ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Πειραματική αποτίμηση και βελτιστοποίηση της επικοινωνίας εικονικών μηχανών που συνυπάρχουν στο ίδιο φυσικό μηχάνημα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Μ. Μουζακίτης

Επιβλέπων: Νεκτάριος Κοζύρης

Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Πειραματική αποτίμηση και βελτιστοποίηση της επικοινωνίας εικονικών μηχανών που συνυπάρχουν στο ίδιο φυσικό μηχάνημα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Μ. Μουζακίτης

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις ……. Ιουλίου 2013.

……..	……..	……..
Νεκτάριος Κοζύρης	Δημήτριος Σούντρης	Δημήτριος Φωτάκης
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2013

.....

Κωνσταντίνος Μ. Μουζακίτης

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

Περίληψη

Τα τελευταία χρόνια, οι συστοιχίες εικονικών μηχανών αποτελούν την εναλλακτική λύση των συστοιχιών πραγματικών υπολογιστών, καθώς επιτρέπουν την εκτέλεση πολυνηματικών απαιτητικών εφαρμογών, αλλά συγχρόνως παρέχουν τη δυνατότητα αποδοτικότερου και οικονομικότερου διαμοιρασμού του hardware ανάμεσα στις εφαρμογές που μπορεί να εκτελούνται ταυτόχρονα σε ένα κέντρο δεδομένων. Παράλληλα, οι εικονικές μηχανές προσφέρουν εύκολη διαχείριση και ασφάλεια. Το μειονέκτημα που παραμένει στις συστοιχίες εικονικών μηχανών είναι η επιβάρυνση στον χρόνο εκτέλεσης των εφαρμογών που απαιτούν πρόσβαση στο hardware, καθώς προστίθενται επίπεδα virtualization μεταξύ της εφαρμογής και αυτού. Το αντικείμενο της διπλωματικής ανήκει στο ευρύτερο επιστημονικό πεδίο της εκτέλεσης HPC (High Performance Computing) εφαρμογών σε συστοιχίες εικονικών μηχανών και εστιάζει στην αποδοτικότερη μεταφορά δεδομένων μεταξύ των εικονικών μηχανών. Πιο συγκεκριμένα, η παρούσα εργασία μελετά την επίδοση της μεταφοράς δεδομένων μεταξύ δύο εικονικών μηχανών καθώς επίσης και τη βελτιστοποίηση της. Το εικονικό περιβάλλον δημιουργείται από τον ελεγκτή εικονικής μηχανής Xen και οι εικονικές μηχανές βρίσκονται στο ίδιο φυσικό μηχάνημα.

Για την αποτίμηση της επίδοσης, χρησιμοποιήθηκε ένα κατάλληλα προσαρμοσμένο μετροπρόγραμμα, το οποίο, με χρήση της βιβλιοθήκης libvchan, υπολογίζει το χρόνο της εγγραφής και της ανάγνωσης των δεδομένων μεταβάλλοντας κάθε φορά το μέγεθος των δεδομένων, το ρυθμό με τον οποίο αυτά γράφονται (granularity) και το μέγεθος των buffers επικοινωνίας (ring size). Ο χρόνος αυτός, μετατρέπεται σε bandwidth και παρουσιάζεται σε διαγράμματα στηλών. Η μεταφορά των δεδομένων επηρεάζεται κυρίως από το ring size και λιγότερο από το granularity. Τέλος, παρατηρείται σημαντική αύξηση στην απόδοση της επικοινωνίας μετά την τροποποίηση τμήματος της βιβλιοθήκης libvchan.

Λέξεις Κλειδιά

Εικονική μηχανή, Ελεγκτής εικονικής μηχανής, Διαχωρισμένος οδηγός, Xen, libvchan, μηχανισμός των grants, HPC.

Abstract

Today, with the advent of Virtualization techniques, Virtual Machine (VM) environments are becoming a great trend, providing flexibility, dedicated execution and isolation to a vast number of services. These infrastructures, built on clusters of multicores, offer huge processing power, ideal for mass deployment of compute-intensive applications. However, the use of virtual machines, in clusters, imposes a delay on execution time of applications that require access to hardware, because of added levels of virtualization between the application and itself. The subject of the thesis belongs to the broader scientific field of executing HPC (High Performance Computing) applications on clusters of virtual machines and focuses on efficient data transfer between virtual machines. More specifically, this study investigates the performance of data transfer between two virtual machines as well as its optimization. The virtual environment is created by the Xen VM controller and virtual machines are located on the same physical machine.

