφής μας διαχωρίζουμε τις διακοπές σε τρεις τύπους:

Διακοπές Ε/Ε Αυτές ενεργοποιούνται όταν μια συσκευή Ε/Ε απαιτεί έναν ειδικό χειρισμό από τον πυρήνα. Η αντίστοιχη ISR πρέπει να την εξυπηρετήσει με έναν ειδικό τρόπο σύμφωνα με τις απαιτήσεις της συσκευής. Στην παρούσα εργασία αυτές είναι αποκλειστικά οι διακοπές που χρησιμοποιούνται και για το λόγο αυτό περιγράφονται διεξοδικά.

Διακοπές Ρολογιού Ένα ρολόι, είτε το τοπικό APIC ρολόι είτε ένα εξωτερικό, έχει ενεργοποιήσει μια διακοπή. Αυτός ο τύπος της διακοπής ενημερώνει τον πυρήνα ότι έχει παρέλθει ένα συγκεκριμένο χρονικό διάστημα.

Διακοπές μεταξύ επεξεργαστών Ένα CPU έχει ενεργοποιήσει μια διακοπή προς ένα άλλο CPU σε ένα σύστημα πολλών επεξεργαστών.

Σε γενικές γραμμές μια συνάρτηση χειρισμού μιας διακοπής Ε/Ε πρέπει να είναι αρκετά ευέλικτη για να αντιμετωπίσει πολλές συσκευές. Για παράδειγμα πολλές συσκευές μπορεί να μοιράζονται την ίδια γραμμή IRQ. Ως αποτέλεσμα το διάνυσμα διακοπών από μόνο του δεν περιέχει όλη την απαραίτητη πληροφορία για την ακριβή προέλευση της διακοπής.

Όταν συμβεί μια διακοπή, δεν είναι όλες οι ενέργειες που λαμβάνονται ίδιας προτεραιότητας. Για την ακρίβεια, η ISR δεν είναι το πιο κατάλληλο μέρος για κάθε ενέργεια. Μεγάλες σε διάρκεια και όχι εξαιρετικά κρίσιμες ενέργειες δεν πρέπει να γίνονται γιατί κατά τη διάρκεια που μια ISR εκτελείται, τα σήματα στην αντίστοιχη γραμμή IRQ προσωρινά αγνοούνται. Πολύ σημαντικό επίσης είναι το γεγονός, ότι η διεργασία για λογαριασμό της οποίας εκτελείται μια ISR, πρέπει να παραμένει σε ενεργή κατάσταση (TASK_RUNNING), διαφορετικά το σύστημα μπορεί να "κολλήσει". Για το λόγο αυτό οι ISR δεν μπορούν να εκτελούν blocking διεργασίες.

2.3.4 Οι ουρές εργασιών στο Linux (Workqueues)

Οι ουρές εργασιών (Work Queues – WQ) εμφανίστηκαν στον πυρήνα του Linux στις εκδόσεις 2.6. Επιτρέπουν σε συναρτήσεις του πυρήνα να ενεργοποιούνται και αργότερα να εκτελούνται από ειδικά νήματα του πυρήνα που καλούνται νήματα εργάτες.

Η κύρια δομή των ουρών εργασιών είναι η struct work_queuestruct, που ορίζεται στο αρχείο κεφαλίδας <linux/workqueue.h>. Η δημιουργία μιας τέτοιας δομής μπορεί να γίνει με τη χρήση δύο συναρτήσεων:

struct workqueue_struct *create_worqueue(**const char** *name);
struct workqueue_struct *create_singlethreaded_worqueue(**const char** *name);

Κάθε ουρά εργασιών έχει μια ή περισσότερες διεργασίες (νήματα πυρήνα), που τρέχουν όταν υποβάλλονται συναρτήσεις στην ουρά. Η χρήση της συνάρτησης create_workqueue() έχει ως αποτέλεσμα μια ουρά η οποία διαθέτει ένα νήμα σε κάθε διαθέσιμο επεξεργαστή στο σύστημα. Στα νήματα αυτά εκτελούνται οι εργασίες που εισέρχονται στην ουρά. Αν ένα μόνο νήμα είναι αρκετό για τις ανάγκες ενός οδηγού, τότε η ουρά εργασιών μπορεί να δημιουργηθεί με την κλήση της συνάρτησης create_singlethreaded_workqueue().

Για να αποδοθεί μια εργασία σε μια ουρά πρέπει να εισαχθεί σε μια δομή struct work_¬ struct. Αυτό μπορεί να γίνει σε χρόνο μεταγλώττισης με τη χρήση της μακροεντολής:

DECLARE_WORK(name, **void** (*function)(**void** *), **void** *data);

Όπου name είναι το όνομα της μεταβλητής τύπου struct struct work_struct που δηλώνεται, function είναι ο δείκτης στη συνάρτηση που καλείται από την ουρά εργασίας και data είναι

ο δείκτης σε μια μεταβλητή που περνά ως όρισμα στη συνάρτηση. Διαφορετικά κάτι τέτοιο μπορεί να γίνει και σε χρόνο εκτέλεσης από τις διαθέσιμες μακροεντολές:

INIT_WORK(**struct** work_struct *work, **void** (*function)(**void** *), **void** *data);
PREPARE_WORK(**struct** work_struct *work, **void** (*function)(**void** *), **void** *data);

Η INIT_WORK κάνει μια πιο ενδελεχή εργασία κατά την αρχικοποίηση της δομής και για αυτό χρησιμοποιείται την πρώτη φορά που δημιουργείται η δομή. Η PREPARE_WORK έχει το ίδιο σχεδόν αποτέλεσμα, αλλά δεν αρχικοποιεί του δείκτες που χρησιμοποιούνται για να συνδεθεί η δομή stuct work_struct με την ουρά εργασιών. Αν μια δομή struct work_struct έχει αποδοθεί σε μια ουρά εργασιών και χρειάζεται να εισέλθει σε κάποια άλλη ουρά τότε πρέπει να χρησιμοποιηθεί η PREPARE_WORK και όχι η INIT_WORK.

2.3.5 Οι οδηγοί προσαρμογέων δικτύου

Οι οδηγοί των προσαρμογέων δικτύου [45] είναι διαφορετικοί από τις υπόλοιπες κλάσεις οδηγών στο Linux γιατί δε χρησιμοποιούν κάποιο αρχείο στους καταλόγους /dev ή /sys για να επικοινωνήσουν με το λειτουργικό σύστημα. Αντίθετα οι εφαρμογές αλληλεπιδρούν με έναν οδηγό προσαρμογέα δικτύου διαμέσου μιας διεπαφής δικτύου. Ένα παράδειγμα είναι η eth0, για την πρώτη διεπαφή Ethernet, η οποία αποτελεί μια αφαίρεση σε μια στοίβα πρωτοκόλλων που βρίσκεται από πίσω.

Ένας οδηγός για έναν προσαρμογέα δικτύου χρησιμοποιεί τις παρακάτω δομές:

1. Δομές που διαμορφώνουν τα δομικά στοιχεία της στοίβας των πρωτοκόλλων δικτύου. Η βασική δομή είναι η struct sk_buff που ορίζεται στο αρχείο κεφαλίδας <include/¬ linux/sk_buff.h> και χρησιμοποιείται στην υλοποίηση του πυρήνα για τη στοίβα TCP/IP.

2. Δομές που ορίζουν μια διεπαφή μεταξύ του οδηγού του προσαρμογέα δικτύου και τη στοίβα πρωτοκόλλων. Η struct net_device που ορίζεται στην κεφαλίδα <include/¬ linux/netdevice.h> είναι η βασική δομή που υλοποιεί αυτή τη διεπαφή.

3. Δομές που αναφέρονται στο δίαυλο Ε/Ε. Ο δίαυλος PCI και τα παράγωγά του είναι οι πιο συχνοί δίαυλοι που χρησιμοποιούνται από τους σύγχρονους προσαρμογείς δικτύου.

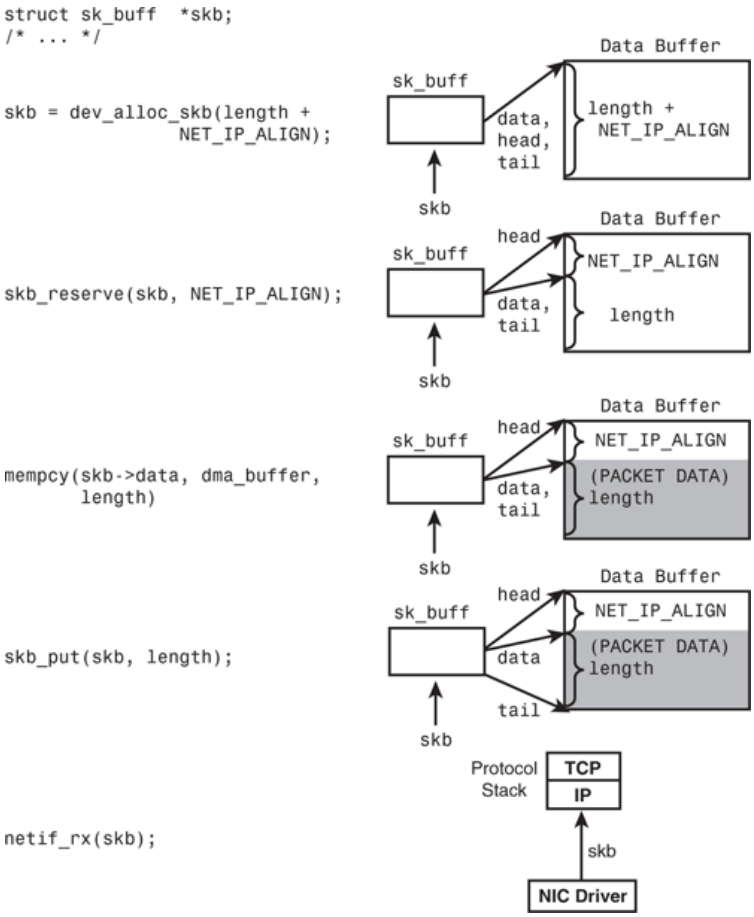
Η δομή struct sk_buff

Με τις δομές αυτές οι οδηγοί του Linux υλοποιούν αποδοτικούς μηχανισμούς χειρισμού και ελέγχου της ροής στα δικτυακά επίπεδα. Μια δομή struct sk_buff εμπεριέχει πληροφορίες ελέγχου που περιγράφουν προσωρινούς χώρους μνήμης για τη χρήση στην προετοιμασία πακέτων δικτύου. Είναι στην ουσία πολύ μεγάλες δομές που περιέχουν ένα μεγάλο αριθμό στοιχείων, η περιγραφή τους όμως θα περιοριστεί στο απαραίτητο υπόβαθρο για την κατανόηση της υπόλοιπης εργασίας. Η struct sk_buff συνδέεται με τον αντίστοιχο προσωρινό χώρο μνήμης των πακέτων χρησιμοποιώντας, κυρίαρχα, πέντε πεδία:

- head, που δείχνει στην αρχή του πακέτου.
- data, που δείχνει στην αρχή των δεδομένων που μεταφέρει το πακέτο.
- tail , που δείχνει στο τέλος των δεδομένων που μεταφέρει το πακέτο.
- end, που δείχνει στο τέλος του πακέτου.
- len, που είναι το μέγεθος των δεδομένων που μεταφέρονται από το πακέτο.

Αν skb είναι ένας δείκτης σε μία δομή struct sk_buff, τα πεδία που περιέχει η δομή skb→head, skb→data, skb→tail και skb→end, όσο αυτό περνάει από επίπεδο σε επίπεδο στη στοίβα πρωτοκόλλων του δικτύου μετακινούνται για να δείχνουν σε κατάλληλες θέσεις μέσα στον προσωρινό χώρο αποθήκευσης του πακέτου. Το skb→data για παράδειγμα δείχνει στην κεφαλίδα του τρέχοντος πρωτοκόλλου που επεξεργάζεται το πακέτο. Όταν το πακέτο φτάσει στο επίπεδο του πρωτοκόλλου IP στην πλευρά του παραλήπτη το skb→data δείχνει στην κεφαλίδα του IP. Όταν το πακέτο περνά στο επίπεδο του TCP, το skb→data μετακινείται στην αρχή της κεφαλίδας του TCP. Όσο το πακέτο συνεχίζει να διατρέχει τα διάφορα επίπεδα μιας στοίβας πρωτοκόλλων που προσθέτουν ή αφαιρούν κεφαλίδες, το skb→len αλλάζει επίσης. Η δομή struct sk_buff περιέχει και άλλους δείκτες εκτός από αυτούς που προαναφέρθηκαν. Ένα παράδειγμα είναι το skb→nh, που αποθηκεύει τη θέση του κατωτέρου στη στοίβα πρωτοκόλλου του δικτύου ανεξάρτητα από την τρέχουσα θέση του skb→data.

Στο σχήμα 2.8 φαίνονται οι αλλαγές στο μονοπάτι παραλαβής ενός πακέτου. Για λόγους διευκόλυνσης της απεικόνισης, έχει γίνει η υπόθεση ότι οι ενέργειες εκτελούνται σειριακά. Στην πραγματικότητα, για λόγους απόδοσης, τα πρώτα δύο βήματα (dev_alloc_skb() και skb_reserve()) γίνονται κατά την αρχικοποίηση ενός οδηγού όπου δεσμεύονται ένα σύνολο προσωρινών χώρων αποθήκευσης για πακέτα που παραλαμβάνονται. Το τρίτο βήμα γίνεται απευθείας από το υλικό του προσαρμογέα δικτύου με τη DMA μεταφορά πακέτων που παραλαμβάνονται σε ένα ήδη δεσμευμένο struct sk_buff. Τα δύο τελευταία βήματα (skb_put()



Σχήμα 2.8: Οι λειτουργίες σε μια δομή struct sk_buff.

και net_if_rx()) εκτελούνται από τη συνάρτηση χειρισμού της διακοπής που ενεργοποιείται από μια παραλαβή.

Για να δημιουργηθεί ένα struct sk_buff που αποθηκεύει ένα πακέτο που παραλαμβάνεται, στο σχήμα χρησιμοποιείται η dev_alloc_skb(). Αυτή είναι μια ασφαλής απέναντι σε διακοπές ρουτίνα που δεσμεύει χώρο για ένα struct sk_buff και το συσχετίζει με έναν προσωρινό χώρο αποθήκευσης των δεδομένων του πακέτου. Η dev_skb_kfree() πραγματοποιεί την αντίστροφη λειτουργία. Στο σχήμα 2.8 στο δεύτερο βήμα καλείται η skb_reserve() για να προσθέσει ένα παραγέμισμα (pad) μεγέθους 2 byte μεταξύ της αρχής του προσωρινού χώρου του πακέτου και του χώρου των δεδομένων. Με τον τρόπο αυτό η κεφαλίδα του IP ξεκινά σε μια θέση με την οποία επιτυγχάνεται η αποδοτικότερη επεξεργασίας της. Δεδομένου ότι η κεφαλίδα του Ethernet είναι 14bytes με την προσθήκη 2bytes, εξασφαλίζεται η

ευθυγράμμιση της κεφαλίδας στα 16bytes. Οι υπόλοιπες λειτουργίες στο σχήμα γεμίζουν τον προσωρινό χώρο με τα δεδομένα του πακέτου και αλλάζουν κατάλληλα τα πεδία skb→data, skb→tail και skb→len.

Στην προγραμματιστική διεπαφή του Linux υπάρχουν πολλές ακόμα σχετικές συναρτήσεις. Για παράδειγμα η skb_clone(), δημιουργεί ένα αντίγραφο ενός struct sk_buff χωρίς να αντιγράφει τα δεδομένα του σχετιζόμενου προσωρινού χώρου αποθήκευσης του πακέτου. Μέσα στο αρχείο <net/core/skbuff.c> μπορούν να βρεθούν οι περισσότερες από τις συναρτήσεις χειρισμού των struct sk_buff.

Η δομή struct net_device

Οι οδηγοί των καρτών δικτύων χρησιμοποιούν μια κοινή διεπαφή με την οποία αλληλεπιδρούν με τη στοίβα TCP/IP. Η δομή struct net_device, η οποία είναι ακόμα μεγαλύτερη από την struct sk_buff, ορίζει τη διεπαφή επικοινωνίας. Παρόμοια με τα βήματα στο σχήμα 2.8, κατά την αρχικοποίηση ενός οδηγού ενός προσαρμογέα δικτύου ακολουθούνται τα εξής βήματα:

- Ο οδηγός δεσμεύει χώρο για ένα struct net_device με τη συνάρτηση alloc_netdev(). Πιο συχνά χρησιμοποιείται μια πιο κατάλληλη συνάρτηση που κάνει με τη σειρά της κλήση της alloc_netdev(). Ένας οδηγός για έναν προσαρμογέα Ethernet, για παράδειγμα, χρησιμοποιεί την alloc_etherdev(). Ένας οδηγός για έναν προσαρμογέα ασύρματου δικτύου χρησιμοποιεί την alloc_ieee80211(). Όλες αυτές οι συναρτήσεις δέχονται ως όρισμα το μέγεθος ενός ιδιωτικού χώρου και δεσμεύουν το χώρο αυτό μαζί με το χώρο για τη δομή struct net_device:

  ```
  struct net_device *netdev;
  struct priv_struct *mycard_priv;
  netdev = alloc_etherdev(sizeof(struct priv_struct));
  /* The private space allocated by alloc_etherdev() */
  mycard_priv = netdev→priv;
  ```

- Ο οδηγός υπολογίζει διάφορα πεδία στο struct net_device που δεσμεύτηκε και το καταχωρεί στο επίπεδο δικτύου με την κλήση της register_netdev(netdev).
- Ο οδηγός διαβάζει τη διεύθυνση MAC από τη μνήμη EEPROM του προσαρμογέα δικτύου και ρυθμίζει το Wake–On–Lan (WOL) αν αυτό χρειάζεται. Οι προσαρμογείς Ethernet συνήθως έχουν μια μη μεταβλητή EEPROM για να κρατά πληροφορίες όπως είναι η MAC διεύθυνση και το πρότυπο του WOL. Η διεύθυνση MAC όπως έχει προ-

αναφερθεί είναι μια τιμή μεγέθους 48bit που είναι καθολικά μοναδική. Το πρότυπο του WOL είναι μια "μαγική" ακολουθία, η οποία αν αυτή βρεθεί μέσα σε ένα πακέτο τότε ο προσαρμογέας δικτύου ενεργοποιείται, αν βρισκόταν σε ανενεργή κατάσταση.

- Αν ο προσαρμογέας χρειάζεται λογισμικό για να λειτουργήσει, ο οδηγός το μεταφορτώνει χρησιμοποιώντας τη συνάρτηση request_firmware().

2.3.6 Δίκτυα υψηλής επίδοσης στο Linux.

Η εξέλιξη του Ethernet σε σχέση με τα υπόλοιπα δίκτυα υψηλής επίδοσης, καθώς και η απλότητα της δικτυακής στοίβας στον πυρήνα του Linux, δίνει τη δυνατότητα να αναπτυχθούν πρωτόκολλα ανταλλαγής μηνυμάτων χωρίς τη χρήση εξειδικευμένου υλικού. Ένα από αυτά είναι το Open-MX που περιγράφεται παρακάτω:

Open-MX

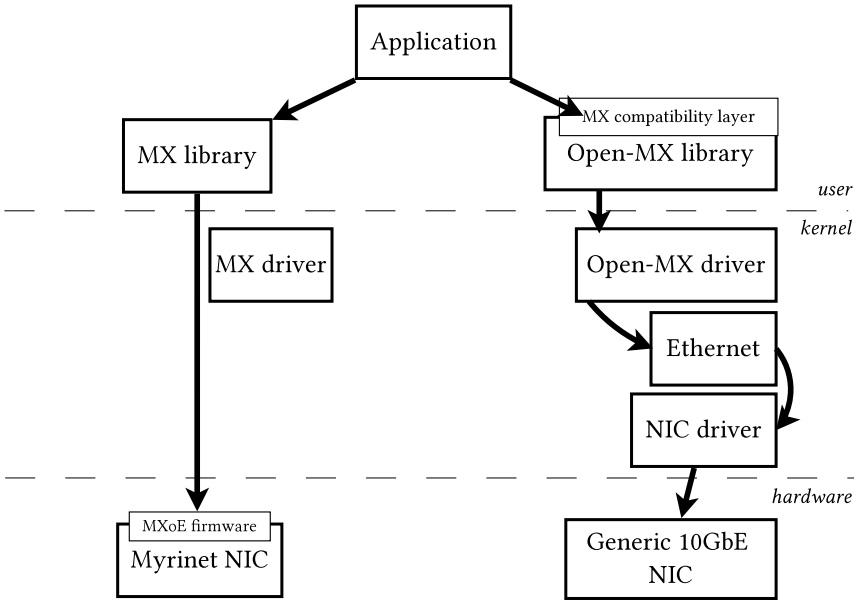
Το Open-MX είναι μια υλοποίηση του πρωτοκόλλου Myrinet/MX σε συμβατικές κάρτες δικτύου Ethernet. Δεν απαιτεί εξειδικευμένο hardware, αφού η επεξεργασία του πρωτοκόλλου γίνεται αποκλειστικά στον πυρήνα του Linux. Ακολουθεί το ίδιο μοντέλο μηνυμάτων όπως το MX· η βασική τους διαφορά έγκειται στο γεγονός ότι δεν μπορεί να χρησιμοποιήσει τη δυνατότητα παράκαμψης του λειτουργικού συστήματος (αφού το Open-MX χρησιμοποιεί κοινές κάρτες δικτύου). Η σημασιολογία είναι ακριβώς ίδια με αυτή του MX και τα μηνύματα κατηγοριοποιούνται με τον ίδιο τρόπο σε SMALL, MEDIUM και LARGE. Τα βασικά στοιχεία που απαρτίζουν το Open-MX είναι τα εξής:

Endpoints: Όπως και στο MX, ένα endpoint είναι ο δίαυλος επικοινωνίας μεταξύ της εφαρμογής που χρησιμοποιεί το πρωτόκολλο και της κάρτας δικτύου. Περιέχει όλη την απαραίτητη πληροφορία που υλοποιεί την επικοινωνία: ουρές αποστολής λήψης, χειριστές γεγονότων για την επικοινωνία με το χώρο χρήστη καθώς και πληροφορίες για τις περιοχές μνήμης (στοιχεία των σελίδων που τις απαρτίζουν κλπ).

Events: Οι εφαρμογές που χρησιμοποιούν το πρωτόκολλο, επικοινωνούν με τη βιβλιοθήκη και το module του πυρήνα, μέσω του μηχανισμού των events. Ουσιαστικά πρόκειται για μια μέθοδο επικοινωνίας του χώρου χρήστη με το χώρο πυρήνα και αναφέρεται σε ειδοποιήσεις για λήψεις μηνυμάτων και επιβεβαιώσεις για αποστολές. Επιπλέον, τα events ενεργοποιούν ειδοποιήσεις για λήξη χρονικών ορίων αποστολής, σχετικές με το φυσικό μέσο (επαναποστολές) ή ακόμα και ενημερώσεις για τυχόν προβλήματα που δημιουργούνται στο ίδιο το πρωτόκολλο.

Regions: Οι περιοχές μνήμης είναι σύνολα από κομμάτια (segments) που περιέχουν λογικά συνεχόμενες σελίδες μνήμης (virtually contiguous pages). Αυτές οι σελίδες δεσμεύονται από την εφαρμογή και με κατάλληλες κλήσεις στον οδηγό προσαρτώνται στην ανάλογη περιοχή. Οι περιοχές μπορούν να αποτελέσουν πηγή ή προορισμό ενός μηνύματος και συνήθως χρησιμοποιούνται για τα μηνύματα τύπου LARGE (με χρήση σημασιολογίας rendezvous).

Το μονοπάτι των δεδομένων από την εφαρμογή προς το δίκτυο χρησιμοποιώντας το Open-MX φαίνεται στο Σχήμα 2.9 (δεξιά), σε αντιπαράθεση με το Myrinet/MX (αριστερά).



Σχήμα 2.9: Αρχιτεκτονική του MX και του συμβατικού Open-MX.

2.4 Πλατφόρμες Virtualization

2.4.1 Εικονικές Μηχανές

Εικονική μηχανή είναι η υλοποίηση μιας μηχανής σε λογισμικό, που εκτελεί προγράμματα όπως μια φυσική μηχανή [47]. Οι Popek και Goldberg [12] έδωσαν τον εξής ορισμό: "Εικονική μηχανή θεωρείται ένα αποδοτικό και απομονωμένο αντίγραφο μιας πραγματικής μηχανής". Ωστόσο η έννοια αυτή σήμερα περιλαμβάνει υλοποιήσεις που δεν αποτελούν αντί-

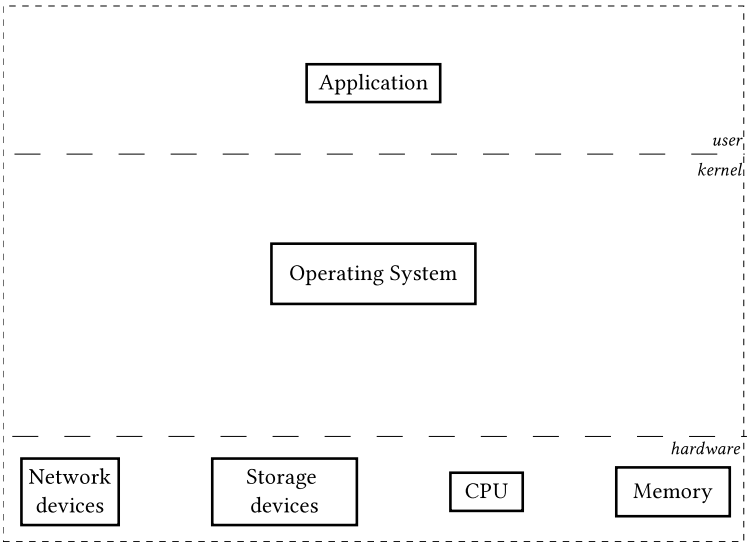
γραφα πραγματικών μηχανών. Πάνω σε αυτό το διαχωρισμό βασίζεται η κατηγοριοποίηση των εικονικών μηχανών σε εικονικές μηχανές συστήματος και εικονικές μηχανές διεργασίας.

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 υλικού. 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.