To evaluate the performance, a properly adjusted benchmark is used, which, using the libvchan library, measures the time of data writing and reading, changing ,every time, the data size, the rate at which they are written (granularity) and the size of the communication buffers (ring size). The time extracted, is converted into bandwidth and this is shown in bar charts. The data transfer is mainly affected by the ring size. Finally, there is a significant increase, in the communication performance, after the modification of a libvchan's proper section.

Key Words

Virtual Machine, Virtual machine monitor, Split driver, Xen, libvchan, grant mechanism, HPC.

Ευχαριστίες

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

Θα ήθελα να ευχαριστήσω θερμά τον κύριο Κοζύρη για την ευκαιρία που μου έδωσε να ασχοληθώ με το συγκεκριμένο τομέα της επιστήμης των υπολογιστών στο Εργαστήριο Υπολογιστικών Συστημάτων, καθώς και για τις πολύτιμες συμβουλές του καθ' όλη τη διάρκεια περάτωσης της εργασίας.

Η εκπόνηση της διπλωματικής αυτής αποτελεί έμπνευση του υποψήφιου διδάκτορα Αναστάσιου Νάνου. Η βοήθεια του ήταν ανεκτίμητη τόσο για τις καίριες παρεμβάσεις του στην υλοποίηση του μετροπρογράμματος για την πειραματική αποτίμηση της μεταφοράς δεδομένων όσο και για τις τεχνικές γνώσεις που μου μετέδωσε. Χωρίς τη συμβολή του η περάτωση της διπλωματικής εργασίας δε θα ήταν εφικτή.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Επικοινωνία εικονικών μηχανών σε συστοιχία υπολογιστών

Τα σύγχρονα κέντρα δεδομένων προσφέρουν ευελιξία, προσαρμοσμένη εκτέλεση και απομόνωση σε ένα μεγάλο αριθμό εφαρμογών, οι οποίες καλύπτουν πολλά επιστημονικά πεδία, όπως σεισμολογία, ιατρική, φυσική. Τα κέντρα αυτά βασίζονται στη λειτουργία συστοιχιών υπολογιστών για την εκτέλεση εφαρμογών με μεγάλες απαιτήσεις σε υπολογιστική ισχύ. Ο λόγος που οι συστοιχίες υπολογιστών χρησιμοποιούνται σε ευρεία κλίμακα στη σύγχρονη εποχή είναι η κλιμακωσιμότητα που προσφέρουν και ο λόγος κόστους προς απόδοση. Οι συστοιχίες υπολογιστών χρησιμοποιούν παράλληλα υπολογιστικά μοντέλα, τα οποία έχουν ήδη γίνει τρόπος σκέψης στους προγραμματιστές, επειδή εκμεταλλεύονται με τον καλύτερο τρόπο την πολυπύρηνη αρχιτεκτονική, πάνω στην οποία είναι βασισμένα, με σκοπό την επίτευξη της μέγιστης δυνατής επίδοσης σε απαιτητικές εφαρμογές.

Μια συστοιχία υπολογιστών, όμως, δεν αποτελείται πάντα από πραγματικούς υπολογιστές. Τα τελευταία χρόνια υπάρχει μεγάλο ενδιαφέρον στη δημιουργία συστοιχιών με εικονικούς υπολογιστές (εικονικές μηχανές). Αυτό συμβαίνει επειδή οι εικονικές μηχανές προσφέρουν απομόνωση και ασφάλεια στην εκτέλεση τους αλλά και εύκολη και οικονομική διαχείριση, αφού μπορούν να δημιουργηθούν και να εκτελεστούν πολλές εικονικές μηχανές, ταυτόχρονα, στο ίδιο φυσικό μηχάνημα. Έπειτα, ο προγραμματιστής, με τη χρήση ειδικού λογισμικού, μπορεί να τις διαχειριστεί εύκολα. Οι εικονικές μηχανές μπορούν να εκτελούν είτε μεμονωμένες εφαρμογές, είτε να συμμετέχουν στην εκτέλεση πολυνηματικών εφαρμογών. Στη δεύτερη περίπτωση, οι εικονικές μηχανές θα πρέπει να επικοινωνήσουν μεταξύ τους για την ανταλλαγή και την χρησιμοποίηση απαραίτητων δεδομένων, τα οποία δεν είναι διαθέσιμα σε όλες. Την επικοινωνία αυτή διευκολύνουν βιβλιοθήκες, οι οποίες προσφέρουν μια διεπαφή προς τον χρήστη, για την φιλικότερη προς αυτόν χρήση των κλήσεων του πυρήνα. Η επίδοση μιας τέτοιας βιβλιοθήκης μελετάται στην παρούσα διπλωματική καθώς επίσης και η προσπάθεια για την βελτιστοποίηση της επίδοσης αυτής.

1.2 Αντικείμενο της διπλωματικής

Αντικείμενο της διπλωματικής αυτής είναι η αποτίμηση της επίδοσης της μεταφοράς δεδομένων μεταξύ δύο εικονικών μηχανών που βρίσκονται στο ίδιο φυσικό μηχάνημα. Το εικονικό περιβάλλον δημιουργείται από τον ελεγκτή εικονικής μηχανής Xen. Για την ανταλλαγή των δεδομένων μεταξύ των εικονικών μηχανών χρησιμοποιούνται οι μηχανισμοί διαμοιρασμού μνήμης του Xen. Για την αποτίμηση των μηχανισμών αυτών χρησιμοποιείται ένα συνθετικό μετροπρόγραμμα, το οποίο χρησιμοποιεί τη βιβλιοθήκη “libvchan”.

Παράλληλα με την πειραματική αποτίμηση της επικοινωνίας των εικονικών μηχανών, αντικείμενο της διπλωματικής είναι και η βελτιστοποίηση της επίδοσης της επικοινωνίας αυτής, τροποποιώντας κατάλληλα τμήμα του κώδικα της βιβλιοθήκης libvchan. Μετά τη συστηματική μελέτη του, βρέθηκε μια λειτουργία της βιβλιοθήκης, η οποία επέβαλε σημαντικό overhead στην επικοινωνία και για το λόγο αυτό αφαιρέθηκε.

Τα αποτελέσματα τόσο από τον γνήσιο κώδικα της βιβλιοθήκης, όσο και από τον τροποποιημένο παρουσιάζονται σε διαγράμματα στηλών για μεγαλύτερη ευκολία του αναγνώστη.

1.3 Οργάνωση Κειμένου

Στο κείμενο, ο αναγνώστης αποκτά αρχικά μια γενική γνώση για τα εργαλεία και τις έννοιες που χρησιμοποιούνται, και στη συνέχεια παρακολουθεί τα αποτελέσματά και τα συμπεράσματα της μελέτης που πραγματοποιήθηκε. Πιο συγκεκριμένα, στο Κεφάλαιο 2 παρουσιάζεται το θεωρητικό υπόβαθρο που χρειάζεται ο αναγνώστης για να κατανοήσει το υπόλοιπο του κειμένου. Στο Κεφάλαιο 3 αναλύονται τα αποτελέσματα της μελέτης αυτής και η προσπάθεια για βελτιστοποίηση της επικοινωνίας. Τέλος στο Κεφάλαιο 4 αναφέρονται τα γενικά συμπεράσματα της διπλωματικής, καθώς και οι μελλοντικές επεκτάσεις της.

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο παρατίθενται και εξηγούνται έννοιες στις οποίες βασίστηκε η παρούσα εργασία. Οι παρακάτω έννοιες ανήκουν κυρίως σε δύο κατηγορίες: στο λειτουργικό σύστημα Linux και στις εικονικές μηχανές. Περιγράφεται το ευρύτερο επιστημονικό πεδίο στο οποίο ανήκουν οι έννοιες αλλά και ο τρόπος με τον οποίο συνδέονται με την εργασία.