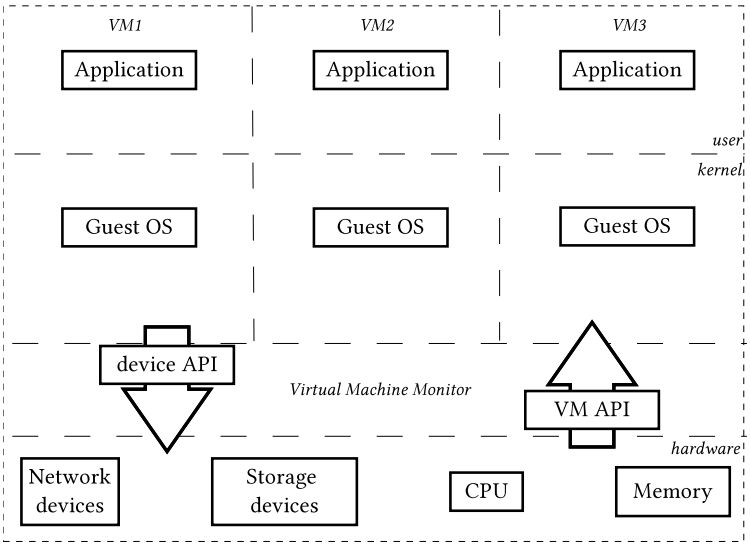
Από την άλλη πλευρά, η εικονική μηχανή συστήματος είναι ουσιαστικά ένα υπολογιστικό σύστημα εμφωλευμένο μέσα σε ένα άλλο υπολογιστικό σύστημα και λειτουργεί σαν να του ανήκουν όλοι οι πόροι του συστήματος, αν και την ίδια στιγμή μπορεί να συνυπάρχει με ένα ή και περισσότερα συστήματα. Πιο συγκεκριμένα, 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, τη διεύθυνση MAC της κάρτας δικτύου καθώς και τη διεύθυνση IP των διεπαφών της. Επίσης 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.

Η ταυτόχρονη συνύπαρξη των εικονικών μηχανών επιτυγχάνεται με χρήση του ”ελεγκτή εικονικής μηχανής” (virtual machine monitor - VMM). Ο ελεγκτής εικονικής μηχανής είναι ουσιαστικά το σύστημα που πολυπλέκει την πρόσβαση στο υλικό για κάθε εικονική μηχανή μεσολαβώντας στην επικοινωνία των λειτουργικών συστημάτων των εικονικών μηχανών με το υλικό. Μερικές από τις λειτουργίες του είναι η διασφάλιση του απομονωμένου περιβάλλοντος εκτέλεσης των εικονικών μηχανών, η πολύπλεξη των αιτημάτων για πρόσβαση στους πόρους του συστήματος από τα φιλοξενούμενα λειτουργικά συστήματα και η αντιστοίχηση των διακοπών υλικού σε διακοπές λογισμικού έτσι ώστε να ειδοποιείται το

κατάλληλο φιλοξενούμενο λειτουργικό σύστημα για διάφορα συμβάντα του υλικού.



Σχήμα 2.10: Συμβατικό υπολογιστικό σύστημα



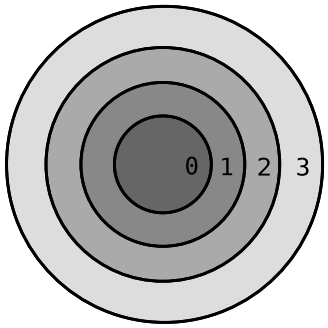
Σχήμα 2.11: Ελεγκτής εικονικών μηχανών

Στο Σχήμα 2.10, αναπαριστάται η ιεραρχική τοποθέτηση των τμημάτων του λογισμικού σε σχέση με το υλικό ενός υπολογιστή που δεν περιέχει εικονικό περιβάλλον. Σε αυτή τη περίπτωση το λειτουργικό σύστημα συνδέεται άμεσα με το υλικό, καθώς περιέχει τους οδη-

γούς 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 υλικού. Με άλλα λόγια, το λειτουργικό σύστημα κατέχει τον πλήρη έλεγχο του συστήματος και βρίσκεται, όπως λέγεται, στο δακτυλίδι 0 (ring 0).

Στο Σχήμα 2.11 παρουσιάζεται η παραπάνω ιεραρχία σε εικονικό περιβάλλον. Σε αυτή 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 2.10 και επιπλέον οργανώνει την εκτέλεση των εικονικών μηχανών που δημιουργούνται. Μέρος της οργάνωσης αυτής είναι ο τρόπος με τον οποίο οι εικονικές μηχανές επικοινωνούν με το υλικό αλλά και ο τρόπος με τον οποίο αυτές 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 υλικό [10].

Τα σύγχρονα λειτουργικά συστήματα δεν επιτρέπουν στις εφαρμογές χώρου χρήστη να εκτελούν λειτουργίες αυξημένων δυνατοτήτων. Για παράδειγμα, μόνο το λειτουργικό σύστημα μπορεί να φορτώσει οδηγούς συσκευών ή να προσπελάσει απευθείας το υλικό. 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 (2.12).



Σχήμα 2.12: CPU privilege levels

Η επεξεργαστική μονάδα εκτελεί εντολές 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 επίπεδα 1 και 2 παραμένουν αχρησιμοποίητα. Όμως, σε έναν υπολογιστή με εικονικό περιβάλλον, ο ελεγκτής εικονικής μηχανής θα πρέπει να μπορεί να προσπελάσει 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 κάποιους από 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.

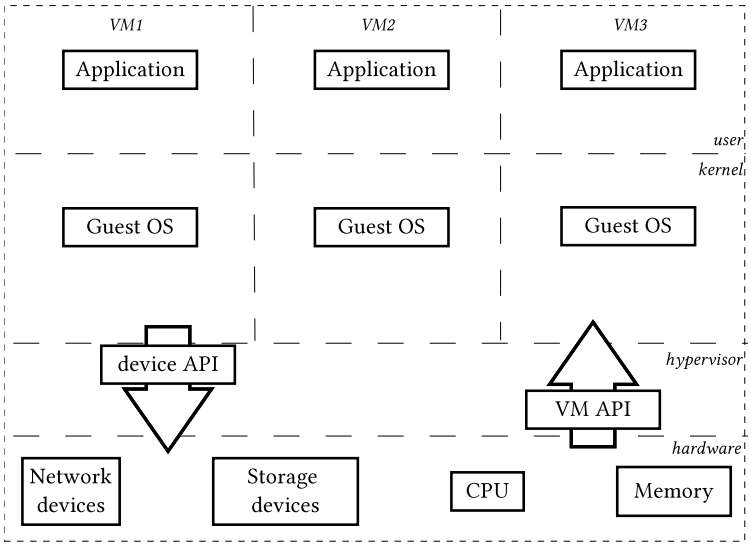
Ο τρόπος με τον οποίο δομείται ένα εικονικό περιβάλλον δεν είναι μοναδικός και εξαρτάται κυρίως από τον σχεδιασμό του ελεγκτή εικονικής μηχανής. Οι δύο βασικές παράμετροι σχεδιασμού είναι οι εξής:

- η άμεση επαφή του ελεγκτή εικονικής μηχανής με το υλικό (σε αντίθεση με την παρεμβολή ενός λειτουργικού συστήματος που θα πολυπλέκει την πρόσβαση)
- η πλήρης προσομοίωση του υλικού για πρόσβαση από τις εικονικές μηχανές

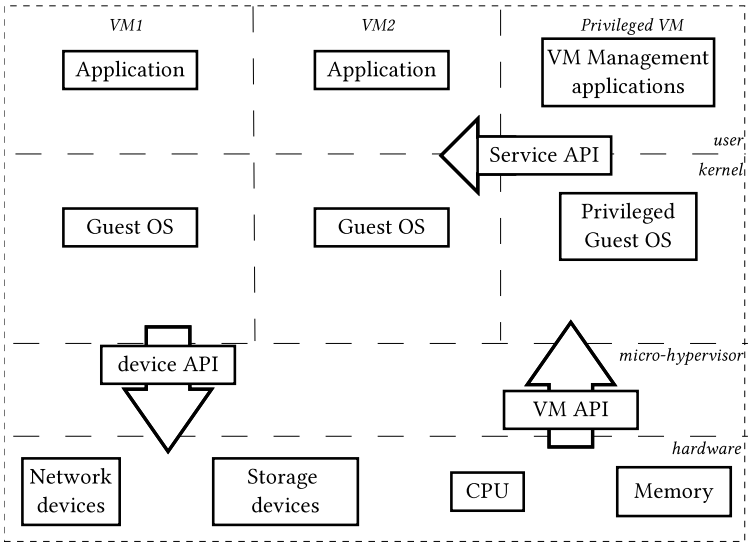
Βάσει των παραπάνω παραμέτρων, προκύπτουν δύο τύποι virtualization: ο τύπος I και ο τύπος II. Στον τύπο I ανήκουν οι ελεγκτές εικονικής μηχανής που τουλάχιστον ένα μέρος τους βρίσκεται σε άμεση επαφή με το υλικό. Σε αυτόν τον τύπο υπάρχουν δύο υποτύποι ελεγκτών εικονικής μηχανής: ο επιβλέπων (Hypervisor), που εκτελείται εξολοκλήρου πάνω στο υλικό (Σχήμα 2.13) και το λειτουργικό σύστημα υπηρεσίας (Service OS), που ουσιαστικά λειτουργεί ως μια εικονική μηχανή αυξημένων δυνατοτήτων (Σχήμα 2.14).

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 Ε/Ε, 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 2.15 παρουσιάζεται η δομή του.

Όσο απομακρύνεται το λογισμικό του ελεγκτή από το υλικό, τόσο μικραίνει η εξάρτησή του από αυτό και τόσο μεγαλώνει η φορητότητά του. Αυτό 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 I θα πρέπει να επικοινωνούν με επιτυχία με τις

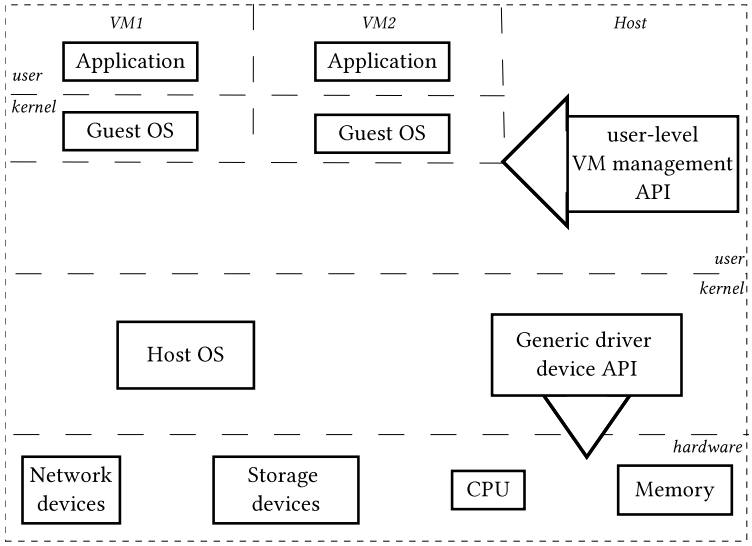


Σχήμα 2.13: Hypervisor



Σχήμα 2.14: Service OS

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 υλικό και έτσι η φορητότητά τους σε άλλο υλικό θα είναι περιορισμένη. Στον ελεγκτή τύπου Ι, οι οδηγοί έχουν μεταφερθεί στο λειτουργικό σύ-



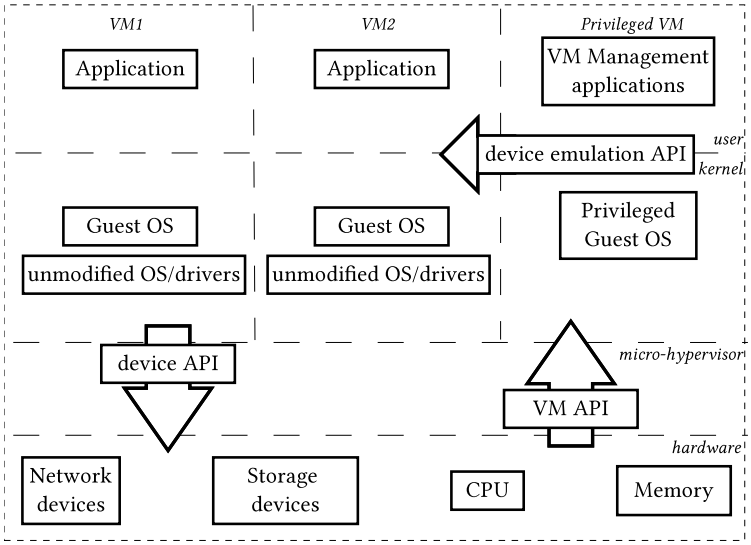
Σχήμα 2.15: Host OS

στημα υπηρεσίας, με το οποίο πρέπει να επικοινωνούν τα υπόλοιπα φιλοξενούμενα λειτουργικά συστήματα έτσι ώστε να αποκτήσουν πρόσβαση στο υλικό. Το λειτουργικό σύστημα υπηρεσίας, όμως, είναι ένα κοινό, ελαφρώς τροποποιημένο λειτουργικό σύστημα. Αυτό σημαίνει ότι ως προς την επικοινωνία με το υλικό, 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 ΙΙ 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 υλικό, τόσο αυξάνεται η ευελιξία.

Όμως, ταυτόχρονα αυξάνεται η πολυπλοκότητα του συστήματος και μειώνεται η απόδοσή του. Αυτό συμβαίνει διότι προστίθενται επιπλέον επίπεδα ανάμεσα σε μία εικονική μηχανή και στο πραγματικό υλικό. Στον τύπο ΙΙ, η εικονική μηχανή συνομιλεί απευθείας με τον ελεγκτή και τους οδηγούς. Στον τύπο Ι, η εικονική μηχανή συνομιλεί πρώτα με την την εικονική μηχανή υπηρεσίας και έπειτα αυτή επικοινωνεί με το υλικό. Τέλος, στον τύπο ΙΙ, η εικονική μηχανή επικοινωνεί με τον ελεγκτή, ο ελεγκτής με τον ελεγκτή επιπέδου χρήστη

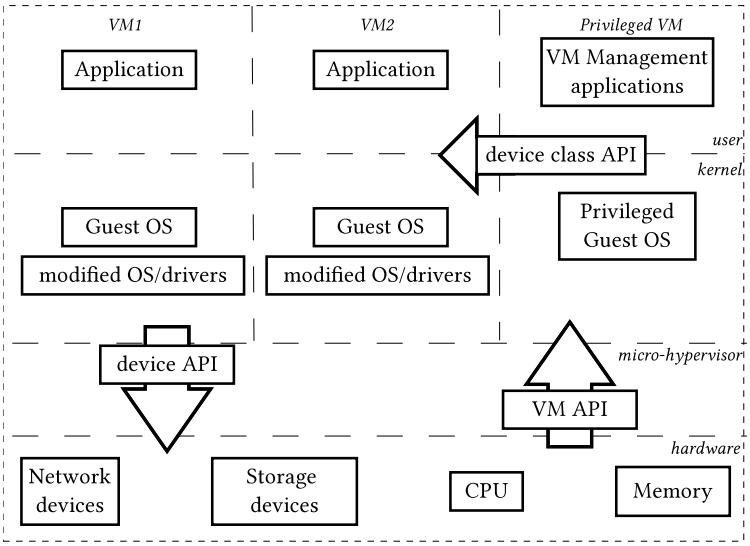
κι εκείνος με το υλικό.

Δημιουργούνται έτσι δύο κατηγορίες τεχνικών virtualization: η *πλήρης* (full virtualization) και η *μερική* (paravirtualization). Οι τεχνικές της πρώτης κατηγορίας (Σχήμα 2.16), απαιτούν ελεγκτές που προσομοιώνουν πλήρως το υλικό προς τη πλευρά των εικονικών μηχανών, ενώ οι τεχνικές της δεύτερης κατηγορίας (Σχήμα 2.17) απαιτούν ελεγκτές που παρουσιάζουν εικονικούς οδηγούς στη πλευρά των εικονικών μηχανών και οργανώνουν την συνομιλία τους με τους πραγματικούς.



Σχήμα 2.16: Full virtualization

Στην τεχνική πλήρους virtualization, όπως αναφέρθηκε παραπάνω, ο ελεγκτής προσομοιώνει πλήρως το υλικό. Αυτό σημαίνει ότι χρησιμοποιεί το λογισμικό για να προσομοιώσει τις λειτουργίες του υλικού 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 την περίπτωση λοιπόν, η λειτουργία του ελεγκτή είναι να ανακατευθύνει και να οργανώνει τις κλήσεις των οδηγών των φιλοξενούμενων λειτουργικών συστημάτων στις πραγματικές συσκευές. Η τεχνική αυτή επιτρέπει στις εικονικές μηχανές να έχουν μη τροποποιημένα λειτουργικά συστήματα και μη τροποποιημένους οδηγούς. Δηλαδή, θα μπορούσε να φιλοξενηθεί οποιοδήποτε Λ/Σ, αφού θα δημιουργείται η εντύπωση στο Λ/Σ ότι εκτελείται πάνω σε πραγματικό υλικό και όχι μέσα σε εικονικό περιβάλλον. Από την άλλη πλευρά, η τεχνική αυτή παρουσιάζει μειωμένη επίδοση, αφού η διαδικασία της προσομοίωσης του υλικού είναι απαιτητική σε πόρους και χρονοβόρα.



Σχήμα 2.17: Paravirtualization

Από την άλλη πλευρά, με το paravirtualization, 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 του 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 κοστίζει σε χρόνο εκτέλεσης καθώς και υλοποίησης. Η τεχνική paravirtualization μπορεί να αποφευχθεί, αν ο επεξεργαστής, το υποσύστημα Ε/Ε καθώς και η εκάστοτε συσκευή Ε/Ε του υπολογιστικού συστήματος υποστηρίζουν στο υλικό επεκτάσεις virtualization (όπως τα extended instruction sets των επεξεργαστών Intel/AMD, τα υποσυστήματα IOMMU, καθώς και οι συσκευές που υποστηρίζουν I/O Virtualization).

Συνδυάζοντας τις τέσσερις παραπάνω κατηγοριοποιήσεις των ελεγκτών εικονικής μηχανής δημιουργούνται τέσσερα σύνολα ελεγκτών στα οποία ανήκουν σύγχρονα και δημοφιλή εικονικά περιβάλλοντα. Στον πίνακα 2.1 παρουσιάζονται αντιπροσωπευτικά παραδείγματα των συνόλων αυτών.

Ο ελεγκτής εικονικής μηχανής UML είναι γνωστός από το λειτουργικό σύστημα Linux,

Πίνακας 2.1: Απλουστευτική ταξινόμηση ελεγκτών εικονικών μηχανών

	Τύπος I	Τύπος II
Fully-virtualized	VMWare ESX	KVM
Para-virtualized	Xen	UML

αφού αποτελεί εργαλείο δοκιμής λογισμικού που προορίζεται για τον πυρήνα του λειτουργικού, ενώ ο ελεγκτής VMware ESX είναι ένας από τους πιο δημοφιλείς εμπορικούς VMMs. Στη συνέχεια, θα αναλύσουμε περαιτέρω την αρχιτεκτονική του Xen και του KVM.

2.4.2 KVM

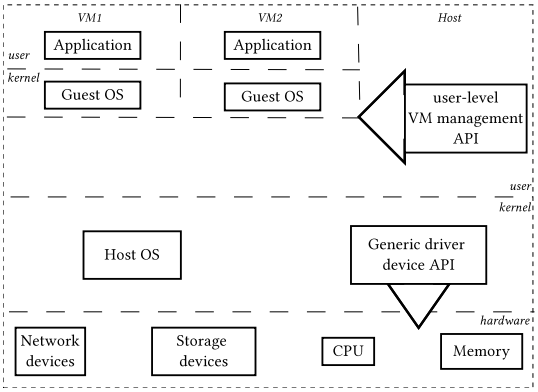
Το KVM (Kernel-based Virtual Machine) είναι το πρώτο σύστημα virtualization που ενσωματώθηκε στον πυρήνα του λειτουργικού συστήματος Linux. Το KVM αρχικά αναπτύχθηκε από τη Qumranet [22], η οποία εξαγοράστηκε από την Redhat το Σεπτέμβριο του 2008. Σήμερα είναι ο προκαθορισμένος ελεγκτής εικονικής μηχανής στο Redhat Enterprise Linux (RHEL)[17].

Το KVM χρησιμοποιεί την τεχνική πλήρους virtualization για την εκτέλεση των εικονικών μηχανών. Ο πυρήνας του κώδικα είναι σχετικά μικρός, αφού έχει σχεδιαστεί να εκμεταλλεύεται πλήρως τις επεκτάσεις για virtualization που διαθέτουν οι σύγχρονοι επεξεργαστές. Το σύστημα αυτό υποστηρίζει αρχιτεκτονικές x86, καθώς και IA64, IBM s390 κλπ.

Αρχιτεκτονική

Στο σύστημα του KVM, τον ρόλο του ελεγκτή εικονικής μηχανής τον παίζει το ίδιο το λειτουργικό Linux, καθώς αυτό διαθέτει πολλούς από τους μηχανισμούς που χρειάζεται για να υποστηριχθούν οι εικονικές μηχανές (χρονοδρομολόγηση, διαμοιρασμός πόρων, διασφάλιση απομονωμένης εκτέλεσης). Οι επιπρόσθετες λειτουργίες που χρειάζονται για την υποστήριξη virtualization αποτελούν το KVM, το οποίο προσαρτάται στον πυρήνα του λειτουργικού μέσω μιας ενός kernel module. Με την προσάρτηση αυτή, το λειτουργικό σύστημα Linux μετατρέπεται σε ελεγκτή εικονικής μηχανής (Σχήμα 2.18).

Όταν φορτώνεται η μονάδα του KVM, εμφανίζεται στο σύστημα αρχείων η συσκευή χαρακτήρων /dev/kvm, που αναπαριστά τη διεπαφή του KVM. Επιτρέπει τον έλεγχο του εικονικού περιβάλλοντος μέσω ενός συνόλου κλήσεων IOCTLs. 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.



Σχήμα 2.18: KVM

Σε ένα συμβατό περιβάλλον Linux, κάθε διεργασία εκτελείται είτε σε κατάσταση χρήστη (user-mode) είτε σε κατάσταση πυρήνα (kernel-mode). Το σύστημα KVM εισάγει μια τρίτη κατάσταση εκτέλεσης, που ονομάζεται κατάσταση φιλοξενούμενου (guest-mode), στην οποία εκτελούνται τα φιλοξενούμενα λειτουργικά συστήματα (εικονικές μηχανές).

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 υλικού.

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 αντιστοιχίζει στο χώρο διευθύνσεων που προορίζεται για κάθε εικονική μηχανή (Ενότητα 2.3.2). 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 αντιστοίχιση καθώς και την αποθήκευση των διευθύνσεων που ανήκουν σε κάθε εικονική μηχανή. Για να επιτευχθεί αυτό, προστίθεται ένα ακόμα επίπεδο αφαίρεσης στη σελιδοποίηση της μνήμης (και στην κατάρτιση πινάκων αντιστοίχησης) που ονομάζεται shadow page table.

Στα μοντέρνα λειτουργικά συστήματα εκτελούνται πολύ περισσότερες διεργασίες από

τον αριθμό των επεξεργαστών που διατίθενται. Αυτό καθίσταται δυνατό λόγω του χρονοδρομολογητή που διαθέτουν, ο οποίος καθορίζει τη σειρά με την οποία κάθε διεργασία θα ανατίθεται στους διαθέσιμους επεξεργαστές καθώς και το χρόνο που κάθε διεργασία θα μπορεί να εκτελείται. Στο KVM, κάθε εικονική μηχανή νοείται ως μία διεργασία και χρονοδρομολογείται μαζί με τις υπόλοιπες διεργασίες του λειτουργικού.

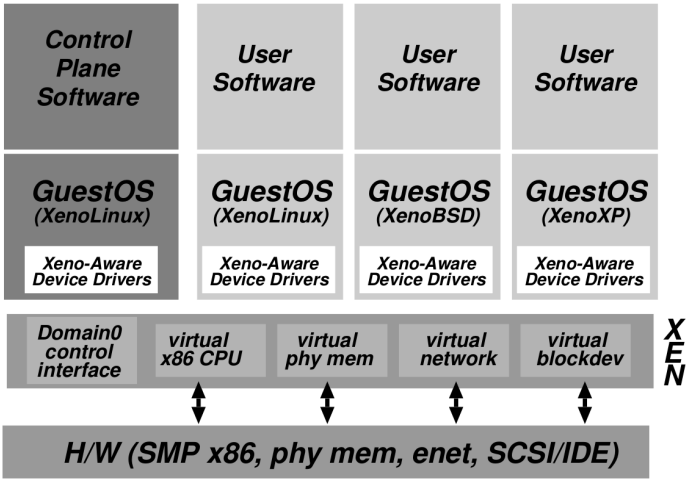
Τέλος, το σύστημα KVM για να παρέχει προσομοιωμένες συσκευές υλικού στις εικονικές μηχανές, όπως σκληρό δίσκο, οδηγούς CD και κάρτες δικτύου, χρησιμοποιεί ένα τροποποιημένο λογισμικό QEMU [4]. Το λογισμικό αυτό είναι ένα εργαλείο virtualization επιπέδου χρήστη, το οποίο επιτρέπει την προσομοίωση ενός υπολογιστικού συστήματος. Για κάθε εικονική μηχανή, μια διεργασία QEMU ξεκινάει σε κατάσταση χρήστη και παρέχει τις προσομοιωμένες συσκευές. Όταν μια εικονική μηχανή εκτελεί μια εντολή Ε/Ε, λαμβάνεται από το KVM και ανακατευθύνεται στην αντίστοιχη διεργασία QEMU.

Ένα από τα βασικά πλεονεκτήματα του συστήματος KVM είναι ότι βασίζεται σε μεγάλο βαθμό στον πυρήνα του Linux και έτσι επωφελείται άμεσα από τις βελτιστοποιήσεις που εφαρμόζονται σε αυτόν. Επίσης, εφαρμόζοντας την τεχνική πλήρους 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 το migration των εικονικών μηχανών ανάμεσα σε συστήματα με διαφορετικό υλικό και παρέχει πλήρη απομόνωση των διάφορων εφαρμογών αυξάνοντας την ασφάλεια του συστήματος. Από την άλλη πλευρά, λόγω της προσομοίωσης του υλικού μειώνεται η ταχύτητα επικοινωνίας με αυτό και απομακρύνεται από τη φυσική επίδοση του συστήματος. Γι' αυτό το λόγο, εφαρμόζονται βελτιστοποιήσεις (ειδικά στο χειρισμό των συσκευών Ε/Ε) που βασίζονται σε λύσεις para-virtualization.

2.4.3 Xen

Αρχιτεκτονική

Στο Σχήμα 2.19 παρουσιάζεται η αρχιτεκτονική του Xen. Το Xen είναι ελεγκτής τύπου I και, πιο συγκεκριμένα, ανήκει στον υποτύπο "λειτουργικό σύστημα υπηρεσίας". Το ρόλο της εικονικής μηχανής υπηρεσίας τον παίζει το driver domain, μια εικονική μηχανή που ονομάζεται επίσης και Domain 0 (Dom0), ενώ η εικονικές μηχανές $VM_{1..N}$ είναι κοινές εικονικές μηχανές (DomUs), που επικοινωνούν με το Dom0 για τη προσπέλαση στο υλικό. 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



Σχήμα 2.19: Η δομή της αρχιτεκτονικής του Xen

στα άλλα να δημιουργεί και να τερματίζει εικονικές μηχανές, να δημιουργεί εικονικές διεπαφές δικτύου και να διαχειρίζεται τη πρόσβαση των εικονικών μηχανών στο υλικό.

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. Για μια εικονική μηχανή, μια υπερκλήση είναι ότι είναι η κλήση συστήματος για μια εφαρμογή.

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 balloon 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 φιλο-

ξενούμενα λειτουργικά συστήματα, τα ίδια δημιουργούν την ψευδαίσθηση ότι κατέχουν συνεχόμενη φυσική μνήμη (ψευδο-φυσική μνήμη, pseudo-physical memory). 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 ισχύος, το Xen χρησιμοποιεί τον αλγόριθμο χρονοδρομολόγησης Borrowed Virtual Time (BVT) [23]. 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 μοντέλο του διαχωρισμένου οδηγού συσκευών (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:

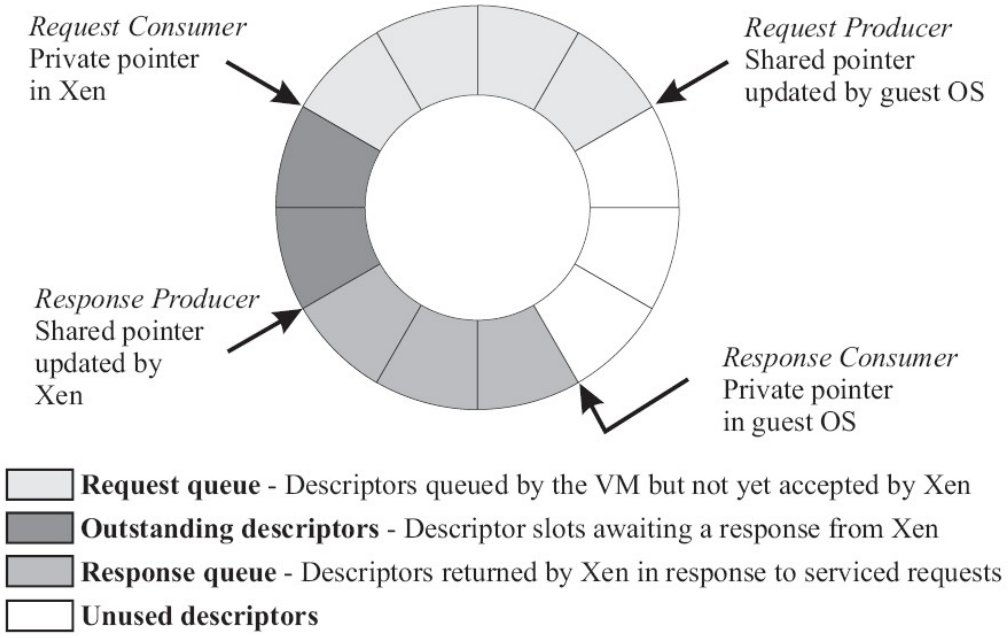
- Τον πραγματικό οδηγό (native driver).
- Το κάτω μισό του διαχωρισμένου οδηγού (backend driver).
- Τους μοιραζόμενους αποθηκευτικούς χώρους σε δομή δακτυλιδιού (ring buffers).
- Το πάνω μισό του διαχωρισμένου οδηγού (frontend driver).

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 υλικό, όπως η ανάγνωση και η εγγραφή 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.

Επικοινωνία μεταξύ εικονικών μηχανών

Απαραίτητο στοιχείο της υλοποίησης του μοντέλου διαχωρισμένου οδηγού είναι η επικοινωνία μεταξύ του driver domain και των εικονικών μηχανών. 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 2.20: Η δομή του Xen Ring

Η δομή του δακτυλιδιού είναι μια κυκλική ουρά με τέσσερις δείκτες και δυο ειδών τύπου δεδομένων ως στοιχεία της (Σχήμα 2.20). 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.

Για την επικοινωνία μεταξύ των εικονικών μηχανών χρησιμοποιείται ο μηχανισμός *XenStore*. Το XenStore έχει μια δεντρική μορφή, στα φύλλα των οποίων υπάρχουν πεδία με τις αντίστοιχες τιμές τους. Τα πεδία αυτά αποτελούν παραμέτρους λειτουργίας των εικονικών μηχανών και είναι ορατά από όλα τα μέρη, βάσει ενός πίνακα δυνατοτήτων πρόσβασης. Επίσης κάθε εικονική μηχανή 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, για την ενημέρωση και την εγκατάσταση διακοπών τόσο στο Dom0 όσο και στα DomUs δημιουργείται ένα σύστημα ενημέρωσης μεταξύ τους, 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 του μερικού virtualization. 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 παρεμβάλλονται ανάμεσα σε αυτά και το υλικό. Με άλλα λόγια υπάρχει δυνατότητα άμεσης πρόσβασης στο υλικό, ειδικά αν αυτό υποστηρίζει εικονικά περιβάλλοντα.

3

Σχετική βιβλιογραφία

Η παρούσα διατριβή περιγράφει ένα πλαίσιο που προσφέρει αποδοτικό διαμοιρασμό πόρων Ε / Ε σε εικονικά περιβάλλοντα με έμφαση στη μεταφορά δεδομένων από / προς το δίκτυο, χωρίς τη χρήση εξειδικευμένου υλικού. Έτσι είναι σχετική με εργασίες από το χώρο των συστημάτων επικοινωνίας υψηλής επίδοσης, την αποδοτική χρήση μοιραζόμενων αρχιτεκτονικών πόρων σε εικονικά περιβάλλοντα, συστήματα μοιραζόμενης πρόσβασης σε συσκευές δικτύου και προγραμματιζόμενες δικτυακές αρχιτεκτονικές.

Με βάση την πρόσφατη ανάπτυξη των τεχνολογιών του virtualization, η επιβάρυνση στην εκτέλεση εφαρμογών σε εικονικά περιβάλλοντα που σχετίζεται με την επεξεργαστική μονάδα ή το διαμοιρασμό της μνήμης ελαχιστοποιείται [50, 51, 52], δίνοντας τη δυνατότητα στις εφαρμογές να επιτυγχάνουν επίδοση ισάξια της αντίστοιχης εκτέλεσης σε φυσικά μηχανήματα. Παρόλα αυτά, η επίδοση σε Ε / Ε παρουσιάζει εντελώς διαφορετικά χαρακτηριστικά: τα ενδιάμεσα επίπεδα του virtualization προσδίδουν σημαντική επιβάρυνση όταν περισσότερα του ενός VMs μοιράζονται δικτυακές ή αποθηκευτικές συσκευές [29, 31, 42].

Η ανάπτυξη του 10G Ethernet και η εκτεταμένη χρήση του ως δίκτυο διασύνδεσης υψηλής επίδοσης έδωσε βήμα σε ένα μεγάλο μέρος ερευνητικών πονημάτων για τη βελτιστοποίηση πρωτοκόλλων ανώτερου επιπέδου, και συγκεκριμένα, βελτιστοποιήσεις που αφορούν τη μείωση της επίδρασης του χειρισμού και της επεξεργασίας των πρωτοκόλλων [41, 19, 13, 43]. Μερικές από τις βελτιστοποιήσεις αφορούν στην παράκαμψη του λειτουργικού συστήματος για την επικοινωνία, το χειρισμό της επικοινωνίας από απομονωμένο πυρήνα επεξεργασίας κλπ. Βάσει αυτών των ευρημάτων επιτυγχάνεται ένα βελτιστοποιημένο μονοπάτι, δίνοντας σημασιολογία δικτύων υψηλής επίδοσης σε εφαρμογές, χωρίς τη χρήση εξειδικευμένου υλικού, όπως το 10G Ethernet.

3.1 Βελτιστοποιήσεις στο λογισμικό

Προηγούμενες εργασίες που αφορούν στη μείωση της επιβάρυνσης των ενδιάμεσων επιπέδων του virtualization έχουν εστιάσει κυρίως στο paravirtualization (μερικό virtualization). Σε αυτό το περιβάλλον, οι εντολές που δεν απαιτούν πρόσβαση σε Ε / Ε (και ουσιαστικά είναι κοινός τόπος για εφαρμογές υψηλών απαιτήσεων), μπορούν να εκτελούνται απευθείας στην επεξεργαστική μονάδα. Ο Menon [29] προτείνει βελτιστοποιήσεις στην αρχιτεκτονική του Xen για δικτυακή πρόσβαση, προσθέτοντας νέες δυνατότητες στην εικονική διεπαφή δικτύου (scatter/gather I/O, TCP/IP checksum offload, TCP segmentation offload). Ο Ram [38] βελτιώνει το μηχανισμό παραχώρησης μνήμης, ενώ ο Dong [8] προτείνει την επέκταση του χρονοδρομολογητή του VMM για απόκριση πραγματικού χρόνου (real-time response support). Οι συγγραφείς στις εργασίες [42] και [39] παρουσιάζουν βελτιστοποιήσεις στη δικτυακή αρχιτεκτονική του Xen που αφορούν στη χρήση της μνήμης, ενώ ο Muliben Yehuda [5] προτείνει βελτιστοποιήσεις στη στοίβα του TCP/IP. Τέλος, στο [25] οι συγγραφείς προτείνουν ένα framework στο οποίο πακέτα ομαδοποιούνται για μεταδίδονται στο δίκτυο μαζικά για το KVM [22]. Οι προηγούμενες εργασίες παρουσιάζονται ιδανικές για λύσεις σε υπηρεσιοστρεφείς εφαρμογές· όμως, για εφαρμογές υψηλών απαιτήσεων, η στοίβα πρωτοκόλλων TCP/IP επιβάλει σημαντικές επιβαρύνσεις όταν χρησιμοποιείται για ανταλλαγή μηνυμάτων.

Σε αντίθεση με τις προηγούμενες βελτιστοποιήσεις, ο Liu [27] περιγράφει το VMM-bypass I/O πάνω από Infiniband. Η προσέγγιση της εργασίας είναι καινοτόμα και βασίζεται στο διαχωρισμένο μοντέλων οδηγών του Xen. Στο [32], οι συγγραφείς παρουσιάζουν το σχεδιασμό ενός παρόμοιου framework χρησιμοποιώντας δικτυακό εξοπλισμό Myrinet. Πολλά χαρακτηριστικά αυτών των μελετών μπορούν να χρησιμοποιηθούν από καινούριες πλατφόρμες virtualization για να ξεπεραστούν τα όρια στη ρυθμαπόδοση και το χρόνο απόκρισης που προκαλούνται από επιβάρυνση στο λογισμικό.

Η ερευνητική κοινότητα έχει εστιάσει και σε βελτιστοποιήσεις που αφορούν την ενδοεπικοινωνία των εικονικών μηχανών που βρίσκονται στο ίδιο φυσικό μηχάνημα: στην εργασία [7], οι συγγραφείς παρουσιάζουν μια βιβλιοθήκη μοιραζόμενης μνήμης που χρησιμοποιείται για ανταλλαγή μηνυμάτων μεταξύ εικονικών μηχανών στο KVM, χωρίς τη μεσολάβηση φυσικού μέσου μετάδοσης. Επιτυγχάνουν εντυπωσιακά αποτελέσματα, εξαλείφοντας πλήρως την επιβάρυνση των επιμέρους επιπέδων virtualization. Ο Huang [18] σχεδιάζει μια βιβλιοθήκη επικοινωνίας με βάση το MPI και αξιολογεί την επίδοσή της. Στην εργασία αυτή, δείχνει ότι ο συνδυασμός μιας βιβλιοθήκης για εικονικό περιβάλλον και βελτιστοποιημένων

μονοπατιών (με βάση το VMM-bypass I/O) προσδίδει ελάχιστη επιβάρυνση στην εκτέλεση εφαρμογών υψηλών απαιτήσεων σε εικονικά περιβάλλοντα.

3.2 Βελτιστοποιήσεις στο υλικό

Για να επιτευχθεί απευθείας επικοινωνία μεταξύ του υλικού και των εικονικών μηχανών, κάποιες εργασίας μελέτησαν βελτιστοποιήσεις στο υλικό: Ο Raj [37] περιγράφει τη δημιουργία συσκευών που αυτο-διαμοιράζονται, ενώ ο Mansley [20] προτείνει τη χρήση επιταχυντών για πρόσβαση σε συσκευές δικτύου. Ο Dong [9] παρουσιάζει την ενσωμάτωση της τεχνολογίας IOV στο Xen· τα αποτελέσματα της εργασίας δείχνουν ότι οι εικονικές μηχανές μπορούν να έχουν απομονωμένη και ασφαλή πρόσβαση στο υλικό. Τέλος ο Auernhammer [3] προτείνει βελτιώσεις στην αρχιτεκτονική των επεξεργαστικών μονάδων υποστήριξη εικονικών διεπαφών. Συγκεκριμένα εξομοιώνουν μια διεπαφή δικτύου παρόμοια με αυτές του Infiniband για να μελετήσουν μηχανισμούς ειδοποίησης και μεταφράσεις διευθύνσεων στο μονοπάτι αποστολής.

3.3 Βελτιστοποιήσεις σε επιμέρους επίπεδα του λειτουργικού συστήματος

Εκτός από προσεγγίσεις που αφορούν στο υλικό, πολλές εργασίες ασχολήθηκαν με βελτιστοποιήσεις στην ενσωμάτωση τεχνικών IOV στις πλατφόρμες virtualization. Ο Gordon [14] μελετά την επιβάρυνση των διακοπών στην κίνηση δικτύου εικονικών μηχανών στο KVM και παρουσιάζει μια προσέγγιση που διακοπές χωρίς έξοδο (exitless interrupts) οδηγούν σε σημαντική βελτίωση στη ρυθμαπόδοση του δικτύου. Η εργασία αυτή είναι συμπληρωματική της [49] όπου οι συγγραφείς βελτιστοποιούν την τεχνική αντιστοίχησης συσκευών σε VMs για αποδοτικότερη απόδοση των μεταφορών με DMA.

Παρόλα αυτά, αυτές οι προσεγγίσεις παρουσιάζουν δύο βασικά μειονεκτήματα:

- (i) οι τεχνικές αυτές βασίζονται στο υλικό για να καταφέρουν υψηλή ρυθμαπόδοση, επομένως ο αριθμός των εικονικών μηχανών που μπορούν να μοιράζονται μια συσκευή περιορίζεται από τα όρια του υλικού.
- (ii) παρουσιάζουν περιορισμένη ή και καθόλου ευελιξία (flexibility), ένα από τα σημαντικότερα πλεονεκτήματα του virtualization. Εξαιτίας της μειωμένης ευελιξίας, η μεταφορά εικονικών μηχανών γίνεται αρκετά δυσκολότερη λόγω της ετερογένειας του

υλικού μεταξύ περιοχών σε ένα κέντρο δεδομένων. Επίσης, καθίσταται δυσκολότερο να επικοινωνούν οι εικονικές μηχανές που συνυπάρχουν στο ίδιο φυσικό μηχάνημα, χωρίς περαιτέρω αλλαγές είτε στο υλικό, είτε στο λογισμικό που ελέγχει την πρόσβαση στις συσκευές.

Για τη βελτιστοποίηση της δικτύωσης σε περιβάλλοντα paravirtualization, ο Gordon [15] παρουσιάζει σημαντικές αλλαγές στη στοίβα του KVM για να ελαχιστοποιηθεί η επιβάρυνση που προκαλείται από μηνύματα ενημέρωσης μεταξύ των επεξεργαστικών μονάδων. Παρόλα αυτά, η εργασία αυτή περιορίζεται στη στοίβα TCP/IP και δεν παρουσιάζει συγκρίσιμα αποτελέσματα με τη δική μας προσέγγισηΕντούτοις, σε συνδυασμό με το Xen2MX, μπορεί να ελαχιστοποιήσει την επιβάρυνση στις επεξεργαστικές μονάδες που σχετίζεται με την επικοινωνία των εικονικών μηχανών με το driver domain.

Παρόλα τα πλεονεκτήματα της απόδοσης των τεχνικών IOV, πρόσφατες εργασίες σε δικτύωση υψηλής επίδοσης με βάση το paravirtualization επιβεβαιώνουν την πορεία προς λύσεις χωρίς τη χρήση εξειδικευμένου υλικού. Ο Ranadive [40] παρουσιάζει ένα πλαίσιο σε λογισμικό που είναι παρόμοιο με το VMM-bypass I/O [27] · συγκεκριμένα, αναπτύσσει μια εικονική διεπαφή που προσφέρει σημασιολογία RDMA σε εικονικές μηχανές για VMWare ESXi.

4

Μεταφορά δεδομένων στο δίκτυο

Οι υπολογιστικές συστοιχίες (clusters) χρησιμοποιούνται όλο και συχνότερα για την παροχή υπολογιστικής ισχύος σε πληθώρα εφαρμογών από διάφορα επιστημονικά πεδία.

Το κυριότερο γνώρισμα μιας τέτοιας σχεδίασης είναι ο διαμοιρασμός πόρων σε διάφορα επίπεδα: πυρήνες μοιράζονται επίπεδα της ιεραρχίας κρυφών μνημών, επεξεργαστές μοιράζονται εύρος ζώνης στον διάδρομο συστήματος, περιφερειακές συσκευές ανταγωνίζονται τους επεξεργαστές για πρόσβαση στη μνήμη κλπ. Ο διαμοιρασμός πόρων απλοποιεί τη σχεδίαση και διευκολύνει τον προγραμματισμό του συστήματος, ωστόσο μπορεί να έχει σημαντική επίπτωση στην επίδοση του, ανάλογα με το είδος των υπό εκτέλεση εφαρμογών.

Εφαρμογές με καλή τοπικότητα αναφορών περνούν μεγάλο τμήμα του χρόνου εκτέλεσής τους χρησιμοποιώντας δεδομένα στην κρυφή μνήμη, ενώ εφαρμογές με μεγάλες απαιτήσεις για δεδομένα επιβαρύνουν το μονοπάτι προς την κύρια μνήμη και, καθώς ο όγκος των δεδομένων αυξάνεται, το μονοπάτι προς συσκευές αποθήκευσης. Σε συστοιχίες υπολογιστών, η πρόσβαση σε αποθηκευτικά μέσα παρέχεται συχνά μέσω ενός δικτύου αποθήκευσης (Storage Area Network) ή μέσω του δικτύου διασύνδεσης της συστοιχίας.

Αφενός η ανάγκη για εκτέλεση εφαρμογών απαιτητικών σε δεδομένα κι αφετέρου η συνεχής αύξηση της διαθέσιμης υπολογιστικής ισχύος ανά επεξεργαστή λόγω του αυξανόμενου αριθμού πυρήνων, αναδεικνύουν τη σημασία του συστήματος Ε / Ε στη δυνατότητα κλιμάκωσης του συνολικού συστήματος. Η υποδομή πρέπει να αντεπεξέλθει στην ανάγκη των πυρήνων για δεδομένα [16]. Χρειαζόμαστε μηχανισμούς για την αποδοτική μετακίνηση μεγάλου συνόλου δεδομένων ανάμεσα σε αποθηκευτικά μέσα και υπολογιστικούς πυρήνες, μέσω ενός δικτύου διασύνδεσης, με ελάχιστη επιβάρυνση στη λειτουργία του συστήματος. Η παράμετρος αυτή ενισχύεται ακόμα περισσότερο όταν η εκτέλεση των εφαρμογών και η μεταφορά των δεδομένων (ανάκτηση ή αποθήκευση) γίνονται σε εικονικά περιβάλλοντα.

Για την παροχή κλιμακούμενης αποθηκευτικής υποδομής σε συστοιχίες χρησιμοποιούνται συχνά συστήματα για μοιραζόμενη χρήση αποθηκευτικών συσκευών σε επίπεδο μπλοκ

(block-level storage sharing). Ένα τέτοιο σύστημα επιτρέπει σε έναν αριθμό από κόμβους-πελάτες να έχουν πρόσβαση σε απομακρυσμένους δίσκους κόμβων-εξυπηρετητών, ως τοπικούς. Η αρχή λειτουργίας του είναι η εξής: αιτήσεις Ε/Ε για μπλοκ μετασχηματίζονται σε δικτυακά μηνύματα, τα οποία περνούν στον εξυπηρετητή. Τέτοια συστήματα είναι γνωστά και ως συστήματα δικτυακών συσκευών μπλοκ ή συστήματα nbd (network block devices).

Ιδανικά, ένα σύστημα nbd παρέχει ένα πολύ λεπτό στρώμα πρόσβασης σε απομακρυσμένες συσκευές, κλιμακούμενο με τον αριθμό τους, με ελάχιστη επιβάρυνση στους πελάτες και τους εξυπηρετητές. Σε σύγχρονα συστήματα, η ανάγκη αυτή μεταφράζεται σε μειωμένο φόρτο CPU και μειωμένο ανταγωνισμό για μοιραζόμενους πόρους. Η πρόσβαση στη μνήμη από περιφερειακές συσκευές επιτείνει το πρόβλημα · σύγχρονα δίκτυα προσφέρουν ρυθμούς της τάξης των 40-80Gbps, αυξάνοντας την πίεση για πρόσβαση στη μνήμη και παρεμποδίζοντας τον υπολογισμό στους τοπικούς επεξεργαστές, με εμφανή επιβάρυνση στην επίδοση. Ένα σύστημα nbd οφείλει να στοχεύει στην ελαχιστοποίηση του κόστους απομακρυσμένης πρόσβασης σε δεδομένα, που αυξάνεται σημαντικά λόγω του ανταγωνισμού για υπολογιστικό χρόνο και εύρος ζώνης σε μοιραζόμενα μονοπάτια.

Τα σύγχρονα συστήματα nbd υστερούν από την άποψη αυτή. Συχνά βασίζονται στο TCP/IP, οπότε εισάγουν σημαντική επεξεργασία στην CPU του κόμβου. Επιπλέον, αντιμετωπίζουν την κύρια μνήμη ως κεντρικό πόρο: το μονοπάτι των δεδομένων διασχίζει πολλές φορές το διάδρομο μνήμης και τους περιφερειακούς διαδρόμους, ακόμη και όταν χρησιμοποιούνται προηγμένα δίκτυα διασύνδεσης με δυνατότητα Απομακρυσμένης Απευθείας Πρόσβασης στη Μνήμη [21, 28, 26]. Έτσι, επιβαρύνουν σημαντικά τους συμμετέχοντες κόμβους, ενώ η απόδοσή τους περιορίζεται λόγω κορεσμού των διαδρόμων Ε / Ε και της δικτυακής πρόσβασης.

Με βάση τα παραπάνω, γίνεται σαφές ότι η εκτέλεση εφαρμογών σε συστοιχίες επηρεάζεται σημαντικά από την επίδοση του υποσυστήματος Ε / Ε. Γι'αυτό το λόγο, είναι καθοριστικό να μελετηθούν εναλλακτικά μονοπάτια μεταφοράς δεδομένων που δε θα επιβαρύνουν το σύστημα και δε θα μειώνουν την επίδοση των εφαρμογών. Μια πρώιμη προσέγγιση εύρεσης ενός αποδοτικού τρόπου μεταφοράς δεδομένων δίνεται από το σύστημα gmblock, ένα σύστημα διαμοιρασμού αποθηκευτικού χώρου που εκμεταλλεύεται τη διαθεσιμότητα επεξεργαστικής ισχύος και μονάδων μνήμης στον προσαρμογέα δικτύου για την κατασκευή απευθείας μονοπατιού δεδομένων ανάμεσα στο αποθηκευτικό μέσο και το δίκτυο. Η υλοποίησή του πάνω από Myrinet επιτρέπει την άμεση μετακίνηση δεδομένων από το δίσκο στο δίκτυο, χωρίς ενδιάμεσο αντίγραφο στη μνήμη του κόμβου και χωρίς την εμπλοκή του επεξεργαστή του. Η προσέγγιση αυτή μετριάζει τον ανταγωνισμό για μοιραζόμενους πόρους,

βελτιώνει τη δυνατότητα κλιμάκωσης και επιτρέπει αύξηση του ρυθμού μεταφοράς έως και δύο φορές, σε σχέση με τις καθιερωμένες τεχνικές. Επιπλέον, ο αποθηκευτικός κόμβος δεν είναι απαραίτητο να χρησιμοποιείται αποκλειστικά για την εξυπηρέτηση αιτήσεων Ε / Ε· μπορεί να λειτουργεί επιπρόσθετα ως υπολογιστικός κόμβος, καθώς η απομακρυσμένη Ε / Ε δεν παρεμβαίνει στην εξέλιξη του τοπικού υπολογισμού. Η σχεδίαση του gmblock συνδυάζει υπάρχοντες μηχανισμούς αφαίρεσης που παρέχονται από το ΛΣ και το δίκτυο διασύνδεσης, επιτρέποντας την κατασκευή του μονοπατιού με γενικό τρόπο, χωρίς αλλαγές συγκεκριμένες για την υφιστάμενη αρχιτεκτονική. Έτσι είναι ανεξάρτητη του είδους της αποθηκευτικής συσκευής και διατηρεί τους μηχανισμούς απομόνωσης και προστασίας μνήμης του λειτουργικού συστήματος.