2.1 Linux

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

Στο Linux υπάρχουν δύο διακριτοί τρόποι λειτουργίας (modes) της CPU (κεντρική μονάδα επεξεργασίας). Ο πρώτος ονομάζεται kernel mode, που αναφέρεται επίσης ως system mode. Όταν η CPU είναι σε kernel mode, εκτελείται ασφαλές λογισμικό, και ως εκ τούτου μπορεί να εκτελέσει οποιαδήποτε εντολή και να προσπελάσει οποιεσδήποτε διευθύνσεις μνήμης. Ο δεύτερος είναι ο user mode, ένας μη-προνομιακός τρόπος λειτουργίας για τα προγράμματα χρήστη. [12]

Ένα σημαντικό χαρακτηριστικό του Linux είναι η δυνατότητα επικοινωνίας μεταξύ του χρήστη και του πυρήνα. Ο χρήστης δεν έχει άμεση πρόσβαση στις λειτουργίες του πυρήνα, παρόλα αυτά σε πολλά προγράμματα ζητάει την άδεια για να χρησιμοποιήσει κάποιες από αυτές. Αυτό επιτυγχάνεται με τις λεγόμενες κλήσεις συστήματος (system calls).

Ένα άλλο βασικό κομμάτι του Linux είναι οι οδηγοί (drivers). Οι οδηγοί κάνουν ένα συγκεκριμένο κομμάτι του hardware να αποκρίνεται σε μια καλώς ορισμένη εσωτερική διεπαφή (interface) και αποκρύπτουν πλήρως τις λεπτομέρειες της λειτουργίας της συσκευής προς τον χρήστη. Οι ενέργειες του χρήστη, που έχουν σχέση με αυτή τη συσκευή, εκτελούνται μέσω ενός συνόλου προκαθορισμένων κλήσεων που είναι ανεξάρτητες από τον συγκεκριμένο οδηγό. Η

αντιστοίχιση αυτών των κλήσεων σε συγκεκριμένες λειτουργίες της συσκευής, που δρουν στο πραγματικό hardware είναι αρμοδιότητα του οδηγού. Γενικά, ένας οδηγός μπορεί να αντιστοιχεί σε πραγματική συσκευή hardware αλλά μπορεί να αποτελεί απλώς μέσο επικοινωνίας μεταξύ των εφαρμογών του χρήστη και του πυρήνα.

Στη συνέχεια παρουσιάζονται αναλυτικότερα τα system calls και οι οδηγοί χαρακτήρων, μία δημοφιλής κατηγορία οδηγών.

2.1.1 System Calls

Οι συναρτήσεις στο περιβάλλον του Linux διαφοροποιούνται, ανάλογα με τον τρόπο που ορίζονται.

Οι συναρτήσεις βιβλιοθήκης βρίσκονται σε μια εξωτερική βιβλιοθήκη και η κλήση τους είναι μια τυπική κλήση συνάρτησης. Τα ορίσματα μιας τέτοιας συνάρτησης τοποθετούνται στους καταχωρητές του επεξεργαστή ή στη στοίβα και έπειτα η εκτέλεση μεταφέρεται στην αρχή του κώδικα της συνάρτησης.

Τα system calls είναι ορισμένα στον πυρήνα του Linux. Όταν ένα πρόγραμμα πραγματοποιεί ένα system call, τα ορίσματα της κλήσης παραδίδονται στον πυρήνα και στη συνέχεια λαμβάνει χώρα μια διακοπή λογισμικού (software interrupt), η οποία δίνει τον έλεγχο του προγράμματος στον πυρήνα μέχρι την ολοκλήρωση του system call. Ένα system call διαφέρει από μια συμβατική συνάρτηση και χρειάζεται μια ειδική διαδικασία για να μεταφερθεί ο έλεγχος του προγράμματος στον πυρήνα. Ωστόσο, στη GNU C βιβλιοθήκη (η υλοποίηση της πρότυπης βιβλιοθήκης της C που παρέχεται με τα GNU/Linux συστήματα) τα system calls ενθυλακώνονται με συναρτήσεις για την ευκολότερη κλήση τους. Το σύνολο των system calls του Linux διαμορφώνει την πιο βασική διεπαφή μεταξύ των προγραμμάτων και του πυρήνα. Κάθε system call αναπαριστά μια βασική λειτουργία ή δυνατότητα. Μερικά system calls είναι πολύ «ισχυρά» και μπορούν να επηρεάσουν σε μεγάλο βαθμό το σύστημα. Για παράδειγμα, κάποια system calls επιτρέπουν την απενεργοποίηση του συστήματος ή την κατανομή πόρων του συστήματος και την απαγόρευση χρήσης τους από άλλους χρήστες. Αυτά τα system calls έχουν τον εξής περιορισμό: μπορούν να τα καλέσουν μόνο διεργασίες που εκτελούνται με δικαιώματα υπερχρήστη (superuser). Τέλος να σημειωθεί ότι υπάρχουν περίπου 272 διαφορετικά system calls και η λίστα με τα ονόματά τους βρίσκεται στο αρχείο `/usr/include/asm/unistd.h`. [10]