Η αρχική υλοποίηση ξεπερνά περιορισμούς εύρους ζώνης στο διάδρομο κύριας μνήμης και στον περιφερειακό διάδρομο, ωστόσο υπολείπεται των δυνατοτήτων του αποθηκευτικού μέσου και του δικτύου διασύνδεσης, λόγω του παράλληλου τρόπου μεταφοράς δεδομένων από αποθηκευτικά μέσα RAID και του περιορισμένου ποσού μνήμης που διαθέτει η κάρτα δικτύου. Για την καλύτερη προσαρμογή του gmblock στην υφιστάμενη αρχιτεκτονική και την υποστήριξη μεγαλύτερων αιτήσεων Ε/Ε χωρίς εμπλοκή του επεξεργαστή του κόμβου, προτείνουμε μια νέα κατηγορία λειτουργιών αποστολής πάνω από το Myrinet, που υποστηρίζουν συγχρονισμό· η σημασιολογία τους επιτρέπει στη δικτυακή μεταφορά να εξελίσσεται ελεγχόμενα, παράλληλα με τη μεταφορά δεδομένων από το δίσκο, επικαλύπτοντας τις δύο φάσεις για την ίδια αίτηση Ε/Ε. Η ενσωμάτωσή τους στο σύστημα βελτιώνει περαιτέρω το ρυθμό μεταφοράς που επιτυγχάνεται συγκριτικά με τη βασική έκδοση. Στην πλευρά του πελάτη, βασιζόμαστε στην προγραμματισιμότητα του προσαρμογέα δικτύου και προτείνουμε τεχνικές μεταφοράς δεδομένων μπλοκ από το δίκτυο σε διάσπαρτες περιοχές φυσικής μνήμης χωρίς αντίγραφο και χωρίς εμπλοκή της CPU. Σε συνδυασμό με το προτεινόμενο μονοπάτι από την πλευρά του εξυπηρετητή, η σχεδίαση αυτή επιτρέπει τη μετακίνηση δεδομένων με μηδενικά αντίγραφα από άκρο σε άκρο. Η εγκατάσταση ενός παράλληλου συστήματος αρχείων, του Oracle OCFS2 πάνω από την υποδομή επιτρέπει την αξιολόγηση της επίδοσης με πραγματικά μετροπρογράμματα. Βρίσκουμε ότι η χρήση του άμεσου μονοπατιού γενικά ευνοεί την επίδοση, με την προϋπόθεση ότι το υποσύστημα αποθηκευτικών μονάδων παρέχει επαρκή ρυθμό μεταφοράς ώστε να μην γίνεται η στενωπός του συστήματος.

4.1 Πρωτόκολλα δικτυακής διασύνδεσης

Στις μέρες μας, η κυρίαρχη προγραμματιστική διεπαφή για υπάρχουσες στοίβες πρωτοκόλλων επικοινωνίας ανταλλαγής μηνυμάτων είναι το MPI (Message-Passing Interface). Εκτός από τη βασική λειτουργία (πέρασμα μηνυμάτων), το MPI προσφέρει καθολική επικοινωνία (collective communication), μια σημαντική μέθοδο που χρησιμοποιούν επιστημονικές εφαρμογές όπως αυτές πολλαπλασιασμού πινάκων, πολυ-διανυσματικές μέθοδοι επίλυσης εξισώσεων κλπ.

Ένα στρώμα ανταλλαγής μηνυμάτων δεν είναι απαραίτητο να υλοποιεί αυτές τις μεθόδους από την αρχή – είναι απαραίτητο να χτιστεί μια ”αφαίρεση” του MPI που ονομάζεται BTL (Byte-transfer layer) και αναλαμβάνει τη μετάφραση της σημασιολογίας του MPI σε πραγματικές κλήσεις του πρωτοκόλλου διασύνδεσης. Ένα διάσημο πρωτόκολλο κατώτερου επιπέδου που υποστηρίζει τις δυνατότητες δικτύωσης που προσφέρει το MPI και λειτουργεί ως BTL για σχεδόν όλες της υλοποιήσεις του είναι το Myrinet eXpress (MX [30]).

Εφαρμογές που είναι απαιτητικές σε επικοινωνία μπορούν να απολαύσουν σημαντική βελτίωση της επίδοσής τους μέσω εξειδικευμένων δικτύων υψηλής επίδοσης. Παρόλα αυτά, οι περισσότερες εφαρμογές δεν αντιμετωπίζουν συμφόρηση λόγω δικτύου, και γι’αυτό το λόγο, πολλές συστοιχίες (περισσότερες από τις μισές στο Top500) χρησιμοποιούν τεχνολογίες δικτύωσης όπως το gigabit Ethernet. Αυτή η κατηγορία παράλληλων εφαρμογών είναι συχνά συνδεδεμένη με υλοποιήσεις του MPI πάνω από TCP/IP. Το TCP/IP παρουσιάζει μειωμένη επίδοση στο πλαίσιο του High-performance Computing. Σαν αποτέλεσμα, η ερευνητική κοινότητα εξετάζει εναλλακτικά πρωτόκολλα επικοινωνίας πάνω από απλό Ethernet, χωρίς δηλαδή τη μεσολάβηση του TCP/IP.

4.1.1 Επιλογές σχεδίασης για πρωτόκολλα ανταλλαγής μηνυμάτων

Ένας από τους κυριότερους στόχους της παρούσας διατριβής είναι η ενσωμάτωση σημασιολογίας δικτύων υψηλής επίδοσης σε περιβάλλοντα εικονικών μηχανών. Αυτό υπονοεί ότι το σύστημα που αναπτύσσουμε πρέπει να υπακούει σε συγκεκριμένους κανόνες: πρέπει να είναι *εύκολα προσαρμόσιμο*, *απλό*, *κλιμακώσιμο* και *στιβαρό* [2]. Πρέπει επίσης να επιτυγχάνει υψηλή επίδοση τόσο όσον αφορά στη ρυθμαπόδοση, όσο και στο χαμηλό χρόνο απόκρισης της επικοινωνίας. Σε ότι ακολουθεί, εξηγούμε κάθε έναν από τους στόχους σχεδίασης.

Εύκολα προσαρμόσιμο: Οι αναλυτές συστημάτων σπάνια διαθέτουν τους απαραίτητους πόρους για να προσαρμόσουν τον κώδικά τους σε διάφορα υποσυστήματα δικτύωσης. Η

προσκόλληση σε συγκεκριμένες προγραμματιστικές διεπαφές επιβραδύνει την υιοθέτηση καινοτόμων και βελτιωμένων τεχνολογιών. Επομένως, η προφανής επιλογή ενός υποσυστήματος δικτύωσης είναι τέτοια που να βασίζεται σε μια διάσημη προγραμματιστική διεπαφή.

Ταυτόχρονα, πολλοί από τους κατασκευαστές υλικού δεν προσφέρουν στην κοινότητα υλικό που διαθέτει εξειδικευμένα χαρακτηριστικά, εκτός αν υπάρχουν καλές πιθανότητες να πιστωθούν μερίδιο της αγοράς. Για όποιο καινούριο χαρακτηριστικό, ειδικά απευθυνόμενο στην HPC κοινότητα, οι κατασκευαστές τείνουν να επενδύουν σε υπάρχουσες τεχνολογίες, που έχουν ήδη υλοποιηθεί και δοκιμαστεί σε λογισμικό.

Απλό: Ο σχεδιασμός και ο κώδικας του συστήματος πρέπει να είναι απλός, κατανοητός και να ταιριάζει στις απαιτήσεις του virtualization. Η προγραμματιστική διεπαφή πρέπει να παραμένει απλή και να προσφέρει δυνατότητες για επαναχρησιμοποίηση του κώδικα, είτε από το λειτουργικό σύστημα, είτε από τη βάση του ανώτερου πρωτοκόλλου.

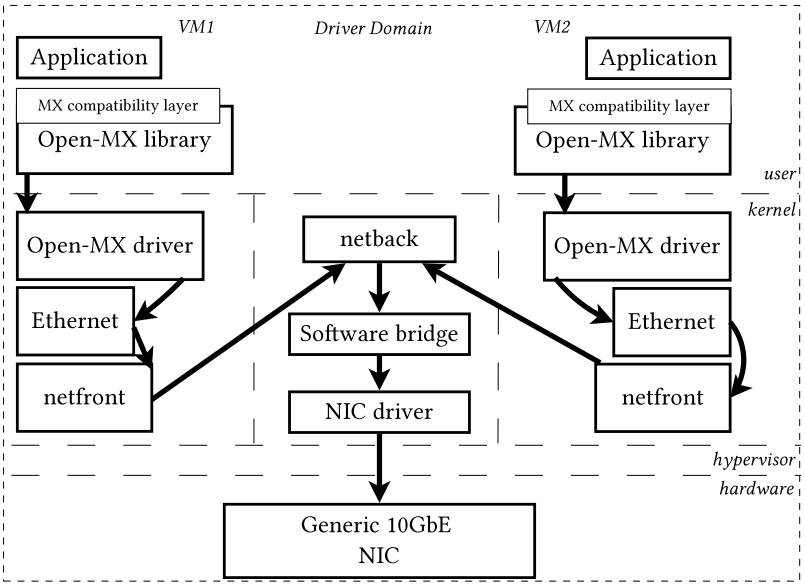
Κλιμακώσιμο: Είναι κοινός τόπος στην κοινότητα του HPC να αντιστοιχίζονται οι χώροι μνήμης που σχετίζονται με την αποστολή και λήψη μηνυμάτων με οποιαδήποτε μορφή επικοινωνιακής διεπαφής (communication port). Αυτό έχει σχολιαστεί έντονα [2], λόγω της έλλειψης κλιμάκωσης που υπονοεί Όσο αυξάνονται οι κόμβοι του δικτύου, η ενδελεχής χαρτογράφησή του επιβαρύνει σημαντικά την επικοινωνία τόσο με όρους χώρου αποθήκευσης στη μνήμη, όσο και επεξεργασίας των δεδομένων. Αντιθέτως, οι τεχνικές τακτικής σάρωσης (polling) για εισερχόμενες συνδέσεις αποτελούν έναν καλό συμβιβασμό, με δεδομένο ότι το μόνο επακόλουθό τους είναι μερικοί ”χαμένοι” κύκλοι επεξεργασίας.

Επιπρόσθετα, η *σύγχρονη* επικοινωνία προσφέρει το μικρότερο δυνατό χρόνο απόκρισης · αντίθετα, η *ασύγχρονη* επικοινωνία μπορεί να βοηθήσει στην επικάλυψη χρήσης των πόρων του συστήματος, και οδηγεί σε βελτιστοποιημένη επικοινωνία και καλύτερη κλιμάκωση.

Στιβαρό: Ένας από τους αρχικούς στόχους του MPI ήταν η απλότητα. Αυτός είναι ένας από τους σημαντικότερους λόγους που ανοχή σφαλμάτων στο MPI δεν είναι εκτεταμένη. Σαν αποτέλεσμα, τα κατώτερα στρώματα πρωτοκόλλων δικτύωσης πρέπει να διασφαλίζουν πως οι καταστάσεις σφαλμάτων αντιμετωπίζονται επιτυχώς. Η σημασιολογία συνδέσεων (connection oriented semantics) μπορεί να λύσει αυτό το πρόβλημα, κρατώντας ταυτόχρονα τη σημασιολογία του πρωτοκόλλου άρρηκτα συνδεδεμένη με το MPI.

4.1.2 Ανταλλαγή μηνυμάτων στο Cloud

Στην περίπτωση του cloud computing, και συγκεκριμένα όσον αφορά στο virtualization, η επικοινωνία υλοποιείται με μια περίπλοκη διαδικασία. Το εικονικό περιβάλλον είναι εχθρικό



Σχήμα 4.1: Paravirtualized συσκευές: Το Open-MX σε ένα κοινό ελεγκτή εικονικών μηχανών, με χρήση του μοντέλου διαχωρισμένου οδηγού, μια γέφυρα Ethernet και έναν κοινό προσαρμογέα δικτύου.

για εφαρμογές απαιτητικές σε επικοινωνία· κυρίως εφαρμογές που παρουσιάζουν ευαισθησία σε χαμηλούς χρόνους απόκρισης. Τα επιπλέον επίπεδα αφαίρεσης που δημιουργούν τα στρώματα του virtualization μειώνουν τη συνολική επίδοση σημαντικά.

Οι τρεις βασικές μέθοδοι για επικοινωνία σε εικονικά περιβάλλοντα είναι οι εξής:

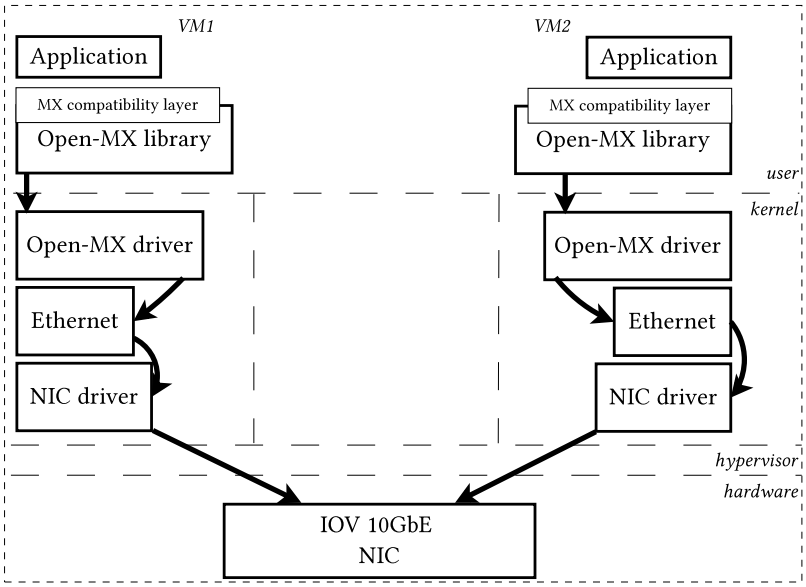
(a) εξομοίωση συσκευών: ο ελεγκτής εικονικών μηχανών προσφέρει μια διεπαφή προς την εικονική μηχανή εξομοιώνοντας τις λειτουργίες της συσκευής.

(b) paravirtualized συσκευές: μια βελτιστοποιημένη έκδοση της εξομοίωσης συσκευών που επιτυγχάνει την καλύτερη απόδοση με κοινού τύπου υλικό.

(c) I/O Virtualization (IOV): η μέθοδος αυτή προϋποθέτει την ύπαρξη εξειδικευμένου υλικού που μπορεί να προσφέρει πολλαπλά PCI functions στον ελεγκτή που με τη σειρά του, τα αντιστοιχίζει απευθείας σε εικονικές μηχανές.

Στις περιπτώσεις που μελετάει η παρούσα διατριβή (εφαρμογές υψηλών απαιτήσεων), οι ρεαλιστικές μέθοδοι με όρους επίδοσης είναι η (b) και η (c). Η εξομοίωση συσκευών επιβάλλει σημαντική συμφόρηση στο μονοπάτι της επικοινωνίας.

Τα Σχήματα 4.1 και 4.2 παρουσιάζουν το μονοπάτι των δεδομένων για τη λειτουργία της αποστολής ενός μηνύματος χρησιμοποιώντας τις μεθόδους (b) και (c) αντίστοιχα.



Σχήμα 4.2: IOV: Το Open-MX πάνω από προσαρμογείς IOV. Κάθε PCI function αντιστοιχίζεται στην εκάστοτε εικονική μηχανή, εγκαθιστώντας ένα απευθείας μονοπάτι χωρίς τη μεσολάβηση του ελεγκτή εικονικών μηχανών.

Παρόλο που η μέθοδος (c) (IOV) επικρατεί σε σχέση με τις υπόλοιπες όσον αφορά στην επίδοση, πολυπλοκοποιεί σημαντικά την εγκατάσταση: αρχικά, προκύπτουν όρια του υλικού όσο κλιμακώνει η εγκατάσταση – οι εικονικές μηχανές μπορούν να έχουν πρόσβαση στο υλικό και να μοιράζονται τις δυνατότητές του ανάλογα με τον αριθμό των PCI functions που υποστηρίζονται. Στη συνέχεια, πολλά από τα πλεονεκτήματα του virtualization χάνονται (μεταφορά εικονικών μηχανών, καταστολή/ανάνηψη, QoS, κλπ.). Επιπρόσθετα, ο ελεγκτής δεν έχει πρόσβαση στη διαχείριση της πρόσβασης στο δίκτυο, αφού η πολύπλεξη των αιτήσεων πρόσβασης στο δίκτυο γίνεται αποκλειστικά στο υλικό. Όλα τα παραπάνω υπονοούν σημαντική απώλεια ευελιξίας. Σαν αποτέλεσμα, στις περισσότερες περιπτώσεις χρησιμοποιείται η μέθοδος (b) που βασίζεται στο paravirtualization.

Λαμβάνοντας υπόψη τα παραπάνω επιχειρήματα, επιλέγουμε να βασίσουμε το σύστημα που υλοποιούμε στο Open-MX και στο paravirtualization. Σχεδιάζουμε το Xen2MX, μια δικτυακή στοίβα που επιτρέπει:

- στο χώρο χρήστη των εικονικών μηχανών να χρησιμοποιεί την προγραμματιστική διεπαφή του Open-MX καθώς και όλα τα χαρακτηριστικά του, συμπεριλαμβανομένης και της πλήρους συμβατότητας με το MX.
- στους παρόχους υπηρεσιών Cloud, να εκμεταλλεύονται την ευελιξία που προσφέρει

η μέθοδος του paravirtualization συγκριτικά με τις τεχνικές IOV που απαιτούν εξειδικευμένο υλικό και αυξάνουν το βαθμό ετερογένειας στο κέντρο δεδομένων της υποδομής.

5

Σχεδίαση και υλοποίηση συστήματος δικτύωσης υψηλής επίδοσης για εικονικές μηχανές

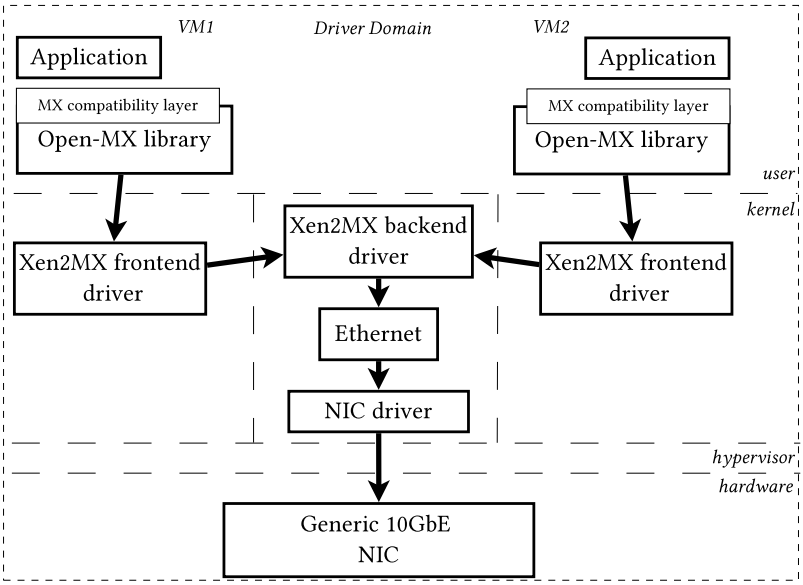
5.1 Σχεδίαση του Xen2MX

Εισαγωγή

Με το σύστημα Xen2MX προτείνουμε την εισαγωγή σημασιολογίας δικτύων υψηλής επίδοσης σε περιβάλλοντα εικονικών μηχανών. Οι βάσεις του συστήματος χτίζονται πάνω σε: endpoints, μηχανισμούς ασύγχρονης επικοινωνίας και διακοπών καθώς και έξυπνων αντιστοιχίσεων μνήμης. Με αυτό τον τρόπο, ο χώρος χρήστη των εικονικών μηχανών μπορεί να επικοινωνεί με το δίκτυο χρησιμοποιώντας το πρωτόκολλο MX χωρίς να υφίσταται τις επιβαρύνσεις του κοινού συστήματος δικτύωσης Ethernet (software bridge), ενώ ταυτόχρονα η εικονική μηχανή ελέγχου έχει πλήρη έλεγχο στην κίνηση του δικτύου των εικονικών μηχανών. Ουσιαστικά, αυτή είναι η βασική ιδέα της σχεδίασης: εφαρμογές που εκτελούνται στο χώρο χρήστη των εικονικών μηχανών αλληλεπιδρούν με τη βιβλιοθήκη του MX· η βιβλιοθήκη εκτελεί κλήσεις προς το frontend, κρατώντας τη σημασιολογία του πρωτοκόλλου ανέπαφη· το frontend προωθεί τις αιτήσεις προς το backend και αντίστροφα (δέχεται απαντήσεις από αυτό), ανάλογα με τη φορά της ροής δεδομένων· τέλος, τα Ethernet frames κατακερματίζονται και διανέμονται στις σελίδες του frontend (λήψη) ή χτίζονται βάσει του μεγέθους του μηνύματος και μεταδίδονται στο δίκτυο μέσω της στοίβας Ethernet του πυρήνα του Linux.

Για να διατηρήσουμε τη συμβατότητα της βιβλιοθήκης με το πρωτόκολλο Myrinet/MX, πρέπει να ακολουθήσουμε αυστηρά τη σημασιολογία του πρωτοκόλλου. Τα μηνύματα κατηγοριοποιούνται βάσει του μεγέθους τους ως εξής: SMALL (μέχρι 128 bytes), MEDIUM (από 129 bytes μέχρι 32 KB) και LARGE (περισσότερο από 32 KB).

Η επικοινωνία μεταξύ του frontend και του backend βασίζεται στους μηχανισμούς του



Σχήμα 5.1: Η αρχιτεκτονική του Xen2MX: ο σχεδιασμός του Xen2MX ακολουθεί το μοντέλο διαχωρισμένου οδηγού (split driver model). Το Xen2MX διαθέτει το frontend και το backend, δύο οδηγούς συσκευών για την εικονική μηχανή και το λειτουργικό σύστημα διαχείρισης αντίστοιχα. Το ανώτερο στρώμα του μοντέλου είναι πλήρως συμβατό με το Open-MX ενώ η διασύνδεση με το δίκτυο υλοποιείται με χρήση κοινών κλήσεων στον πυρήνα του Linux και την αντίστοιχη διεπαφή της δικτυακής στοίβας του Ethernet.

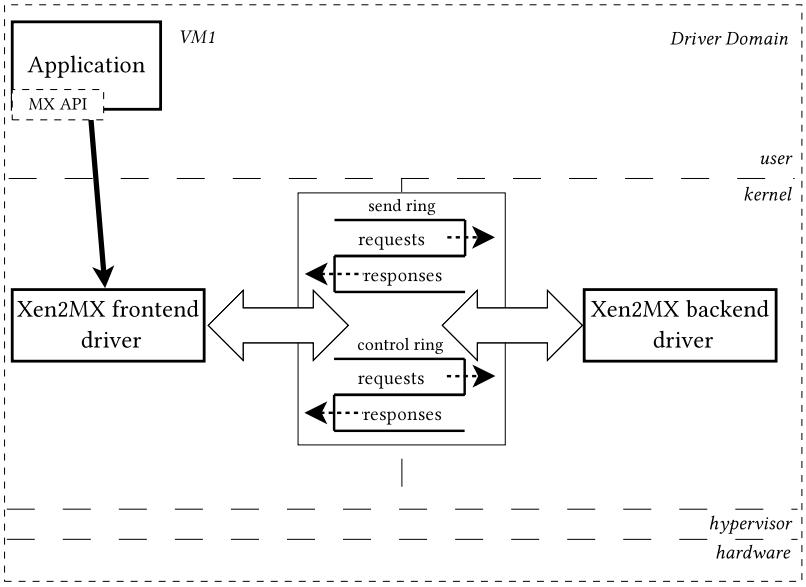
ελεγκτή εικονικών μηχανών για ανταλλαγή μηνυμάτων και παραχώρησης μνήμης. Σχεδόν όλη η επικοινωνία από το χώρο χρήστη στο χώρο πυρήνα προωθείται στο backend, που αναλαμβάνει την αντιστοίχηση των σχετικών περιοχών μνήμης του frontend και το μεγαλύτερο μέρος της επεξεργασίας του πρωτοκόλλου.

Η καταχώρηση μνήμης είναι σύγχρονη, δίνοντας τη δυνατότητα στο backend να προσπελάσει το χώρο μνήμης που έχει δεσμεύσει ο χώρος χρήστη της εικονικής μηχανής. Οι σελίδες μνήμης που απαρτίζουν αυτό το χώρο μνήμης μπορούν να αγκιστρωθούν σε socket buffers σαν fragments, δημιουργώντας ένα πακέτο έτοιμο να μεταδοθεί στο δίκτυο. Ταυτόχρονα, αυτές οι σελίδες, μπορούν να είναι και ο προορισμός μιας αίτησης για λήψη δεδομένων και να γεμίσουν με δεδομένα από το δίκτυο.

Τα endpoints περιέχουν ουρές αποστολής και λήψης; πρόκειται για στατικούς buffers που δρουν ως η πηγή ή ο προορισμός μηνυμάτων τύπου MEDIUM. Οι ουρές αυτές, μαζί με τους buffers, αντιστοιχίζονται στο backend, παραχωρούνται με χρήση των μηχανισμών του Xen και διευθυνσιοδοτούνται από τα κατώτερα στρώματα του πρωτοκόλλου.

5.1.1 Επικοινωνία ενδιάμεσων οδηγών εικονικών διεπαφών

Ο βασικός πυλώνας της σχεδίασης του συστήματος είναι η διεπαφή μεταξύ της εικονικής μηχανής και της εικονικής μηχανής ελέγχου. Ο VMM προσφέρει βασικές μεθόδους επικοινωνίας για σχήματα καταναλωτών–παραγωγών (Ενότητα 2.4.3), με βάση τις οποίες υλοποιούμε ένα μηχανισμό ειδοποίησης χρησιμοποιώντας διακοπές λογισμικού (soft interrupts) και τεχνικές τακτικής σάρωσης (polling), ανάλογα με τις εκάστοτε απαιτήσεις· για παράδειγμα, τα μηνύματα επιβεβαίωσης (acknowledgment notifications) ενεργοποιούν έναν "οκνηρό" χειριστή (lazy handler) ενώ οι λήψεις μηνυμάτων από το δίκτυο (όταν το σύστημα περιμένει για λήψεις) παραδίδονται σε πραγματικό χρόνο.