2.1.2 Οδηγοί χαρακτήρων

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

Οι συσκευές χαρακτήρων (πραγματικές ή όχι) έχουν δύο πλευρές. Η μία προορίζεται για το περιβάλλον χρήστη, έτσι ώστε ο ίδιος να γνωρίζει ποια και πως να την προσπελάσει και η άλλη προορίζεται για τον πυρήνα. Η πρώτη ονομάζεται αρχείο συσκευής (device file), αποτελεί κόμβο του συστήματος αρχείων (filesystem) και ταυτίζεται με το όνομα του κόμβου αυτού. Βρίσκεται συνήθως στον φάκελο `/dev` και χαρακτηρίζεται μοναδικά από δύο αριθμούς, τον κύριο (major) και τον ελάσσονα (minor). Ο κύριος αριθμός καθορίζει τον οδηγό που ελέγχει τη συσκευή και ο ελάσσων αριθμός ξεχωρίζει τη συγκεκριμένη συσκευή από τις υπόλοιπες που μπορεί να ελέγχει ο ίδιος οδηγός. Η δεύτερη αναπαράσταση είναι η δομή με όνομα cdev (struct cdev). Ο πυρήνας, δηλαδή, συντηρεί μια δομή cdev για κάθε συσκευή χαρακτήρων. Ένα από τα πεδία που ορίζονται κατά τη δημιουργία της δομής είναι το ops. Το πεδίο αυτό περιέχει δείκτες στις συναρτήσεις (κλήσεις συστήματος) που παρέχουν τις λειτουργίες του οδηγού και τις οποίες μπορεί να καλεί ο χρήστης από τις εφαρμογές του.

Όταν καλούνται οι συναρτήσεις ενός οδηγού, το λειτουργικό σύστημα περνάει ως επιπρόσθετη παράμετρο μία δομή τύπου file. Η δομή αυτή δημιουργείται από το λειτουργικό σύστημα κατά την κλήση της συνάρτησης open, διαγράφεται πάλι από το ίδιο κατά την κλήση της συνάρτησης close και περιέχει πεδία που περιγράφουν το συγκεκριμένο άνοιγμα της συσκευής. Η δομή αυτή δεν αφορά μόνο τους οδηγούς, αλλά κάθε ανοιχτό αρχείο που βρίσκεται στο σύστημα. Για να γίνει πιο κατανοητό, πρέπει να τονιστεί ότι υπάρχει μία δομή τύπου struct cdev που χαρακτηρίζει τη συσκευή (πραγματική ή όχι) αλλά πολλές δομές τύπου file που αναφέρονται στη συσκευή αυτή και δημιουργούνται κάθε φορά που μία εφαρμογή χρήστη ζητά να προσπελάσει τη συσκευή. [7]