Σχήμα 5.2: Οι δακτύλιοι επικοινωνίας του Xen2MX: η αλληλεπίδραση μεταξύ εικονικής μηχανής και ελεγκτή υλοποιείται με χρήση των I/O rings (ελέγχου και αποστολής). Οι αιτήσεις προέρχονται από το frontend ενώ οι απαντήσεις παράγονται στο backend. Τα I/O rings υλοποιούνται χρησιμοποιώντας τους μηχανισμούς του Xen για παραχώρηση σελίδων και τα κανάλια γεγονότων.

Στο Σχήμα 5.2 παρουσιάζεται η βασική σημασιολογία του Xen2MX. Υλοποιούμε απλούς κυκλικούς buffers E/E (I/O rings) με βάσει τους μηχανισμούς παραχώρησης μνήμης και τα κανάλια γεγονότων. Ένα I/O ring αποτελείται από μια κοινή σελίδα μνήμης που παραχωρείται από το frontend και ένα κανάλι γεγονότων που προσδίδει λειτουργικότητα παρόμοια με αυτή μιας εικονικής ουράς διακοπών. Το κάθε I/O ring μπορεί να μεταφέρει αιτήσεις (VM–to–host) και απαντήσεις (host–to–VM). Για την αποστολή ενός μηνύματος, η εικονική μη-

χανή χτίζει μια αίτηση και εκτελεί μια εντολή PUSH_REQUESTS, ειδοποιώντας το backend. Με αυτό τον τρόπο, η εικονική μηχανή ενημερώνει το backend ότι υπάρχουν αιτήσεις που εκκρεμούν. Στο μέρος του Host, καλείται ο σχετικός χειριστής και αναλαμβάνει να επεξεργαστεί την αίτηση. Μετά το πέρας της επεξεργασίας, το backend χτίζει την απάντηση και καλεί την PUSH_RESPONSES. Ανάλογα με την κατεύθυνση της ροής δεδομένων και τον τύπο της αίτησης, ο Host ενδέχεται να ειδοποιήσει την εικονική μηχανή με διακοπή.

Η ανταλλαγή πληροφοριών ελέγχου (control data exchange) υλοποιείται χρησιμοποιώντας δύο κανάλια: το send_ring αναλαμβάνει εντολές ελέγχου και το μεγαλύτερο μέρος του μονοπατιού αποστολής. Το control_ring αναλαμβάνει κυρίως ειδοποιήσεις και επιβεβαιώσεις.

Τα μηνύματα τύπου SMALL αντιγράφονται μέσα από αυτά τα I/O rings. Αυτή η δυνατότητα, που συνδυάζεται με τεχνικές τακτικής σάρωσης και στα δύο άκρα, δίνει το χαμηλότερο δυνατό χρόνο απόκρισης σε αυτό το σύστημα. Η ανταλλαγή μηνυμάτων τύπου MEDIUM και LARGE υλοποιείται με ένα πιο πολύπλοκο σενάριο. Όπως και στην περίπτωση του Open-MX, τα MEDIUM μηνύματα χρησιμοποιούν τις ουρές αποστολής και λήψης· ο χώρος που καταλαμβάνουν οι ουρές είναι στατικός, δεσμεύεται με τη δημιουργία του σχετικού endpoint και διευθυνσιοδοτείται χρησιμοποιώντας εσωτερικούς δείκτες. Τα LARGE μηνύματα χρησιμοποιούν τις καταχωρημένες περιοχές μνήμης (memory regions). Στη συνέχεια περιγράφουμε το χειρισμό αυτών των μηνυμάτων με μεγαλύτερη λεπτομέρεια.

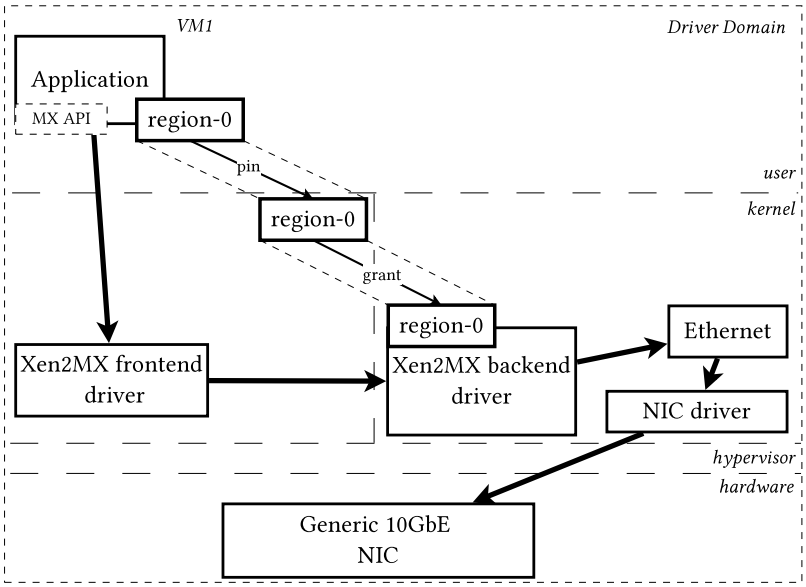
5.1.2 Ουρές αποστολής και λήψης

Το Open-MX δεσμεύει χώρο μνήμης εκ των προτέρων για τα MEDIUM μηνύματα. Αυτός ο χώρος δεσμεύεται με τη δημιουργία του endpoint και δεικτοδοτείται χρησιμοποιώντας εσωτερικά πεδία του endpoint. Παρόλα αυτά, στην περίπτωση του Xen2MX αυτός ο χώρος πρέπει να είναι προσπελάσιμος τόσο από το frontend όσο και απο το backend. Για να καταστεί αυτό δυνατό, το frontend παραχωρεί τις σελίδες μνήμης που αποτελούν αυτές τις ουρές στο backend με αποτέλεσμα να προκύπτουν πανομοιότυπες δομές και στα δύο άκρα. Για να διατηρείται συνοχή στο δεικτοδότηση, το frontend παραχωρεί και τους δείκτες που ελέγχουν το φόρτο των ουρών.

Σαν αποτέλεσμα, το backend μπορεί να τοποθετήσει δεδομένα που προέρχονται από το δίκτυο στην ουρά λήψης και να ειδοποιήσει το frontend· αντίστροφα, το backend μπορεί να στείλει δεδομένα στο δίκτυο από την ουρά αποστολής αμέσως μόλις το frontend σημάνει μια αίτηση αποστολής.

5.1.3 Περιοχές μνήμης

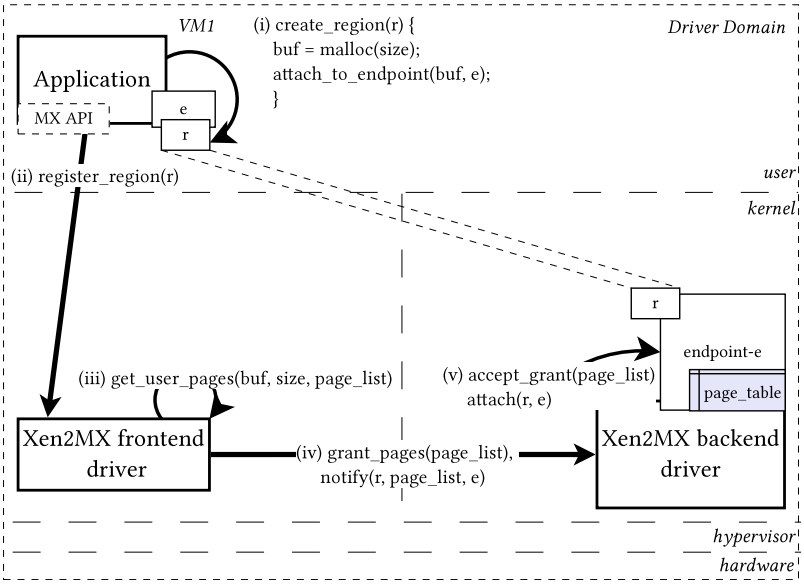
Οι περιοχές μνήμης στο Open-MX δεσμεύονται στο χώρο χρήστη της εικονικής μηχανής και καταχωρούνται χρησιμοποιώντας μια συγκεκριμένη εντολή IOCTL. Στο Xen2MX, το frontend παραχωρεί στο backend όλα τα κομμάτια (segments) του χώρου της περιοχής μνήμης. Στο frontend, κάθε κομμάτι περιέχει επιπλέον πεδία που κρατούν στοιχεία για τις σελίδες που έχουν παραχωρηθεί. Οι σελίδες αυτές απελευθερώνονται όταν το backend δεν τις χρησιμοποιεί πλέον. Ο τρόπος με τον οποίο το Xen2MX χειρίζεται τις περιοχές μνήμης φαίνεται στο Σχήμα 5.3.



Σχήμα 5.3: Οι περιοχές μνήμης δεσμεύονται στο χώρο χρήστη της εικονικής μηχανής, εντοπίζονται οι σελίδες που αντιστοιχούνται σ' αυτές τις περιοχές στο χώρο πυρήνα της εικονικής μηχανής και παραχωρούνται στο backend από το frontend. Το backend μπορεί να επισυνάψει τις σελίδες αυτές σε socket buffers ή να τις χρησιμοποιήσει ως προορισμό για λήψεις πακέτων από το δίκτυο, χωρίς να χρειάζεται επιπλέον αντιγραφή των δεδομένων που οδηγεί σε σημαντική επιβάρυνση στην επίδοση.

Η διαδικασία παραχώρησης των περιοχών μνήμης από το frontend στο backend αποτελείται από δύο βήματα: στην πλευρά της εικονικής μηχανής, το frontend χρησιμοποιεί τις συναρτήσεις του Open-MX για να δημιουργήσει μια περιοχή σύμφωνα με το πρωτόκολλο· στη συνέχεια, παραχωρεί στο backend κάθε μία σελίδα που ανήκει στα κομμάτια της περιοχής, διατηρώντας σε επιπλέον χώρο τις αναφορές παραχώρησης (grant references). Για να μπορεί το backend να αποκτήσει πρόσβαση σε αυτές τις σελίδες που το frontend παραχωρεί,

θα πρέπει να έχει πρόσβαση στις αναφορές παραχώρησης. Αντί το frontend να τις στέλνει μία αναφορά παραχώρησης ανά μήνυμα, τις στοιβάζει σε ένα σύνολο από info_pages τις οποίες παραχωρεί με ένα μήνυμα. Το backend δέχεται την παραχώρηση και χρησιμοποιεί το περιεχόμενο των σελίδων αυτών για να αποδεχτεί τις σελίδες που απαρτίζουν την περιοχή μνήμης. Με αυτό τον τρόπο, το backend διαθέτει πρόσβαση στις σελίδες της περιοχής μνήμης που δέσμευσε ο χώρος χρήστη του frontend.



Σχήμα 5.4: Βασικά βήματα της παραχώρησης μιας περιοχής μνήμης: (i) η εφαρμογή δημιουργεί μια περιοχή μνήμης, την επισυνάπτει στο σχετικό endpoint και (ii) την καταχωρεί· (iii) το frontend βρίσκει τις σελίδες που την απαρτίζουν και (iv) τις παραχωρεί στο backend· (v) το backend αποδέχεται τις σελίδες και ανανεώνει τις σχετικές δομές. Τα δεδομένα της εφαρμογής είναι διαθέσιμα στο backend.

Για την περαιτέρω κατανόηση της παραπάνω διαδικασίας, ας δούμε ένα παράδειγμα: για να παραχωρηθεί ένα κομμάτι 8 MB χρειάζονται $\frac{8MB}{PAGE\_SIZE}$ αναφορές παραχώρησης. Για μέγεθος σελίδας 4 KB δηλαδή, χρειάζονται 2048 αναφορές, που σημαίνει: $\frac{2048*4bytes}{4096} = 2$ info_pages.

1. Το frontend δεσμεύει τα info_pages και σαρώνει επαναληπτικά το κομμάτι των 8 MB για να παραχωρήσει κάθε μία από τις 2048 σελίδες στο backend, ενώ ταυτόχρονα, γεμίζει τα info_pages με τις σχετικές αναφορές παραχώρησης (grefs). Τα info_pages περιέχουν τις αναφορές παραχώρησης που πρέπει να μεταφερθούν στο backend.
2. Το frontend παραχωρεί τα info_pages στο backend και το ειδοποιεί για την αίτηση που

εκκρεμεί

3. Το backend αποδέχεται τα info_pages για τα grefs του κομματιού των 8 MB.
4. Αφού λάβει τις αναφορές παραχώρησης, αποδέχεται τις σελίδες μία προς μία και έτσι, το κομμάτι των 8 MB είναι διαθέσιμο και στα δύο άκρα του διαχωρισμένου οδηγού.

Αυτή η προσέγγιση είναι διαφορετική από αυτή που χρησιμοποιείται στο Xen τόσο για την κλάση συσκευών δικτύου, όσο και για την αντίστοιχη των συσκευών αποθήκευσης. Αυτό συμβαίνει λόγω του μεγάλου αριθμού αναφορών παραχώρησης που χρειάζεται να μεταφερθούν στην περίπτωση του Xen2MX. Ουσιαστικά πρόκειται για διαφορά στη σημασιολογία των δύο μεθόδων: οι οδηγοί netfront/netback χειρίζονται δεδομένα με μέγιστο μέγεθος μηνύματος όσο και το MTU (αφού κάθε λειτουργία αναφέρεται σε Ethernet frames). Αντίθετα, το Xen2MX χειρίζεται δεδομένα στο πλαίσιο ανταλλαγής μηνυμάτων ανεξαρτήτου μεγέθους, αφού το σχεδιάζουμε ως το υπόστρωμα για ένα δίκτυο διασύνδεσης υψηλής επίδοσης.

5.1.4 Ανταλλαγή μηνυμάτων

Ένα εικονικό περιβάλλον δεν προσφέρει την ευελιξία των λειτουργικών συστημάτων για τη βελτίωση της επικοινωνίας με το δίκτυο. Για παράδειγμα, οι τεχνικές *zero-copy* υλοποιούνται με εντελώς διαφορετικό τρόπο σε εικονικές μηχανές συγκριτικά με τις αντίστοιχες σε φυσικά μηχανήματα. Το επιπλέον βήμα στη διαδικασία εκτέλεσης αναφέρεται στη μεταφορά (ή και την παραχώρηση) δεδομένων μεταξύ εικονικών μηχανών και του ελεγκτή. Για να μπορέσουμε να διατηρήσουμε τη συμβατότητα με το MX και να απαλείψουμε ενδιάμεσες επιβαρύνσεις πρέπει να σχεδιάσουμε τη στοίβα του πρωτοκόλλου προσεκτικά.

Για κάθε τύπο μηνύματος, περιγράφουμε παρακάτω τον τρόπο που τα δεδομένα διατρέχουν τη στοίβα:

- SMALL: Τα μηνύματα αντιγράφονται από το χώρο χρήστη. Το μέγεθός τους είναι μικρότερο από 128 bytes, επομένως η παραχώρηση μιας σελίδας μεγέθους 4 KB θα δημιουργούσε σημαντική επιβάρυνση, αφού, εκτός από την καθυστέρηση της ειδοποίησης (από το frontend στο backend), περιλαμβάνει τουλάχιστον 2 hypercalls. Τα δεδομένα ενσωματώνονται στη δομή της αίτησης μέσω του control_ring · για να αποφευχθεί η επιπλέον αντιγραφή, τα δεδομένα του χώρου χρήστη αντιγράφονται απευθείας στο buffer της αίτησης. Επομένως, ο αριθμός των αντιγραφών παραμένει ίδιος με τον αντίστοιχο του Open-MX και η μόνη επιβάρυνση που υφίσταται η διαδικασία αποστολής είναι η σταθερή καθυστέρηση του μηνύματος ενημέρωσης προς το backend.

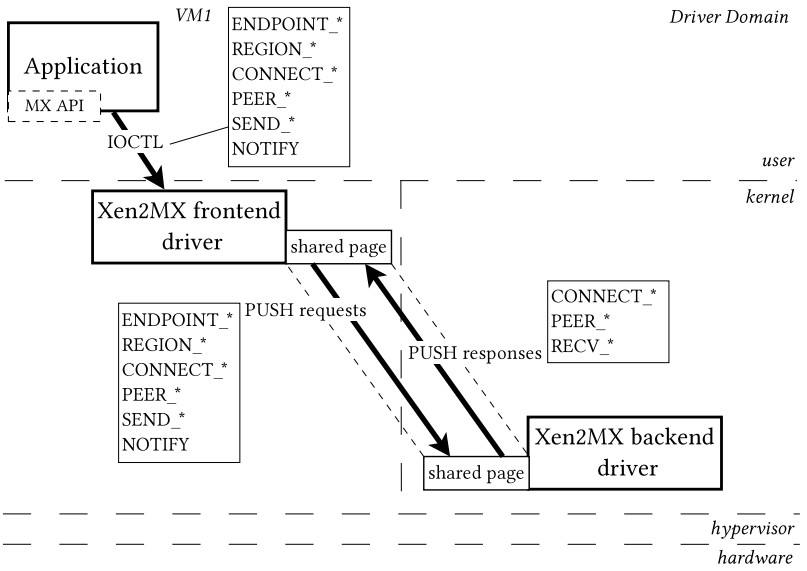
- MEDIUM: Τα μηνύματα μεταφέρονται από τις αντίστοιχες ουρές αποστολής/λήψης. Ο μηχανισμός που χρησιμοποιείται για αυτά τα μηνύματα είναι ο ίδιος με τα SMALL. Η επιπλέον επιβάρυνση αναφέρεται στη δέσμευση των σελίδων και στην παραχώρησή τους που συνδέεται με το άνοιγμα του αντίστοιχου endpoint (το οποίο δε συμβαίνει στο κρίσιμο μονοπάτι). Τα δεδομένα τοποθετούνται στην ουρά αποστολής από το χώρο χρήστη, περνώντας μόνο το δείκτη και το μέγεθος του μηνύματος μέσω μιας κλήσης IOCTL. Στη συνέχεια, η αίτηση προωθείται στο backend, οπότε τα δεδομένα ακολουθούν το κοινό μονοπάτι, με την επιπλέον επιβάρυνση της ειδοποίησης του frontend προς το backend.
- LARGE Τα μηνύματα αυτά μεταφέρονται από περιοχές μνήμης. Προκειμένου να χρησιμοποιηθεί μια περιοχή μνήμης πρέπει να καταχωρηθεί (register). Αυτό είναι ένα περίπλοκο βήμα της διαδικασίας: η καταχώρηση μιας περιοχής μνήμης είναι χρονοβόρα διαδικασία, λόγω του αριθμού των σελίδων που πρέπει να παραχωρηθούν στο backend από το frontend. Επιπλέον, για να εδραιωθεί η αντιστοίχηση, το backend πρέπει να αποδεχτεί την παραχώρηση.

5.1.5 Επικοινωνία χώρου–χρήστη με το δίκτυο

Όπως αναφέρθηκε παραπάνω, η σχεδίαση του Xen2MX έχει σαν στόχο να μειώσει στο ελάχιστο την επιβάρυνση που υφίστανται οι εφαρμογές από τα επιμέρους στρώματα του virtualization με εκτενή μελέτη όλων των μερών του συστήματος που λαμβάνουν μέρος στην επικοινωνία: user–to–kernel, guest–to–host και host–to-network. Το Σχήμα 5.5 περιγράφει τις δυνατότητες του frontend σε σχέση με την επικοινωνία με την εφαρμογή (στο χώρο χρήστη του VM) καθώς και με το backend (στο Host). Αυτά τα βήματα συμπληρώνουν το μονοπάτι της επικοινωνίας από το χώρο χρήστη στο δίκτυο.

Συγκεκριμένα:

- *user–to–frontend*: Η εφαρμογή εκτελεί απλές κλήσεις Open-MX μέσω IOCTLs. Το frontend καλεί τον ανάλογο χειριστή μέσω ενός state machine, ο οποίος αναλαμβάνει να επεξεργαστεί τις αιτήσεις. Για παράδειγμα, αν η αίτηση αφορά το άνοιγμα ενός endpoint (ENDPOINT_OPEN), το frontend δεσμεύει τις σχετικές δομές και παραχωρεί τις σχετικές σελίδες μνήμης στο backend.
- *guest–to–host*: Το frontend τοποθετεί ένα request στο αντίστοιχο ring (που ουσιαστικά είναι μια μοιραζόμενη σελίδα) και ειδοποιεί το backend (PUSH_REQUESTS) ότι μια αίτηση εκκρεμεί μέσω του καναλιού γεγονότων του Xen. Το backend ειδο-



Σχήμα 5.5: Χαρακτηριστικά του οδηγού που διαμορφώνουν το μονοπάτι χώρος-χρήστη–δίκτυο.

ποιείται και ενεργοποιεί το χειριστή για την επεξεργασία της αίτησης. Όταν ολοκληρωθεί, το backend ενημερώνει το frontend για την περάτωση της λειτουργίας. Στο παραπάνω παράδειγμα, το backend ανοίγει ένα endpoint, το συνδέει με τη σχετική δικτυακή διεπαφή, αποδέχεται τις σελίδες μνήμης που έχουν παραχωρηθεί και χτίζει μια επιβεβαίωση την οποία προωθεί στο frontend (PUSH_RESPONSES).

- *host–to–network*: Το backend εκτελεί κλήσεις Open-MX για να στείλει δεδομένα στο δίκτυο χρησιμοποιώντας την κλασική ροή δεδομένων: την επεξεργασία των πακέτων και την αποστολή τους στο επίπεδο Ethernet του πυρήνα του Linux.

5.1.6 Χαρτογράφηση δικτύου

Τα σύγχρονα πρωτόκολλα ανταλλαγής μηνυμάτων, πριν από την επικοινωνία, χαρτογραφούν το σύνολο των δικτυακών κόμβων για να καταγράψουν το γράφο σύνδεσης του δικτύου. Στο Open-MX, όταν ένας κόμβος εισέρχεται στο δίκτυο διαφημίζει την ύπαρξή του χρησιμοποιώντας μια απλοποιημένη έκδοση των endpoints, τα *raw endpoints*. Ο κόμβος μεταδίδει το hostname του, μαζί με ένα αναγνωριστικό της φυσικής διεπαφής δικτύου με την οποία συνδέεται. Αυτή η πληροφορία αποθηκεύεται στον κάθε κόμβο σε μια δομή που καλείται peer_table, μέσω της οποίας οποιαδήποτε εφαρμογή θελήσει να επικοινωνήσει με το δίκτυο ανακτά το μοναδικό peer_index. Αυτή η πληροφορία είναι βασική για τα πρωτόκολλα

ανώτερου επιπέδου (όπως το MPI), που χρησιμοποιούν hostnames για να εγκαταστήσουν την αρχική επικοινωνία (συνήθως μέσω TCP/IP) με τους υπόλοιπους κόμβους.

Στη σχεδίαση του Xen2MX, η κατάρτιση του χάρτη του δικτύου είναι μια πιο περίπλοκη διαδικασία: οι εικονικές μηχανές που συνυπάρχουν στο ίδιο φυσικό μηχάνημα, μοιράζονται την ίδια φυσική διεπαφή δικτύου. Έτσι, αν η χαρτογράφηση γίνει με τον παραπάνω τρόπο, θα είναι ανεπαρκής, αφού πολλαπλοί κόμβοι-VMs θα έχουν ένα κοινό αναγνωριστικό. Για να ξεπεραστεί αυτό, το peer_table επεκτάθηκε έτσι ώστε να υποστηρίζει πολλαπλούς κόμβους συνδεδεμένους στην ίδια διεπαφή Ethernet. Στην αρχικοποίηση, το backend χαρτογραφεί όλα τα VMs που είναι συνδεδεμένα στα Ethernet interfaces του κόμβου, ενώ ελέγχει τους υπόλοιπους κόμβους του δικτύου. Με αυτό τον τρόπο, τα απομακρυσμένα πακέτα Open-MX φτάνουν στο VM-προορισμό βάσει της διεύθυνσης MAC του φυσικού κόμβου, ενώ τα τοπικά πακέτα Open-MX (που έχουν ως πηγή VMs που συνυπάρχουν στον κόμβο) προωθούνται στο εκάστοτε VM.

5.1.7 Ενδοεπικοινωνία εικονικών μηχανών

Οι πάροχοι υπηρεσιών IaaS υιοθετούν μια προσέγγιση στην οποία ο τελικός χρήστης δε γνωρίζει σε ποιο φυσικό μηχάνημα τοποθετούνται οι εικονικές μηχανές που χρησιμοποιεί. Επιπρόσθετα, ο χρήστης δεν έχει απολύτως κανένα έλεγχο στην τοποθέτηση αυτή. Σαν αποτέλεσμα, το σύνολο των εικονικών μηχανών ενός χρήστη, μπορεί να καταλήξουν να συνυπάρχουν στο ίδιο φυσικό μηχάνημα, εκτελώντας εφαρμογές απαιτητικές σε Ε/Ε. Σε αυτή την περίπτωση, οι μέθοδοι IOV αποτυγχάνουν παταγωδώς: για να επικοινωνήσει ένα VM με ένα άλλο, τα δεδομένα θα πρέπει να κινηθούν από το χώρο χρήστη του ενός, στο υλικό και από εκεί στο χώρο χρήστη του δεύτερου VM· οι δύο περιοχές μνήμης όμως, βρίσκονται στο ίδιο φυσικό μέσο (τα στοιχεία μνήμης του φυσικού μηχανήματος) επομένως η όλη διαδικασία επιβάλει σημαντική επιβάρυνση χωρίς ιδιαίτερο λόγο.

Με χρήση τεχνικών paravirtualization (όπως το Xen2MX), αυτό το πρόβλημα εξαλείφεται, αφού ο hypervisor έχει πλήρη έλεγχο της επικοινωνίας. Ο σχεδιασμός του Xen2MX βασίζεται στο γεγονός ότι τα endpoints των εφαρμογών των VMs συνυπάρχουν στο backend. Με αυτό τον τρόπο, το backend μπορεί να αναζητήσει τον ανάλογο peer_index του endpoint-προορισμού και να ενεργοποιήσει το χειριστή του μηχανισμού μοιραζόμενης μνήμης. Τα δεδομένα καταλήγουν στο χώρο χρήστη του VM-προορισμού χωρίς καμία επιπλέον αντιγραφή, ή περιττά μονοπάτια.