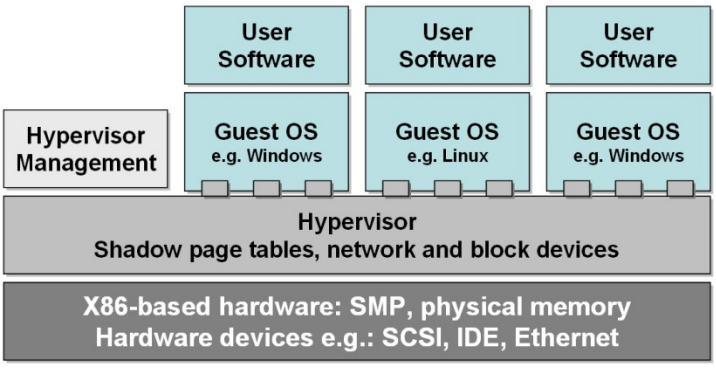
2.2 Εικονική Μηχανή

Εικονική μηχανή είναι η υλοποίηση μιας μηχανής σε λογισμικό, που εκτελεί προγράμματα όπως μια φυσική μηχανή [1]. Ο όρος της εικονικής μηχανής ορίστηκε αρχικά από τους Popek και Goldberg ως εξής: "Εικονική μηχανή θεωρείται ένα αποδοτικό και απομονωμένο αντίγραφο μιας πραγματικής μηχανής"[2]. Ωστόσο η έννοια αυτή σήμερα περιλαμβάνει υλοποιήσεις που δεν αποτελούν αντίγραφα πραγματικών μηχανών. Πάνω σε αυτό το διαχωρισμό βασίζεται η κατηγοριοποίηση των εικονικών μηχανών, πρώτον σε εικονικές μηχανές συστήματος και δεύτερον σε 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 hardware.

Μια εικονική μηχανή συστήματος λειτουργεί σαν να της ανήκουν όλοι οι πόροι του υπολογιστή, αν και την ίδια στιγμή μπορεί να συνυπάρχει με μία ή περισσότερες εικονικές μηχανές συστήματος. Πιο συγκεκριμένα, η εικονική μηχανή συστήματος, που στη συνέχεια του κειμένου αποκαλείται απλώς «εικονική μηχανή» (Virtual Machine - VM), αποτελεί μία προσομοίωση ενός πραγματικού υπολογιστικού συστήματος κατά τη δημιουργία της οποίας ο χρήστης μπορεί να ορίσει γνωστές παραμέτρους συστήματος, όπως τον αριθμό των (εικονικών) πυρήνων, το μέγεθος της μνήμης, το μέγεθος της cache, τον αριθμό mac της κάρτας δικτύου καθώς και τη διεύθυνση IP των διεπαφών της. Επίσης, όπως κάθε υπολογιστής, υποστηρίζει ένα λειτουργικό σύστημα καθώς και την εκτέλεση οποιασδήποτε εφαρμογής του χρήστη. Δύο ή παραπάνω εικονικές μηχανές μπορούν να συνυπάρξουν ταυτόχρονα σε ένα μηχάνημα. Αυτό επιτυγχάνεται με τη βοήθεια του "ελεγκτή εικονικής μηχανής" (Virtual Machine Monitor - VMM). Ο ελεγκτής εικονικής μηχανής είναι ένα λογισμικό που αναλαμβάνει την κατανομή των φυσικών πόρων του hardware σε κάθε εικονική μηχανή μεσολαβώντας στην επικοινωνία των λειτουργικών συστημάτων των εικονικών μηχανών, που στη συνέχεια αποκαλούνται φιλοξενούμενα λειτουργικά συστήματα (guest operating systems), με το hardware.

Για την ευκολότερη κατανόηση των παραπάνω, ακολουθεί ένα σχηματικό διάγραμμα, στο οποίο αναπαριστάται η ιεραρχική τοποθέτηση των τμημάτων του λογισμικού σε σχέση με το hardware ενός υπολογιστή που περιέχει εικονικό περιβάλλον.



Σχήμα 2.1 : Υπολογιστής με εικονικό περιβάλλον

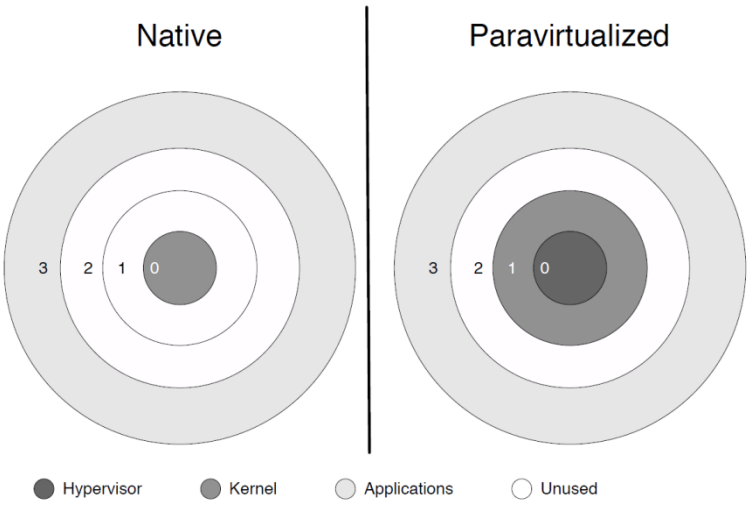
Στην περίπτωση αυτή, ο ελεγκτής εικονικής μηχανής κατέχει τον πλήρη έλεγχο του συστήματος και για το λόγο αυτό ονομάζεται αλλιώς και hypervisor (επιβλέπων). Αυτό σημαίνει ότι εκτελεί τόσες λειτουργίες όσες θα εκτελούσε το λειτουργικό σύστημα σε έναν συμβατικό υπολογιστή και επιπλέον οργανώνει την εκτέλεση των εικονικών μηχανών που δημιουργούνται. Από την 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 hardware [3].