Για τη συγκεκριμένη βελτιστοποίηση στην επικοινωνία, υπάρχουν δύο επιλογές για με-

ταφορά των δεδομένων:

Αντιγραφή (copy): τα δεδομένα αντιγράφονται στο VM-προορισμό, βάσει της τεχνικής ροής δεδομένων που χρησιμοποιείται στην αποστολή/λήψη.

Ανταλλαγή (flip): τα δεδομένα που προέρχονται από σελίδες του frontend, υπάρχουν με φυσικούς όρους σε σελίδες μηχανής (machine pages, που αναγνωρίζονται από το machine frame number, mfn). Αυτά τα mfns μπορούν να γεμίσουν σελίδες του VM-προορισμού και επομένως, τα δεδομένα φτάνουν στο χώρο χρήστη του VM χωρίς καμία αντιγραφή.

6

Πειραματική αποτίμηση του συστήματος Xen2MX

Στο κεφάλαιο αυτό, περιγράφουμε τα πειράματα που εκτελέσαμε για να αναλύσουμε τη συμπεριφορά του Xen2MX και να αναδείξουμε συγκεκριμένα χαρακτηριστικά της προσέγγισής μας, σε σύγκριση με συμβατικούς τρόπους επικοινωνίας που χρησιμοποιούνται σήμερα σε εικονικά περιβάλλοντα.

6.1 Επισκόπηση

Το υλικό που χρησιμοποιείται περιγράφεται στον Πίνακα 6.1. Η αποτίμηση του συστήματος γίνεται με δύο κατηγορίες πειραμάτων, με στόχο την ανάδειξη των πλεονεκτημάτων και μειονεκτημάτων του Xen2MX συγκριτικά με τις άλλες μεθόδους επικοινωνίας, καθώς και την παρουσίαση της συμπεριφοράς του σε πραγματικές συνθήκες λειτουργίας.

6.1.1 Κατηγορία Ι: Επικοινωνία εικονικών μηχανών με φυσικό μηχάνημα

Σε αυτή την κατηγορία, γίνεται λεπτομερής αποτίμηση του Xen2MX όσον αφορά στη ρυθμαπόδοση του συστήματος καθώς και στον χρόνο απόκρισης. Με αυτό τον τρόπο, καθίσταται δυνατή η ακριβής εκτίμηση της επιβάρυνσης που προσδίδει το Xen2MX συγκριτικά με την περίπτωση των φυσικών μηχανημάτων (όπου δεν υπάρχει κανένα επίπεδο virtualization).

Δημιουργούμε το περιβάλλον ενός VM container ($Host_0$) που μπορεί να περιέχει μέχρι 8 μονοπύρηνες εικονικές μηχανές ($VM_1 \ldots VM_8$). Με δεδομένο ότι οι διαθέσιμοι πόροι επεξεργασίας του συστήματος είναι 12 πυρήνες, αφήνουμε 4 για την εικονική μηχανή υπηρεσίας (driver domain). Στο απέναντι άκρο της επικοινωνίας δημιουργούμε ένα πανομοιότυπο περιβάλλον με τη διαφορά ότι δεν παρεμβάλλονται πουθενά επίπεδα virtualization. Επομένως πρόκειται για μια συμβατική εγκατάσταση Linux ($Host_1$).

Το μετροπρόγραμμα που χρησιμοποιείται είναι το mx_pingpong, που περιέχεται στη σουίτα μετροπρογραμμάτων του Myrinet/MX. Η λειτουργίες του αφορούν ανταλλαγές μηνυμάτων μεταβλητού μεγέθους μεταξύ ακροσημείων MX (MX endpoints). Η εκτέλεσή του γίνεται ανάμεσα στο Host_1 (φυσικό μηχάνημα, 8 διακριτές διεργασίες) και τις εικονικές μηχανές $\text{VM}_1 \ldots \text{VM}_8$ στο Host_0.

6.1.2 Κατηγορία II: Επικοινωνία εικονικών μηχανών που δε συνυπάρχουν στο ίδιο φυσικό μηχάνημα

Σε αυτή την κατηγορία, γίνεται λεπτομερής αποτίμηση του συστήματος σε συνθήκες πραγματικής λειτουργίας. Με αυτό τον τρόπο, καθίσταται δυνατή η αποτίμηση της επιβάρυνσης που ενδεχομένως να προστίθεται στην επικοινωνία εφαρμογών όταν αυτές εκτελούνται σε συστοιχίες εικονικών μηχανών.

Χρησιμοποιούμε την παραπάνω διάταξη (Ενότητα 6.3) με τη διαφορά ότι αντί για φυσικό μηχάνημα, εγκαθιστούμε περιβάλλον εικονικών μηχανών. Έτσι, η επικοινωνία γίνεται μεταξύ των εικονικών μηχανών $\text{VM}_1 \ldots \text{VM}_8$ στο Host_0 και των $\text{VM}_9 \ldots \text{VM}_{16}$ στο Host_1. Εκτελούμε το mx_pingpong σε αντικριστές εικονικές μηχανές, ως εξής: VM_1 προς VM_9, VM_2 προς VM_{10} κλπ. – μία διεργασία ανά εικονική μηχανή.

6.2 Λεπτομέρειες της πειραματικής διάταξης

Η προγραμματιστική διεπαφή, καθώς και η βιβλιοθήκη χώρου χρήστη του Xen2MX, είναι πανομοιότυπες με τις αντίστοιχες του Myrinet/MX· πρόκειται για επιλογή σχεδίασης έτσι, ώστε να διατηρηθεί ανέπαφη η σημασιολογία του πρωτοκόλλου. Σαν αποτέλεσμα, μπορούμε να χρησιμοποιούμε τα αυθεντικά μετροπρογράμματα του MX χωρίς μεταβολή-/προσαρμογή, και να συγκρίνουμε το Xen2MX με περιβάλλοντα χωρίς παρεμβολή επιπέδων virtualization καθώς και απλές/εναλλακτικές μεθόδους επικοινωνίας.

Χρησιμοποιούμε τρεις διαφορετικές διατάξεις για τα πειράματα:

- Bridged: Πρόκειται για την πιο συνηθισμένη μέθοδο επικοινωνίας σε εικονικά περιβάλλοντα. Στην περίπτωση του Xen, χρησιμοποιείται μία ”γέφυρα” Ethernet σε λογισμικό (Ethernet software bridge) για να μεταδίδει / λαμβάνει δεδομενογράμματα και τροποποιημένοι οδηγοί συσκευών (paravirtualized drivers) για την επικοινωνία των εικονικών μηχανών με το υλικό.

- PCI-attached: Σε αυτή την περίπτωση, το υλικό αντιστοιχίζεται απευθείας στην εικονική μηχανή, με χρήση της μεθόδου *pass-through* του Xen. Πρόκειται για μέθοδο που επιτυγχάνει τη μέγιστη επίδοση σε σύγκριση με περιβάλλοντα χωρίς επίπεδα virtualization, αφού οι αιτήσεις για πρόσβαση στο δίκτυο (αποστολή / λήψη δεδομένων) δεν πολυπλέκονται ούτε στον ελεγκτή εικονικών μηχανών, ούτε στο υλικό. Στις περισσότερες περιπτώσεις, αυτή η μέθοδος επιτυγχάνει επίδοση ανώτερη της μεθόδου IOV που περιγράφηκε στην Ενότητα 4.1.2.

- Xen2MX: Αυτή είναι μέθοδος που υλοποιήθηκε στην παρούσα διατριβή. Χρησιμοποιούμε δύο υπο-μεθόδους: τη Xen2MX-plain, όπου απενεργοποιούμε την επαναχρησιμοποίηση μνήμης που έχει δηλωθεί για επικοινωνία (memory registration caching disabled)· και τη Xen2MX-tuned, όπου ενεργοποιούμε την επιλογή αυτή (MX_RCACHE=1).

Μετά από μεθοδικές εκτελέσεις τόσο για τη μέθοδο Bridged όσο και για την PCI-attached διαπιστώθηκε ότι αυτή η παράμετρος (MX_RCACHE) δεν επηρεάζει τα αποτελέσματα (λιγότερο από 1% διαφορά)· επομένως σε ό,τι ακολουθεί, παρουσιάζουμε την καλύτερη περίπτωση. Για πληρότητα, σχεδιάζουμε και μια επιπλέον καμπύλη της περίπτωσης όπου δεν παρεμβάλλονται επίπεδα virtualization (Native). Με αυτό τον τρόπο, επισημαίνουμε επιβαρύνσεις που δεν οφείλονται στο σύστημα Xen2MX.

	VM containers/Native Linux boxes
Επεξεργαστές	2x Intel Xeon X5650@2.67GHz
Μητρική πλακέτα	Supermicro X8DTG-D
Chipset	Intel 5520
Κεντρική Μνήμη	12x4GB ECC@1333MHz
Υλικό δικτύου	Myrinet 10G-PCIE-8A-C

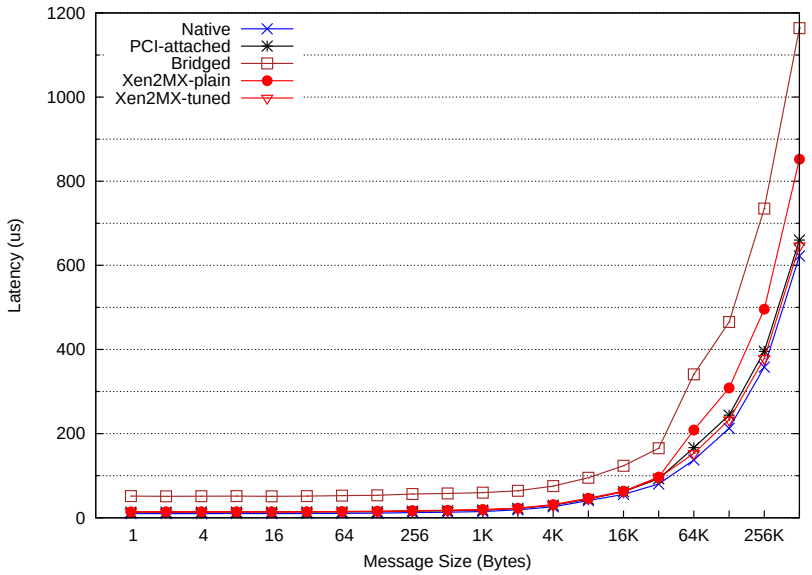
Πίνακας 6.1: Τεχνικές προδιαγραφές των μηχανημάτων στα οποία εκτελέστηκαν τα πειράματα

Τα πειράματα εκτελέστηκαν σε μηχανήματα με προδιαγραφές που φαίνονται στον Πίνακα 6.1, με χρήση του Xen 4.3-unstable[1] και πυρήνα Linux έκδοσης 3.7.1. Όλες οι περιπτώσεις, εκτός του Xen2MX χρησιμοποιούν Open-MX έκδοσης 1.5.2. Για όλα τα πειράματα χρησιμοποιήθηκε το mx_pingpong με MTU 9000 bytes. Για την αποτίμηση της χρήσης του CPU κατά τη διάρκεια εκτέλεσης των πειραμάτων χρησιμοποιήθηκε το /proc/stat.

[1] changeset 26059:c1c549c4fe9e

6.3 Πειραματική αποτίμηση του συστήματος Xen2MX – Κατηγορία I

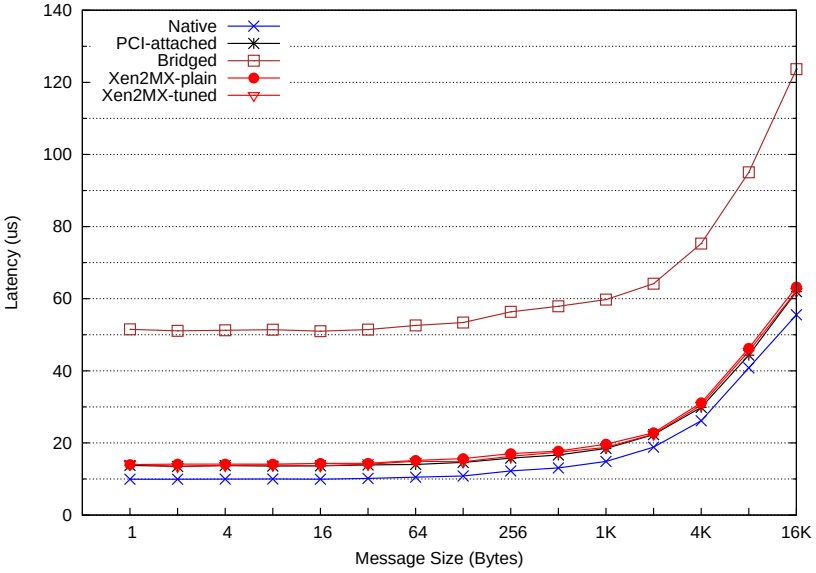
Για μια ακριβή εκτίμηση της επιβάρυνσης που προστίθεται στο μονοπάτι της επικοινωνίας, γίνεται, καταρχάς, αποτίμηση του χρόνου απόκρισης (round-trip latency, RTL) και της ρυθμαπόδοσης του συστήματος μεταξύ εικονικών μηχανών και ενός συμβατικού μηχανήματος, χωρίς τη μεσολάβηση τεχνικών virtualization. Χρησιμοποιείται το mx_pingpong για τη μεταφορά 10 GB για 10000 επαναλήψεις και τα αποτελέσματα παρουσιάζονται στα Σχήματα 6.1 και 6.3 αντίστοιχα.



Σχήμα 6.1: Χρόνος απόκρισης με χρήση του Xen2MX μεταξύ μονο-πύρηνης εικονικής μηχανής και κοινού συστήματος (χωρίς virtualization).

Στα Σχήματα 6.1 και 6.2 παρουσιάζεται ο χρόνος απόκρισης που μετρήθηκε με βάση τις προαναφερθείσες μεθόδους, συναρτήσει του μεγέθους του μηνύματος. Η μέθοδός μας (Xen2MX-plain) είναι σχεδόν αντίστοιχη με την PCI-attached για μικρά και μεσαία μηνύματα (SMALL και MEDIUM). Με χρήση του Xen2MX, η μεταφορά 1 byte στο απέναντι άκρο χρειάζεται 14μs – λιγότερο από το μισό της μεθόδου Bridged και $\approx 1\mu$s περισσότερο από την ελάχιστη τιμή μέτρησης με τη μέθοδο PCI-attached. Μια σημαντική επισήμανση είναι ότι μετά τα 4 KB, οι μετρήσεις χρόνου απόκρισης αυξάνονται. Οι τιμές τους μεταβάλλονται ανάλογα με το μέγεθος του μηνύματος μετά το κατώφλι του MTU (9000 bytes), λόγω της διάσπασης του μηνύματος ανά δύο σελίδες (MTU=9000 bytes, PAGE_SIZE=4 KB). Για μεγάλα μηνύματα (πάνω από 32 KB, LARGE) η επιβάρυνση της καταχώρησης μνήμης (memory

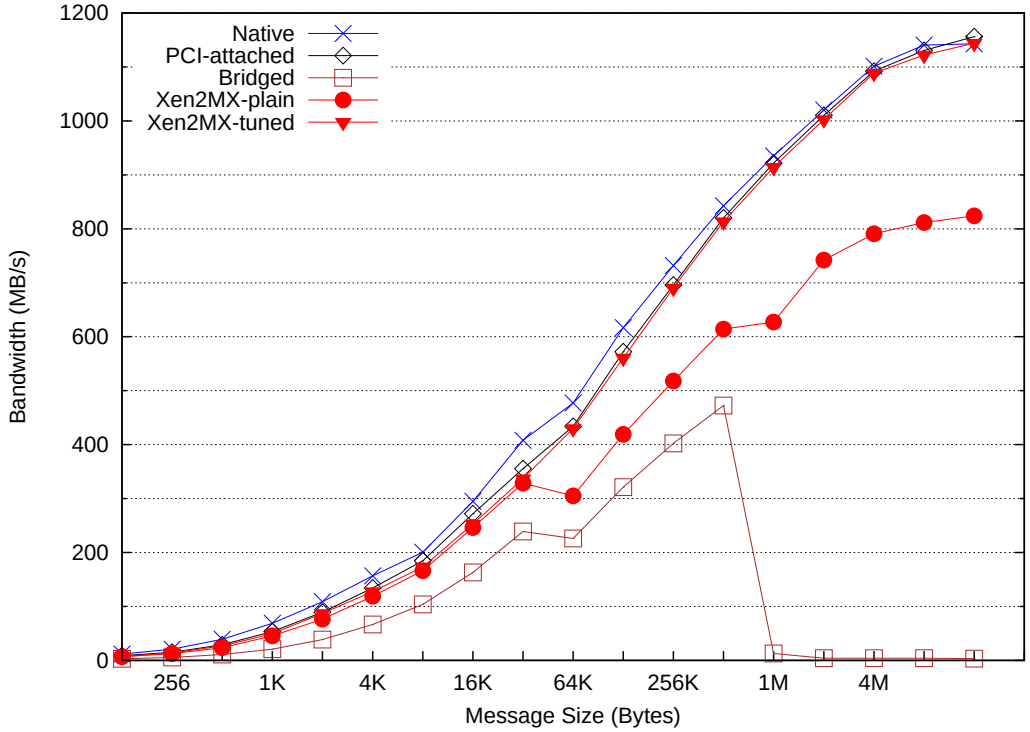
registration) προστίθεται στο Xen2MX-Plain και γίνεται ιδιαίτερα εμφανής σε σύγκριση με τη μέθοδο PCI-attached.



Σχήμα 6.2: Λεπτομέρεια του χρόνου απόκρισης στο Xen2MX μεταξύ μονο-πύρηνης εικονικής μηχανής και συμβατικού συστήματος (χωρίς virtualization).

Μια ενδιαφέρουσα παρατήρηση είναι ότι χωρίς τη χρήση εξειδικευμένου υλικού, το Xen2MX σχεδόν εξαφανίζει την επιβάρυνση στο χρόνο απόκρισης (μειώνοντάς την σε λιγότερο από 1μs)· αυτή η επιβάρυνση ($\approx 4\%$) αποδίδεται στο χρόνο που χρειάζεται για να μεταφερθεί το μήνυμα του frontend στο backend (σχεδόν 200-300ns), καθώς και στη λογική χειρισμού των κλήσεων από το backend (300-400ns).

Στο Σχήμα 6.3 παρουσιάζεται η ρυθμαπόδοση του συστήματος συναρτήσει του μεγέθους του μηνύματος, με βάση τις προηγούμενες μεθόδους. Η μέθοδος Xen2MX-plain είναι φανερά ανώτερη από την Bridged, η οποία παρουσιάζει ασταθή συμπεριφορά· μετά τα 512 KB, η επίδοσή της αγγίζει τα επίπεδα των 10 MB/s. Αυτή η συμπεριφορά αποδίδεται στον τρόπο που οι οδηγοί netback/netfront χειρίζονται τη μεταφορά δεδομένων σε τέτοιους ρυθμούς. Η εξακρίβωση και εκσφαλμάτωση αυτής της συμπεριφοράς ξεπερνά τους σκοπούς της διατριβής και χρειάζεται προσεκτική και εκτενή ανάλυση των στρωμάτων μεταφοράς που λαμβάνουν μέρος στην επικοινωνία. Μια πιθανή εξήγηση είναι η υλοποίηση των κύκλων δέσμευσης / αποδέσμευσης μοιραζόμενης μνήμης που χρησιμοποιούν οι συγκεκριμένοι οδηγοί για τη μεταξύ τους επικοινωνία και ανταλλαγή δεδομένων με το δίκτυο.



Σχήμα 6.3: Ρυθμαπόδοση μεταξύ μονο-πύρηνης εικονικής μηχανής και συμβατικού συστήματος (χωρίς virtualization).

Η μέθοδός μας (Xen2MX-plain) κλιμακώνει όπως η PCI-attached μέχρι ένα συγκεκριμένο μέγεθος μηνύματος (32 KB). Σε αυτό το σημείο γίνεται ορατή η καταχώρηση μνήμης[2]. Μετά το συγκεκριμένο κατώφλι, η Xen2MX-plain παρουσιάζει μειωμένη επίδοση, ανάλογη του μεγέθους μηνύματος, επιτυγχάνοντας 810 MB/s για μηνύματα των 16 MB ενώ η PCI-attached ”γεμίζει” το δίαυλο (1192 MB/s).

Για να εντοπίσουμε το πρόβλημα, αναλύουμε προσεκτικά το κόστος της καταχώρησης μνήμης με στόχο την περαιτέρω εμβάθυνση στα επιμέρους επίπεδα και την ελαχιστοποίηση αυτής της επιβάρυνσης.

Κόστος καταχώρησης μνήμης

Τα Σχήματα 6.4 και 6.5 παρουσιάζουν αναλυτικά το χρόνο που καταναλώνεται σε διάφορα στάδια κατά τη διάρκεια μιας ανταλλαγή μηνύματος, συναρτήσει του μεγέθους του,

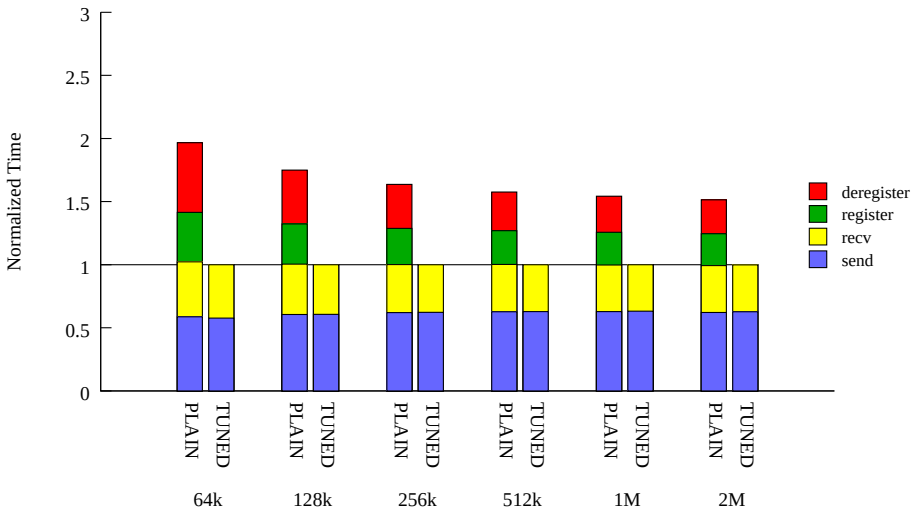
[2] Η καταχώρηση μνήμης δεν είναι πάντα στο κρίσιμο μονοπάτι, ανάλογα με την υλοποίηση του κατώτερου πρωτοκόλλου επικοινωνίας και με τις ανάγκες της εφαρμογής.

για τον ελεγκτή εικονικής μηχανής και την εικονική μηχανή. Συνολικά ανταλλάσσονται (αποστολή / λήψη) 1 GB τυχαίων δεδομένων για 1000 επαναλήψεις και αποτιμάται ο χρόνος των σημαντικότερων επιπέδων του συστήματος. Η πρώτη στήλη (PLAIN) δείχνει το αποτέλεσμα χρησιμοποιώντας τη μέθοδο Xen2MX-plain (MX_RCACHE=0), ενώ η δεύτερη παρουσιάζει το χρόνο των επιμέρους βημάτων με την παράμετρο MX_RCACHE ενεργοποιημένη (TUNED). Ο όγκος των δεδομένων είναι σταθερός για κάθε μέγεθος μηνύματος· παρόλα αυτά, το μέγεθος της μνήμης που καταχωρείται είναι διαφορετικό. Για κάθε επανάληψη, ο αριθμός των μηνυμάτων που ανταλλάσσονται προκύπτει από το εξής: $\frac{1GB}{messagesize}$, όπου το μέγεθος της μνήμης που καταχωρείται είναι ίσο με το μέγεθος του μηνύματος Το πείραμα αυτό δείχνει την επιβάρυνση της καταχώρησης μνήμης δεδομένου του μεγέθους του μηνύματος και καταδεικνύει έναν πιθανό τρόπο παράκαμψης του προβλήματος. Για απλούστευση της παρουσίασης, ο χρόνος κανονικοποιείται στην περίπτωση TUNED για κάθε μέγεθος μηνύματος.

Και στις δύο περιπτώσεις, ο χρόνος της ανταλλαγής των δεδομένων (αποστολή / λήψη) είναι ο ίδιος, δεδομένου του μεγέθους του μηνύματος. Αυτός ο χρόνος αναφέρεται στην επεξεργασία του πρωτοκόλλου καθώς και στη μεταφορά των σχετικών δεδομενογραμμάτων από/προς το δίκτυο. Παρόλα αυτά, ο χειρισμός της καταχώρησης μνήμης γίνεται με διαφορετικό τρόπο. Παρακάτω παρουσιάζουμε αναλυτικά τις μεθόδους που χρησιμοποιούνται:

Περίπτωση PLAIN

Στην περίπτωση του driver domain, η καταχώρηση μνήμης είναι τόσο αργή όσο και η μεταφορά των μικρών μηνυμάτων (SMALL)· μειώνεται σταδιακά όσο το μέγεθος αυξάνει, για να φτάσει περίπου στο μισό για μηνύματα 2 MB. Στην εικονική μηχανή δε συμβαίνει το ίδιο. Αντίστροφα, για μεγάλα μηνύματα ο χρόνος καταχώρησης είναι περίπου 3 φορές μεγαλύτερος από το χρόνο ανταλλαγής δεδομένων. Αυτό είναι αναμενόμενο, καθώς ο διαμοιρασμός μνήμης στο Xen γίνεται κβαντισμένα στο μέγεθος της σελίδας (PAGE_SIZE). Για παράδειγμα, η παραχώρηση μιας περιοχής μνήμης δεσμευμένης από το χώρο χρήστη γίνεται ως εξής: (i) το σύστημα πρέπει να βρει τα mfns για κάθε μία από τις σελίδες, (ii) να αποδεχτεί της αναφορές παραχώρησης, καθώς και (iii) να εκτελέσει όσες κλήσεις παραχώρησης (grant operations) χρειάζονται στον ελεγκτή (για μια περιοχή 2 MB, χρειάζονται 512 αναφορές παραχώρησης). Το Xen προσφέρει λειτουργικότητα για μαζικές αιτήσεις παραχώρησης από το backend. Έτσι, για μικρά μηνύματα, η καταχώρηση μνήμης είναι αργή στο

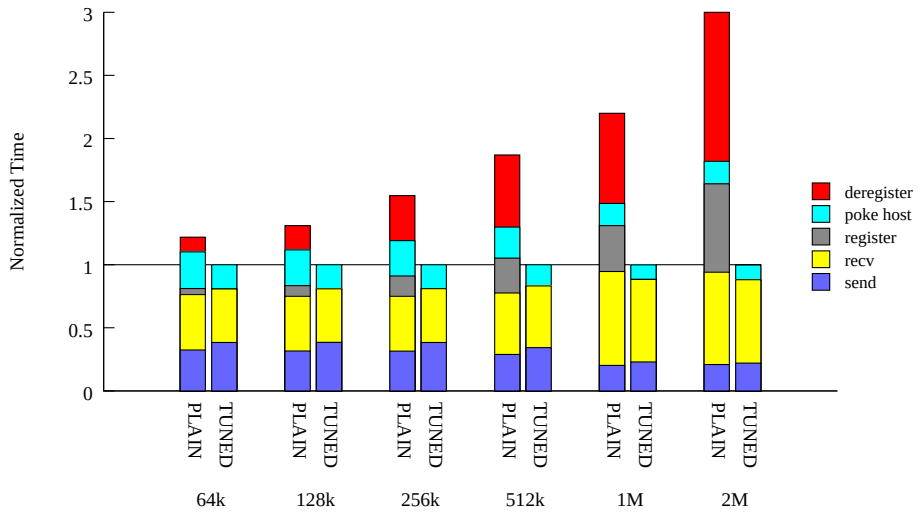


Σχήμα 6.4: Driver domain: ανάλυση του χρόνου που χρειάζεται για την καταχώρηση μνήμης, την ειδοποίηση προς τον ελεγκτή εικονικής μηχανής καθώς και της μεταφοράς δεδομένων συναρτήσει του μεγέθους του μηνύματος. Ο χρόνος είναι κανονικοποιημένος στην TUNED περίπτωση για κάθε μέγεθος μηνύματος.

backend, ενώ για μεγάλα μηνύματα, η συμφόρηση δημιουργείται στο frontend. Και στις δύο περιπτώσεις, ο χειρισμός των αναφορών παραχώρησης είναι το σημαντικότερο κομμάτι της καταχώρησης μνήμης, μειώνοντας τη ρυθμαπόδοση μέχρι και 30% (810 MB/s vs. 1192 MB/s στην περίπτωση της μεθόδου PCI-attached – Σχήμα 6.3).

Περίπτωση TUNED

Όπως περιγράφηκε στην Ενότητα 5.1.3, τα πρωτόκολλα επικοινωνίας υψηλής επίδοσης απαιτούν η μνήμη που χρησιμοποιείται στην επικοινωνία να καταχωρείται εκ των προτέρων. Ένα κομμάτι επιπλέον επιβάρυνσης προστίθεται σε περιβάλλοντα virtualization, όπου η καταχώρηση συνοδεύεται από την απόδοση και αποδοχή μοιραζόμενης μνήμης μεταξύ της εικονικής μηχανής και του ελεγκτή εικονικής μηχανής, λειτουργία εξαιρετικά χρονοβόρα. Για να ξεπεραστεί αυτό, ενεργοποιούμε τη λειτουργία του caching στην καταχώρηση μνήμης (memory registration caching)· αυτή η ρύθμιση επιτρέπει συνεχόμενες μεταφορές μηνυμάτων χωρίς την επιβάρυνση της καταχώρησης για κάθε μεταφορά. Αυτή η προσέγγιση υποθέτει ότι η εφαρμογή δε θα κάνει εξειδικευμένους χειρισμούς των μηχανισμών δέσμευσης μνήμης, καθώς και ότι θα επαναχρησιμοποιεί περιοχές μνήμης για να ισοσταθμίσει το υψηλό κόστος της καταχώρησης μνήμης στην εκκίνησή της. Για πολλές εφαρμογές αυτή η



Σχήμα 6.5: Εικονική μηχανή: ανάλυση του χρόνου που χρειάζεται για την καταχώρηση μνήμης, τη λήψη της ειδοποίησης από την εικονική μηχανή καθώς και της μεταφοράς δεδομένων συναρτήσει του μεγέθους του μηνύματος. Ο χρόνος είναι κανονικοποιημένος στην TUNED περίπτωση για κάθε μέγεθος μηνύματος.

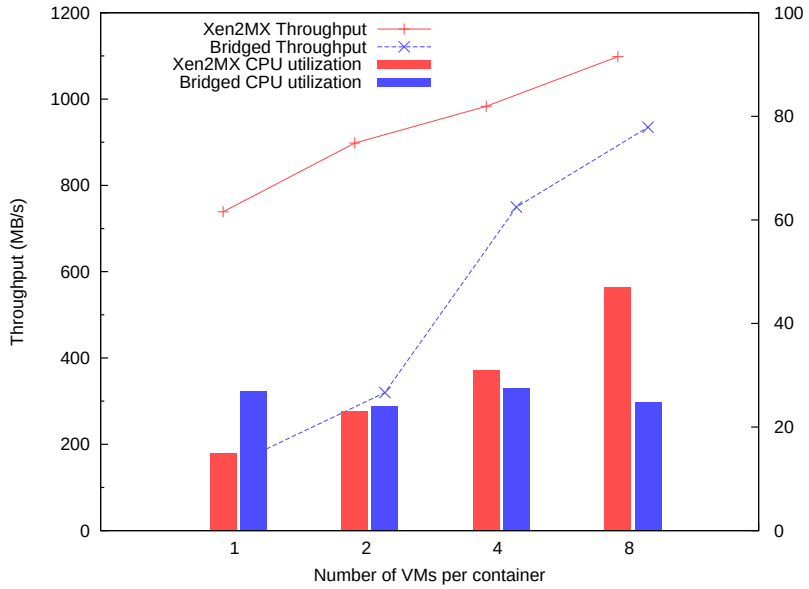
υπόθεση είναι λογική.

Επομένως, με χρήση της παραμέτρου MX_RCACHE=1, ο χρόνος καταχώρησης μνήμης μειώνεται στο ελάχιστο (Σχήματα 6.4 και 6.5, δεξιά). Η βιβλιοθήκη του Open-MX τηρεί αρχείο από ήδη καταχωρημένες περιοχές μνήμης και αντί να δεσμεύει και να αποδίδει (allocate and grant) όλες τις σχετικές σελίδες μιας περιοχής εκ νέου, χρησιμοποιεί τις ήδη υπάρχουσες, μοιραζόμενες περιοχές. Αυτό έχει άμεση επίδραση στο χρόνο απόκρισης και τη ρυθμαπόδοση του συστήματος που παρουσιάζονται στα Σχήματα 6.1 και 6.3.

Με χρήση αυτής της μεθόδου, το Xen2MX μπορεί να κλιμακώνει *ακριβώς* όπως η περίπτωση PCI-attached ακόμα και μετά το κατώφλι των 32 KB, τόσο για το χρόνο απόκρισης όσο και για τη ρυθμαπόδοση. Το Xen2MX εξαφανίζει την επιβάρυνση στο χρόνο απόκρισης που προσδίδεται από τα ενδιάμεσα επίπεδα virtualization (Σχήμα 6.1, Xen2MX-tuned) και σχεδόν αγγίζει το θεωρητικό μέγιστο της δικτυακής γραμμής (Σχήμα 6.3, Xen2MX-tuned), χωρίς την ανάγκη για χρήση εξειδικευμένου εξοπλισμού ή ανελαστικών διατάξεων όπως η περίπτωση της απευθείας πρόσβασης σε υλικό από εικονικές μηχανές.

6.4 Πειραματική αποτίμηση του συστήματος Xen2MX – Κατηγορία II

Το δεύτερο πείραμα περιλαμβάνει δύο επιμέρους διατάξεις: η πρώτη αφορά την αποτίμηση της συνολικής ρυθμαπόδοσης του συστήματος πάνω από το Xen2MX, μεταξύ μεταβλητού αριθμού εικονικών μηχανών που ανταλλάσσουν μηνύματα και βρίσκονται σε διαφορετικά φυσικά μηχανήματα· η δεύτερη αφορά το βαθμό που το Xen2MX επηρεάζει τη χρήση της επεξεργαστικής μονάδας του συστήματος συγκριτικά με την περίπτωση Bridged. Με αυτό τον τρόπο, μπορούν να εξαχθούν συγκεκριμένες επιλογές που πρέπει να γίνουν με στόχο την επίδοση όσον αφορά είτε στη δικτυακή ρυθμαπόδοση είτε στη χρήση των επεξεργαστικών πόρων του συστήματος. Στα πειράματα που ακολουθούν ενεργοποιούμε το caching για την καταχώρηση μνήμης, επομένως το Xen2MX αναφέρεται στο Xen2MX-tuned της προηγούμενης ενότητας. Τα Σχήματα 6.6, 6.7 και 6.8 παρουσιάζουν τα αποτελέσματα των πειραμάτων.



Σχήμα 6.6: Χρήση CPU και ρυθμαπόδοση συναρτήσει του αριθμού των εικονικών μηχανών ανά φυσικό μηχάνημα. Εκκινούμε μέχρι 16 εικονικές μηχανές σε δύο φυσικά μηχανήματα, εκτελούμε το mx_pingpong και καταγράφουμε τη συνολική ρυθμαπόδοση ανά μηχάνημα, καθώς και την ποσοστιαία χρήση των επεξεργαστικών πόρων. Οι στήλες δείχνουν τη ρυθμαπόδοση όπως καταγράφεται στον ελεγκτή εικονικής μηχανής (αριστερά για το Xen2MX, δεξιά για τη μέθοδο Bridged). Αναθέτουμε 4 πυρήνες για κάθε ελεγκτή και παρουσιάζουμε τη χρήση των CPU για κάθε φυσικό μηχάνημα (+ για το Xen2MX και X για τη μέθοδο Bridged).

Το Σχήμα 6.6 παρουσιάζει την αθροιστική ρυθμαπόδοση του πρώτου φυσικού μηχανήματος όταν ανταλλάσσονται μηνύματα μεγέθους 256 KB συναρτήσει του αριθμού των εικονικών μηχανών ανά μηχάνημα που συμμετέχουν στο πείραμα. Οι στήλες δείχνουν τη ρυθμαπόδοση, ενώ οι γραμμές δείχνουν την ποσοστιαία χρήση της επεξεργαστικής μονάδας για το φυσικό μηχάνημα.

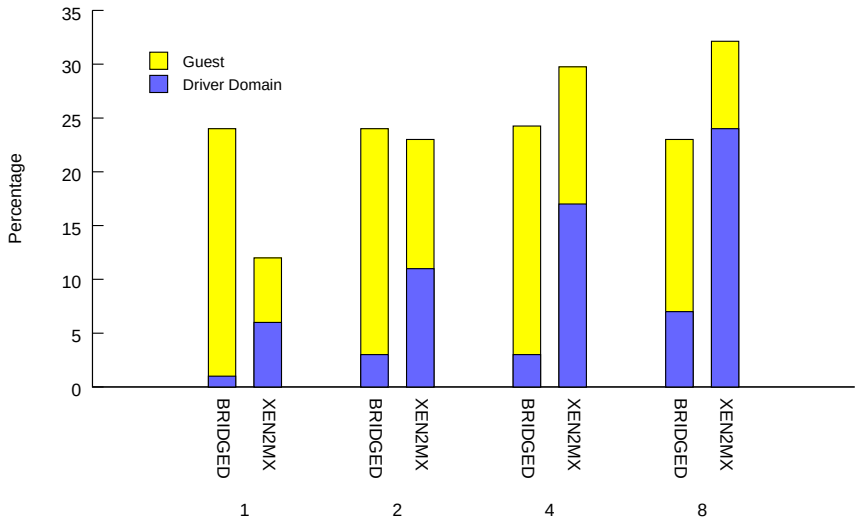
Το Xen2MX αποδίδει καλύτερα από τη μέθοδο Bridged για μηνύματα 256 KB σε όλες τις περιπτώσεις (1 έως 8 εικονικές μηχανές), ενώ στην περίπτωση των 8 εικονικών μηχανών είναι 20% καλύτερο. Η χρήση του CPU για το Xen2MX αυξάνεται γραμμικά με τον αριθμό των εικονικών μηχανών, σε αντίθεση με τη μέθοδο Bridged όπου καταναλώνεται περίπου το ίδιο ποσοστό χρόνου στο CPU ανεξάρτητα από τον αριθμό των εικονικών μηχανών.

Μία από τις βασικές διαφοροποιήσεις του Xen2MX από τη μέθοδο Bridged είναι ότι στην τελευταία, η επεξεργασία του πρωτοκόλλου γίνεται ως επί το πλείστον στην εικονική μηχανή, ενώ στο Xen2MX γίνεται στο backend (δηλαδή στον ελεγκτή εικονικής μηχανής). Για να επιβεβαιωθεί αυτή η υπόθεση, παρουσιάζουμε το μέσο όρο της ποσοστιαίας χρήσης CPU για τις εικονικές μηχανές μαζί με την αντίστοιχη του φυσικού μηχανήματος στο Σχήμα 6.7. Οι σκούρες στήλες δείχνουν τη χρήση CPU για το φυσικό μηχάνημα, ενώ οι ανοιχτές την αντίστοιχη για τις εικονικές μηχανές[3].

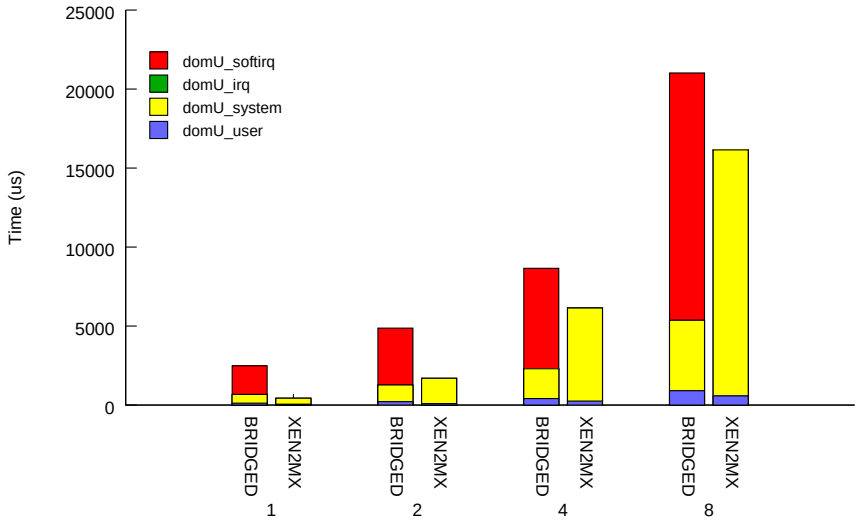
Πράγματι, ο χρόνος χρήσης των επεξεργαστικών μονάδων στην εικονική μηχανή για το Xen2MX είναι λιγότερος από το μισό του χρόνου της μεθόδου Bridged. Για μία εικονική μηχανή, το Xen2MX καταναλώνει περίπου τις μισές μονάδες επεξεργασίας. Για 8 εικονικές μηχανές, καταναλώνει περίπου 8% περισσότερο. Παρόλα αυτά, το Xen2MX ξεπερνά σε επίδοση τη μέθοδο Bridged σε κάθε περίπτωση (αριθμού εικονικών μηχανών καθώς και μεγέθους μηνυμάτων) τόσο όσον αφορά στη ρυθμαπόδοση όσο και στο χρόνο απόκρισης.

Για την περαιτέρω ανάλυση του διαμοιρασμού του χρόνου στα στα επιμέρους επίπεδα της στοίβας του Xen2MX, στο Σχήμα 6.8 παρουσιάζονται τα σημαντικότερα επίπεδα της στοίβας και ο αντίστοιχος χρόνος που καταναλώνεται σε αυτά. Οι μετρήσεις αυτές εξήχθησαν με την ακόλουθη μέθοδο: μεταφέρονται 1G δεδομένων με χρήση του mx_pingpong μεταξύ όλων των εικονικών μηχανών (8 + 8) σε μηνύματα μεγέθους 128 KB για 1000 επαναλήψεις. Στη συνέχεια, παρατηρούνται για συγκεκριμένο χρονικό διάστημα όλες οι μετρήσεις χρόνου των επεξεργαστικών μονάδων τόσο στην εικονική μηχανή, όσο και στο φυσικό μηχάνημα, καθώς και η συνολική ρυθμαπόδοση όπως φαίνεται από τον προσαρμογέα δικτύου.

[3] Τα ποσοστά είναι ανάλογα των συνολικών επεξεργαστικών μονάδων του εκάστοτε συστήματος· για παράδειγμα, 100% χρήση μιας διπύρηνης εικονικής μηχανής είναι δύο φορές περισσότερο από 100% χρήση μίας αντίστοιχης μονοπύρηνης.



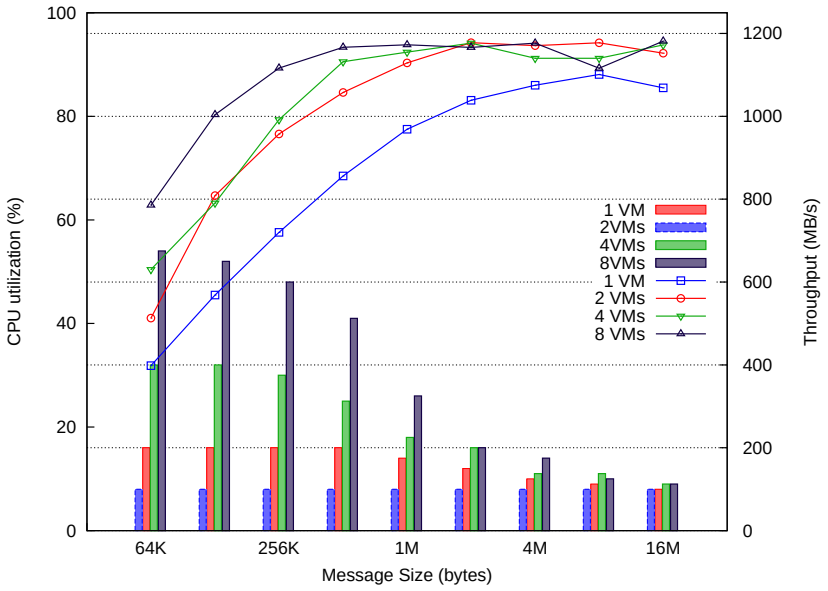
Σχήμα 6.7: Χρήση του CPU για τον ελεγκτή εικονικής μηχανής και τις εικονικές μηχανές συναρτήσει του αριθμού των εικονικών μηχανών ανά φυσικό μηχάνημα. Οι σκούρες στήλες δείχνουν τη χρήση του CPU για το φυσικό μηχάνημα, ενώ οι ανοιχτές στήλες την αντίστοιχη μέτρηση για τις εικονικές μηχανές (μέσος όρος).



Σχήμα 6.8: Χρόνος στα σημαντικότερα επίπεδα επεξεργασίας της εικονικής μηχανής συναρτήσει του αριθμού των εικονικών μηχανών ανά φυσικό μηχάνημα.

Στο Σχήμα 6.8, παρουσιάζεται ο μέσος όρος της κάθε μέτρησης για όλες τις εικονικές μηχανές. Σαν αποτέλεσμα, είναι δυνατή η ποιοτική ανάλυση του βαθμού επιρροής στη χρήση επεξεργαστικών πόρων και για τις δύο μεθόδους (Xen2MX και Bridged).

Ο χρόνος του Xen2MX είναι κυρίως χρόνος συστήματος (system time), ενώ στη μέθοδο Bridged είναι χρόνος σε χειρισμό εικονικών διακοπών (softirq). Μια σημαντική παρατήρηση που επιβεβαιώνει ευρήματα προηγούμενων ενοτήτων είναι ότι η δική μας προσέγγιση (Xen2MX) καταναλώνει σημαντικά λιγότερο χρόνο από τη μέθοδο Bridged φτάνοντας περίπου το 20% λιγότερο για 8 εικονικές μηχανές (≈ 5ms για το παραπάνω πείραμα). Ο λόγος αυτής της συμπεριφοράς είναι ότι το Xen2MX μπορεί να χειριστεί λειτουργίες σημασιολογικά, με κλήσεις δικτύου διασύνδεσης υψηλής επίδοσης (λειτουργίες αποστολής / λήψης μεγάλου όγκου δεδομένων, ομαδοποίηση λειτουργιών για εκτέλεση στο backend κλπ.), σε αντίθεση με τη μέθοδο Bridged που απλά χειρίζεται δεδομενογράμματα Ethernet (με μέγιστο κατακερματισμό τάξης μεγέθους MTU). Παρόλα αυτά, λόγω της υλοποίησης του backend, οι συνολικές μετρήσεις για χρήση επεξεργαστικών πόρων (τόσο στον ελεγκτή όσο και στις εικονικές μηχανές) είναι 8% παραπάνω από τη μέθοδο Bridged.



Σχήμα 6.9: Ρυθμαπόδοση και ποσοστό χρήσης του CPU συναρτήσει του μεγέθους του μηνύματος και του αριθμού των εικονικών μηχανών ανά φυσικό μηχάνημα. Εκκινούμε μέχρι 16 εικονικές μηχανές και παρουσιάζουμε τη συνολική ρυθμαπόδοση ανά φυσικό μηχάνημα για όλες τις περιπτώσεις (1, 2, 4 και 8 εικονικές μηχανές), όπως καταγράφεται από τις εικονικές μηχανές. Ταυτόχρονα, παρατηρούμε τη χρήση του CPU του φυσικού μηχανήματος που αφορά στον ελεγκτή εικονικής μηχανής (στον οποίο αναθέτουμε 4 πυρήνες). Η απόφαση για την αντιστοίχηση εικονικών – φυσικών πυρήνων γίνεται από το χρονοδρομολογητή του Xen.

Για την περαιτέρω εμβάθυνση στο βαθμό κλιμάκωσης του Xen2MX όσο προστίθενται εικονικές μηχανές παρουσιάζονται στο Σχήμα 6.9 τα αποτελέσματα της ποσοστιαίας χρήσης CPU καθώς και της συνολικής ρυθμαπόδοσης του φυσικού μηχανήματος συναρτήσει του μεγέθους των μηνυμάτων και του συνολικού αριθμού εικονικών μηχανών ανά φυσικό μηχάνημα.

Εκκινούμε μέχρι 8 μονοπύρηνες εικονικές μηχανές ανά φυσικό μηχάνημα και καταγράφουμε τη συνολική ρυθμαπόδοση όλων των περιπτώσεων (1, 2, 4 και 8 εικονικές μηχανές) συναρτήσει του μεγέθους του μηνύματος. Ταυτόχρονα, παρατηρούμε τη χρήση του CPU στο φυσικό μηχάνημα, στο οποίο αναθέτουμε 4 πυρήνες. Το Σχήμα 6.9 επιβεβαιώνει τα προηγούμενα ευρήματα και δείχνει ότι η χρήση του CPU αυξάνει αναλογικά με τον αριθμό των εικονικών μηχανών αλλά μειώνεται για μηνύματα μεγάλου μεγέθους. Για μικρότερα μηνύματα ($<$ 256 KB) η συνολική ρυθμαπόδοση αυξάνεται αναλογικά με τον αριθμό των εικονικών μηχανών· το Xen2MX φτάνει τη μέγιστη σχεδόν δικτυακή επίδοση από μηνύματα μεγέθους 256 KB όταν 8 εικονικές μηχανές πιέζουν το σύστημα ($\approx$ 55% χρήση των επεξεργαστικών πόρων).

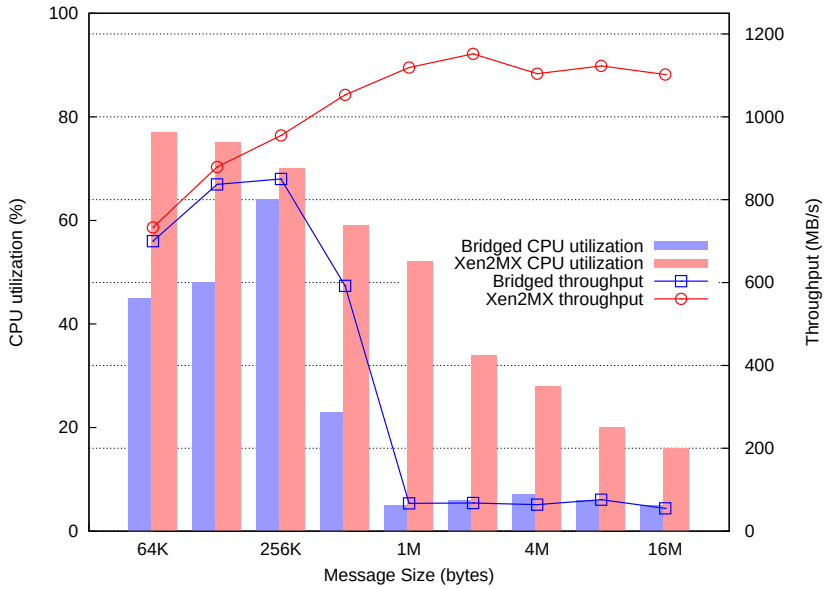
Για μηνύματα μεγέθους $>$ 512 KB, το Xen2MX φτάνει τη μέγιστη επίδοση ενώ, ταυτόχρονα, η χρήση CPU μειώνεται σημαντικά ($\approx$ 9% για 8 εικονικές μηχανές που ανταλλάζουν μηνύματα μεγέθους 16 MB).

Επιπλέον, μια ενδιαφέρουσα παρατήρηση είναι ότι όσο προστίθενται εικονικές μηχανές στο σύστημα, το Xen2MX μειώνει το κατώφλι για τη μέγιστη δικτυακή επίδοση· για παράδειγμα, όταν μία εικονική μηχανή εκτελείται, η συνολική ρυθμαπόδοση είναι $\approx$ 850 MB/sec για μηνύματα 512 KB, ενώ όταν εκτελούνται 8, επιτυγχάνεται ρυθμός αντίστοιχος του θεωρητικού μέγιστου.

6.4.1 Βαθμός κλιμάκωσης

Σε αυτή την ενότητα παρουσιάζεται ο τρόπος με τον οποίο το Xen2MX κλιμακώνει με μεγάλο αριθμό εικονικών μηχανών όσον αφορά στη ρυθμαπόδοση και το χρόνο απόκρισης σε σύγκριση με τη μέθοδο Bridged. Εκκινούμε μέχρι 20 εικονικές μηχανές σε κάθε φυσικό μηχάνημα (40 σύνολο) και πιέζουμε το σύστημα με αποστολές και λήψεις μηνυμάτων προσομοιώνοντας μια ρεαλιστική περίπτωση εκτέλεσης.

Ρυθμίζουμε το σύστημα να υποστηρίζει hyper-threading για να αξιοποιήσουμε την πολυνηματική αρχιτεκτονική των επεξεργαστών (Πίνακας 6.1), και δεν αναθέτουμε εικονικά CPU σε φυσικούς πυρήνες. Ο ελεγκτής εικονικής μηχανής διαθέτει 4 πυρήνες και, όπως



Σχήμα 6.10: Ρυθμαπόδοση και ποσοστιαία χρήση επεξεργαστικών πόρων ενώ πιέζουμε το σύστημα. Εκκινούμε μέχρι 40 εικονικές μηχανές στα δύο φυσικά μηχανήματα και εκτελούμε το πείραμα της Κατηγορίας II. Καταγράφουμε τη συνολική, αθροιστική ρυθμαπόδοση όπως παρουσιάζεται από το φυσικό μηχάνημα και ταυτόχρονα παρατηρούμε τη χρήση των επεξεργαστικών πόρων του συστήματος και για τις δύο περιπτώσεις (Bridged και Xen2MX). Ενεργοποιούμε τη ρύθμιση για πολυνηματική εκτέλεση (hyper-threading) και αφήνουμε την αντιστοίχιση των εικονικών – φυσικών πυρήνων στο χρονοδρομολογητή του Xen.

διαπιστώνουμε, ο χρονοδρομολογητής του Xen διασφαλίζει ότι δεν επικαλύπτονται με πυρήνες στους οποίους εκτελούνται εικονικές μηχανές. Η διάταξη του πειράματος είναι όπως στην Κατηγορία II, με τη διαφορά ότι αντί για 16, δημιουργούνται 40 εικονικές μηχανές (VM_1...VM_{20} στο $Host_0$ και VM_{21}...VM_{40} στο $Host_1$). Η ανταλλαγή μηνυμάτων γίνεται μεταξύ αντιδιαμετρικών εικονικών μηχανών (VM_1 με VM_{21}, VM_2 με VM_{22} κλπ.).

Οι στήλες παρουσιάζουν τη χρήση του CPU της κάθε περίπτωσης για το φυσικό μηχάνημα, ενώ οι καμπύλες δείχνουν τη συνολική ρυθμαπόδοση όπως δίνεται από το φυσικό μηχάνημα. Οι μετρήσεις αυτές (από το φυσικό μηχάνημα) επιβεβαιώθηκαν με τις αντίστοιχες από τις επιμέρους εικονικές μηχανές.

Με χρήση αυτής της διάταξης, επανεκτελούμε το πείραμα της Κατηγορίας II. Το Σχήμα 6.10 δείχνει τα ακόλουθα: η μέθοδος Bridged παρουσιάζει ασταθή συμπεριφορά για μηνύματα μεγέθους $>$ 512 KB (όπως και στην Ενότητα 6.3). Το Xen2MX ξεπερνά τη μέθοδο Bridged όσον αφορά τη ρυθμαπόδοση για όλα τα μεγέθη μηνύματος (64 KB – 16 MB). Η χρήση

του CPU μειώνεται για μεγάλα μηνύματα, φτάνοντας το 18% για 16 MB, ενώ σχεδόν αγγίζει τη μέγιστη θεωρητική δικτυακή επίδοση. Αυτό επιβεβαιώνει ότι το Xen2MX κλιμακώνει αποδοτικά.

7

Σύνοψη

Οι υποδομές cloud computing προσφέρουν μεγάλη υπολογιστική ισχύ και φιλοξενούν ένα ευρύ φάσμα εφαρμογών που κυμαίνονται από υπηρεσιοστρεφείς εφαρμογές μέχρι επιστημονικές προσομοιώσεις υψηλών απαιτήσεων. Οι εφαρμογές υψηλών απαιτήσεων (HPC applications) συνήθως εκτελούνται κατανεμημένα, σε μεγάλο αριθμό κόμβων, με αποτέλεσμα η επικοινωνία μεταξύ των κόμβων να παίζει σημαντικό ρόλο στη συνολική επίδοση. Η εκτέλεσή τους σε εικονικά περιβάλλοντα προϋποθέτει την απαλοιφή των ενδιάμεσων επιπέδων του virtualization που προσδίδουν σημαντική επιβάρυνση τόσο στη ρυθμαπόδοση της επικοινωνίας όσο και στο χαμηλό χρόνο απόκρισης κατά την ανταλλαγή μηνυμάτων μεταξύ των κόμβων. Ταυτόχρονα, πρέπει να διατηρούνται τα πλεονεκτήματα του εικονικού περιβάλλοντος, όπως ευελιξία στην εκτέλεση, απομόνωση και ευκολία στη διαχείριση των υποδομών. Οι σύγχρονες μέθοδοι για Ε/Ε σε εικονικά περιβάλλοντα είτε παρουσιάζουν μειωμένη επίδοση στην επικοινωνία, είτε απαιτούν εξειδικευμένο υλικό που πολυπλοκοποιεί την εγκατάσταση των υποδομών και μειώνει σημαντικά την ευελιξία στη διαχείρισή τους.

Ξεκινώντας από αυτές τις παρατηρήσεις, εστιάσαμε σε μηχανισμούς αποδοτικής ανταλλαγής μηνυμάτων μεταξύ εικονικών μηχανών εικονικές με στόχο την εξάλειψη των επιμέρους επιβαρύνσεων λόγω των υποσυστημάτων του virtualization.

Συγκεκριμένα, περιγράψαμε ενδελεχώς τις τεχνολογίες στις οποίες βασίζεται η παρούσα μελέτη: αναλύσαμε τη βιβλιογραφία για δίκτυα υψηλής επίδοσης σε συστοιχίες υπολογιστικών κόμβων, καθώς και για τεχνολογίες virtualization κατώτερων στρωμάτων. Στη συνέχεια, περιγράψαμε εναλλακτικά μονοπάτια που αναδεικνύουν το ενδιαφέρον ανορθόδοξων λύσεων για αποδοτική μεταφορά δεδομένων από / προς το δίκτυο και καταγράψαμε τις βασικές επιλογές σχεδίασης για ένα πρωτόκολλο ανταλλαγής μηνυμάτων υψηλής επίδοσης για εικονικά περιβάλλοντα. Επιπρόσθετα, περιγράψαμε αναλυτικά τις δυνατότητες δικτύωσης σε σύγχρονες πλατφόρμες virtualization.

Μελετήσαμε τους συμβιβασμούς που γίνονται μεταξύ της χρήσης βελτιστοποιήσεων στο

λογισμικό και στο υλικό και αναλύσαμε τους τρόπους αξιοποίησης των μεθόδων δικτύωσης για την απρόσκοπτη εκτέλεση εφαρμογών υψηλής επίδοσης στο cloud με τη μέγιστη δυνατή επίδοση.

Περιγράψαμε το σχεδιασμό και την υλοποίηση του Xen2MX, ενός πρωτοκόλλου ανταλλαγής μηνυμάτων υψηλής επίδοσης για εικονικά περιβάλλοντα. Τα χαρακτηριστικά του Xen2MX ελαχιστοποιούν και, σε περιπτώσεις, εξαλείφουν προβλήματα που σχετίζονται με τους κοινούς οδηγούς συσκευών που χρησιμοποιούνται σε υποδομές cloud στο πλαίσιο του HPC. Πιο συγκεκριμένα, ελαχιστοποιεί την επιβάρυνση του χειρισμού γεγονότων για ανταλλαγή μηνυμάτων και βελτιώνει σημαντικά τη ρυθμαπόδοση με: (a) χρήση τεχνικών μηδενικών αντιγραφών (zero-copy) για μεγάλα μηνύματα, (b) επαναχρησιμοποίηση αντιστοιχίσεων μνήμης μεταξύ εικονικών μηχανών και του ελεγκτή, (c) αποδέσμευση των μηνυμάτων ελέγχου από τη μεταφορά δεδομένων, χάρη στη βελτιστοποιημένη έκδοση του μηχανισμού καταναλωτή–παραγωγού για την επικοινωνία με στρώματα του λειτουργικού συστήματος και του ελεγκτή εικονικών μηχανών. Ο σχεδιασμός του Xen2MX μπορεί να εφαρμοστεί σε οποιοδήποτε ελεγκτή υποστηρίζει δυνατότητες paravirtualization.

Εκτελέσαμε πειράματα για την αποτίμηση του Xen2MX καθώς και για να αναλύσουμε τη συμπεριφορά του. Στόχος μας ήταν να αναδείξουμε συγκεκριμένα χαρακτηριστικά της προσέγγισής μας, σε σύγκριση με συμβατικούς τρόπους επικοινωνίας που χρησιμοποιούνται σήμερα σε εικονικά περιβάλλοντα. Δείχνουμε πως το Xen2MX επιφέρει σημαντική αύξηση του ρυθμού ανταλλαγής μηνυμάτων (ανεξαρτήτως μεγέθους) και, έτσι, επιτρέπει την απρόσκοπτη εκτέλεση εφαρμογών απαιτητικών σε Ε / Ε σε εικονικά περιβάλλοντα. Τέλος, μετρήσεις επίδοσης με ρεαλιστικά μετροπρογράμματα πάνω από ένα πραγματικό εικονικό περιβάλλον έδειξαν ότι το Xen2MX μπορεί να μειώσει την επίδραση των επιπέδων του virtualization δίνοντας τον ελάχιστο χρόνο απόκρισης, άμεσα συγκρίσιμο ($\approx$ 96%) με μεθόδους που απαιτούν εξειδικευμένο υλικό.

Ο κυριότερος λόγος για τον οποίο οι τεχνικές virtualization δεν αποτελούν φιλόξενο περιβάλλον για εφαρμογές υψηλής επίδοσης είναι η επιβάρυνση των επιπέδων του virtualization στο υποσύστημα Ε / Ε. Πρόσφατες εργασίες σε αυτόν τον τομέα [44] δείχνουν πως η κλιμάκωση πολλών εικονικών μηχανών σε ένα φυσικό μηχάνημα καθώς και η αποδοτική μεταφορά δεδομένων από / προς το δίκτυο αποτελούν τα σημαντικότερα προβλήματα.

Με βάση τα παραπάνω, το Xen2MX επιτυγχάνει χαμηλό χρόνο απόκρισης καθώς και ρυθμούς μετάδοσης δεδομένων σχεδόν ισάξιους με τους αντίστοιχους σε φυσικά μηχανήματα (περίπου στο 96%) χωρίς τη χρήση εξειδικευμένου υλικού με κόστος περίπου 8% των επεξεργαστικών μονάδων του συστήματος συγκριτικά με την κλασική περίπτωση. Σχετικά

με την κλιμάκωση, το Xen2MX χειρίζεται αποδοτικά μεγάλο αριθμό εικονικών μηχανών στο ίδιο φυσικό μηχάνημα χωρίς να επηρεάζεται ο ρυθμός μετάδοσης.

Ο σχεδιασμός του Xen2MX είναι αρκετά γενικός ώστε να εφαρμόζεται σε όλες τις περιπτώσεις paravirtualization, δίνοντας, με αυτό τον τρόπο, μια απλή, κλιμακώσιμη μέθοδο για εκτέλεση εφαρμογών απαιτητικών σε δικτυακή Ε / Ε σε περιβάλλοντα Cloud με χρήση συμβατικού εξοπλισμού. Η διάρθρωση της σχεδίασης, βοηθά επίσης την ενσωμάτωση διαφορετικών πρωτοκόλλων, όπως για παράδειγμα το CCI [2].

Η χρήση περισσότερων εικονικών μηχανών υπηρεσίας (driver domains) θα μπορούσε να βελτιώσει το βαθμό συγχώνευσης του συστήματος. Η διμερής λειτουργία καταχώρησης μνήμης, σε συνδυασμό με το μηχανισμό καταναλωτή–παραγωγού υψηλής επίδοσης μπορεί να επιτύχει την ίδια απόδοση με το ελάχιστο κόστος σε επεξεργαστικές μονάδες.

Βιβλιογραφία

[1] InfiniBand Trade Association. Infiniband architecture specification, release 1.0, 2000, 2000. http://www.infinibandta.org/specs.

[2] Scott Atchley, David Dillow, Galen M. Shipman, Patrick Geoffray, Jeffrey M. Squyres, George Bosilca, and Ronald Minnich. The Common Communication Interface (CCI). In *IEEE 19th Annual Symposium on High Performance Interconnects, HOTI 2011, Santa Clara, CA, USA, August 24-26, 2011*, 2011.

[3] Florian Auernhammer and Patricia Sagmeister. Architectural support for user-level network interfaces in heavily virtualized systems. In *WIOV'10: Proceedings of the 2nd conference on I/O virtualization*, pages 7–7, Berkeley, CA, USA, 2010. USENIX.

[4] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, pages 41–41, Berkeley, CA, USA, 2005. USENIX Association.

[5] Muli Ben-Yehuda, David Breitgand, Michael Factor, Hillel Kolodner, Valentin Kravtsov, and Dan Pelleg. NAP: a building block for remediating performance bottlenecks via black box network analysis. In *ICAC '09: Proc. of the 6th Intern. Conference on Autonomic computing*, pages 179–188, NY, USA, 2009. ACM.

[6] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. Su. Myrinet: A gigabit-per-second local area network. *IEEE Micro*, 15(1):29–36, Feb 1995.

[7] François Diakhaté, Marc Pérache, Raymond Namyst, and Hervé Jourdren. Efficient shared memory message passing for inter-vm communications. In *4th Workshop on Virtualization in High-Performance Cloud Computing (VHPC '08), Euro-par 2008*, 2008.

[8] Yaozu Dong, Jinquan Dai, Zhiteng Huang, Haibing Guan, Kevin Tian, and Yunhong Jiang. Towards high-quality I/O virtualization. In *SYSTOR '09: Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, pages 1–8, NY, USA, 2009. ACM.

[9] Yaozu Dong, Zhao Yu, and Greg Rose. SR-IOV networking in Xen: architecture, design and implementation. In *WIOV'08: Proceedings of the First conference on I/O virtualization*, pages 10–10, Berkeley, CA, USA, 2008. USENIX.

[10] Computer Desktop Encyclopedia. virtual machine monitor.

[11] P. Geoffray. Opiom: off-processor io with myrinet. In *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, pages 261 –268, 2001.

[12] Robert P. Goldberg Gerald J. Popek. Formal requirements for virtualizable third generation architectures. *ACM*, 17:412–421, 1974.

[13] Brice Goglin. High-Performance Message Passing over generic Ethernet Hardware with Open-MX. *Parallel Computing*, 37(2):85–100, February 2011. Open-MX.

[14] Abel Gordon, Nadav Amit, Nadav Har'El, Muli Ben-Yehuda, Alex Landau, Dan Tsafrir, and Assaf Schuster. ELI: Bare-Metal Performance for I/O Virtualization. In *ACM Architectural Support for Programming Languages & Operating Systems (ASPLOS)*, 2012.

[15] Abel Gordon, Nadav Har'El, Alex Landau, Muli Ben-Yehuda, and Avishay Traeger. Towards Exitless and Efficient Paravirtual I/O. In *The 5th Annual International Systems and Storage Conference (SYSTOR)*, 2012.

[16] S. Gurumurthi, S. Sankar, and M. R. Stan. Using intradisk parallelism to build energy-efficient storage systems. *Micro, IEEE*, 29(1):50–61, 2009.

[17] Timo Hirt. Kvm - the kernel-based virtual machine. February 2010.

[18] Wei Huang, Matthew J. Koop, Q. Gao, and Dhabaleswar K. Panda. Virtual machine aware communication libraries for high performance computing. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, NY, USA, 2007. ACM.

[19] Sven Karlsson, Stavros Passas, George Kotsis, and Angelos Bilas. MultiEdge: An Edge-based Communication Subsystem for Scalable Commodity Servers. page 28, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[20] David Riddoch Kieran Mansley, Greg Law. Getting 10 gb/s from xen: Safe and fast device access from unprivileged domains. In *Euro-Par 2007 Workshops: Parallel Processing*, 2007.

[21] Kangho Kim, Jin-Soo Kim, and Sung-In Jung. Gnbd/via: a network block device over virtual interface architecture on linux. In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 7 pp–, 2002.

[22] Kivity, Avi. KVM: the Linux Virtual Machine Monitor. In *OLS '07: The 2007 Ottawa Linux Symposium*, pages 225–230, July 2007.

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

[24] Evangelos Koukis, Anastassios Nanos, and Nectarios Koziris. Gmblock: Optimizing data movement in a block-level storage sharing system over myrinet. *Cluster Computing*, 13:349–372, 2010.

[25] Alex Landau, David Hadas, and Muli Ben-Yehuda. Plugging the hypervisor abstraction leaks caused by virtual networking. In *SYSTOR '10: Proceedings of the 3rd Annual Haifa Experimental Systems Conference*, pages 1–9, NY, USA, 2010. ACM.

[26] Shuang Liang, Weikuan Yu, and D. K. Panda. High performance block i/o for global file system (gfs) with infiniband rdma. In *Parallel Processing, 2006. ICPP 2006. International Conference on*, pages 391–398, 2006.

[27] Jiuxing Liu, Wei Huang, Bulent Abali, and Dhabaleswar 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*, pages 3–3, Berkeley, CA, USA, 2006. USENIX.

[28] Jiuxing Liu, Dhabaleswar K. Panda, Jiuxing Liu Dhabaleswar K. P, and Mohammad Banikazemi. Evaluating the impact of rdma on storage i/o over infiniband. In *SAN-03 Work- shop (in conjunction with HPCA), 2004*, 2004.

[29] Aravind Menon, Alan L. Cox, and Willy Zwaenepoel. Optimizing network virtualization in Xen. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pages 2–2, Berkeley, CA, USA, 2006. USENIX.

[30] Myricom. Myrinet eXpress (MX): A High Performance, Low-Level, Message-Passing Interface for Myrinet, 2006. http://www.myri.com/scs/MX/doc/mx.pdf.

[31] Anastassios Nanos, Georgios Goumas, and Nectarios Koziris. Exploring I/O virtualization data paths for MPI applications in a cluster of VMs: a networking perspective. In *5th Workshop on Virtualization in High-Performance Cloud Computing (VHPC '10)*, Ischia - Naples, Italy, 8 2010.

[32] Anastassios Nanos and Nectarios Koziris. MyriXen: Message Passing in Xen Virtual Machines over Myrinet and Ethernet. In *4th Workshop on Virtualization in High-Performance Cloud Computing*, The Netherlands, 2009.

[33] Anastassios Nanos and Nectarios Koziris. Xen2MX: towards high-performance communication in the cloud. In *7th Workshop on Virtualization in High-Performance Cloud Computing (VHPC '12)*, Rhodes Island, Greece, August 2012.

[34] Anastassios Nanos, Nikos Nikoleris, Stratos Psomadakis, Elisavet Kozyri, and Nectarios Koziris. A Smart HPC Interconnect for Clusters of Virtual Machines. In *6th Workshop on Virtualization in High-Performance Cloud Computing (VHPC '11)*, Bordeaux,France, 2011.

[35] PCI SIG. SR-IOV specification, 2007. http://www.pcisig.com/specifications/iov/single_root/.

[36] H. E. Bal R.A.F Bhoedjang, T. Ruhl. User-level network interface protocols. *IEEE Computer*, pages 53–60, November 1998.

[37] Himanshu Raj and Karsten Schwan. High performance and scalable I/O virtualization via self-virtualized devices. In *HPDC '07: Proceedings of the 16th international symposium on High performance distributed computing*, pages 179–188, NY, USA, 2007. ACM.

[38] Kaushik Kumar Ram, Jose Renato Santos, and Yoshio Turner. Redesigning xen's memory sharing mechanism for safe and efficient I/O virtualization. In *WIOV'10: Proceedings of the 2nd conference on I/O virtualization*, pages 1–1, Berkeley, CA, USA, 2010. USENIX.

[39] Kaushik Kumar Ram, Jose Renato Santos, Yoshio Turner, Alan L. Cox, and Scott Rixner. Achieving 10 Gb/s using safe and transparent network interface virtualization. In *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 61–70, NY, USA, 2009. ACM.

[40] Adit Ranadive and Bhavesh Davda. Toward a Paravirtual vRDMA Device for VMware ESXi Guests. *VMware Technical Journal, Winter 2012*, 1(2), December 2012.

[41] R. Recio, P. Culley, D. Garcia, and J. Hilland. An RDMA Protocol Specification (Version 1.0) This document is a Release Specification of the RDMA Consortium., 2002.

[42] Jose Renato Santos, Yoshio Turner, G. Janakiraman, and Ian Pratt. Bridging the gap between software and hardware techniques for I/O virtualization. In *ATC'08: USENIX 2008 Annual Technical Conference on Annual Technical Conference*, pages 29–42, Berkeley, CA, USA, 2008. USENIX.

[43] Leah Shalev, Julian Satran, Eran Borovik, and Muli Ben-Yehuda. IsoStack—Highly Efficient Network Processing on Dedicated Cores. In *USENIX ATC '10: USENIX Annual Technical Conference*, 2010.

[44] Joshua E. Simons and Jeffrey Buell. Virtualizing high performance computing. *SIGOPS Oper. Syst. Rev.*, 44(4):136–145, December 2010.

[45] Sreekrishnan Venkateswaran. Essential linux device drivers. (TR-0168):231–244, July 2008.

[46] Andrew Whitaker, Marianne Shaw, and Steven D. Gribble. Denali: Lightweight virtual machines for distributed and networked applications. In *In Proc. of the USENIX Annual Technical Conference*, 2002.

[47] Wikipedia. Virtual machine.

[48] B. A. Yassour, M. Ben-Yehuda, and O. Wasserman. Direct device assignment for untrusted fully-virtualized virtual machines. Technical Report H-0263, IBM Research, 2008.

[49] Ben-Ami Yassour, Muli Ben-Yehuda, and Orit Wasserman. On the dma mapping problem in direct device assignment. In *SYSTOR 2010: The 3rd Annual Haifa Experimental Systems Conference*, 2010.

[50] Lamia Youseff, Keith Seymour, Haihang You, Dmitrii Zagorodnov, Jack Dongarra, and Rich Wolski. Paravirtualization effect on single- and multi-threaded memory-intensive linear algebra software. *Cluster Computing*, 12:101–122, 2009.

[51] Lamia Youseff, Rich Wolski, Brent Gorda, and Ra Krintz. Paravirtualization for HPC Systems. In *In Proc. Workshop on Xen in High-Performance Cluster and Grid Computing*, pages 474–486. Springer, 2006.

[52] Alin Zhong, Hai Jin, Song Wu, Xuanhua Shi, and Wei Gao. Performance implications of non-uniform vcpu-pcpu mapping in virtualization environment. *Cluster Computing*, pages 1–12, 2012.

Publications

- A. Nanos and N. Koziris Xen2MX: *Towards High-performance communication in the Cloud*, Proceedings of the 7th Workshop on Virtualization in High-Performance Cloud computing (VHPC 2012), held in conjunction with Euro-par 2012, Rhodes Island, Greece, August 27-31 2012
- A. Nanos, N. Nikoleris, S. Psomadakis, E. Kozyri and N. Koziris *A Smart HPC interconnect for clusters of Virtual Machines*, Proceedings of the 6th Workshop on Virtualization in High-Performance Cloud computing (VHPC 2011), held in conjunction with Euro-par 2011, Bordeaux, France, 29 August - 2 September, 2011
- D. Aragiorgis, A. Nanos and N. Koziris, *Coexisting Scheduling Policies boosting I/O Virtual Machines*, Proceedings of the 6th Workshop on Virtualization in High-Performance Cloud computing (VHPC 2011), held in conjunction with Euro-par 2011, Bordeaux, France, 29 August - 2 September, 2011
- A. Nanos, G. Goumas and N. Koziris, *Exploring I/O Virtualization Data paths for MPI Applications in a Cluster of VMs: A Networking Perspective*, Proceedings of the 5th Workshop on Virtualization in High-Performance Cloud computing (VHPC 2010), held in conjunction with Euro-par 2010, Ischia - Naples 31 August - 3 September, 2010
- A. Nanos and N. Koziris, *MyriXen: Message Passing in Xen Virtual Machines over Myrinet and Ethernet*, Proceedings of the 4th Workshop on Virtualization in High-Performance Cloud computing (VHPC 2009), held in conjunction with Euro-par 2009, Delft, The Netherlands, 24-28 August, 2009
- E. Koukis, A. Nanos and N. Koziris, *GMBlock: Optimizing data movement in a block-level storage sharing system over Myrinet*, Cluster Computing, Vol. 13, No. 4, pp. 349-372, 2010. DOI: 10.1007/s10586-009-0106-y
- E. Koukis, A. Nanos and N. Koziris, *Synchronized Send Operations for Efficient Streaming Block I/O over Myrinet*, Proceedings of the Workshop on Communication Architecture for Clusters (CAC 2008), held in conjunction with the 22nd International Parallel and Distributed Processing Symposium (IPDPS 2008), Miami, FL, USA, 14-18 April, 2008

Βιογραφικό Σημείωμα

Ο Αναστάσιος Νάνος γεννήθηκε στις 17 Μαρτίου 1983 στην Αθήνα. Αποφοίτησε από το Ενιαίο Λύκειο το 2000, οπότε και εισήχθη στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιοου Πολυτεχνείου. Παρακολούθησε με επιτυχία τον κύκλο σπουδών και αποφοίτησε το 2006 με βαθμό 7.10/10. Το ίδιο έτος ξεκινά και η ερευνητική του συνεργασία με το Εργαστήριο Υπολογιστικών Συστημάτων της Σχολής ΗΜΜΥ του ΕΜΠ και το Δεκέμβριο του 2006 γίνεται δεκτός ως υποψήφιος διδάκτορας. Το Δεκέμβριο του 2013 εξετάζεται επιτυχώς στη διδακτορική του διατριβή από την επταμελή εξεταστική επιτροπή και το Μάρτιο του 2013 ορκίζεται διδάκτωρ της Σχολής ΗΜΜΥ του ΕΜΠ.