Τα πιο μοντέρνα λειτουργικά συστήματα δεν επιτρέπουν στις εφαρμογές να εκτελούν συγκεκριμένες λειτουργίες. 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.

Σε κάθε στιγμή ο επεξεργαστής λειτουργεί σε ένα από αυτά τα επίπεδα. Το επίπεδο (δακτυλίδι) 0 έχει τα περισσότερα δικαιώματα και το επίπεδο 3 τα λιγότερα. Οι πόροι που προστατεύονται μέσω των επιπέδων είναι: η μνήμη, οι θύρες εισόδου/εξόδου και οι εντολές του επεξεργαστή. Το λειτουργικό σύστημα εκτελείται στο επίπεδο 0 (κατάσταση πυρήνα – kernel mode), οι εφαρμογές του χρήστη στο επίπεδο 3 (κατάσταση χρήστη – user mode) και τα επίπεδα 1 και 2 παραμένουν αχρησιμοποίητα (Σχήμα [2.2], αριστερά). [11]

Όμως, σε έναν υπολογιστή με εικονικό περιβάλλον, ο ελεγκτής εικονικής μηχανής πρέπει να έχει πρόσβαση στο υλικό, δηλαδή στη μνήμη, στον επεξεργαστή και στις συσκευές εισόδου/εξόδου. Αυτό σημαίνει ότι πρέπει να εκτελείται στο επίπεδο με τα περισσότερα δικαιώματα (Σχήμα [2.2], δεξιά). Τα φιλοξενούμενα λειτουργικά συστήματα θα πρέπει να έχουν κι αυτά πρόσβαση σε όλους τους πόρους. Επειδή, όμως, μόνο ένας πυρήνας λειτουργικού συστήματος μπορεί να

βρίσκεται στο επίπεδο 0, τα φιλοξενούμενα λειτουργικά συστήματα πρέπει είτε να εκτελούνται σε άλλο επίπεδο, με λιγότερα δικαιώματα, (επίπεδο 1) είτε να τροποποιηθούν για να εκτελούνται στο επίπεδο 3. [4]



Σχήμα 2.2 : CPU privilege levels σε συμβατικό (native) και εικονικό (paravirtualized) περιβάλλον

Λαμβάνοντας υπόψη δύο κριτήρια, οι ελεγκτές εικονικών μηχανών χωρίζονται σε αντίστοιχες υποκατηγορίες.

Το πρώτο κριτήριο είναι αν ο VMM είναι σε άμεση επαφή με το hardware ή παρεμβάλλεται ένα αρχικό λειτουργικό σύστημα. Το κριτήριο αυτό δημιουργεί δύο κατηγορίες ελεγκτών εικονικής μηχανής: τον τύπο I και τον τύπο II. Στον τύπο I ανήκουν οι ελεγκτές εικονικής μηχανής που τουλάχιστον ένα μέρος τους βρίσκεται σε άμεση επαφή με το hardware, δηλαδή εκτελείται απευθείας πάνω στο hardware. Σε αυτόν τον τύπο υπάρχουν δύο υποκατηγορίες ελεγκτών εικονικής μηχανής: ο επιβλέπων (Hypervisor), που εκτελείται εξολοκλήρου πάνω στο hardware και το λειτουργικό σύστημα υπηρεσίας (Service OS), που ένα μικρό μέρος εκτελείται πάνω στο hardware ενώ το υπόλοιπο στη πρωτεύουσα εικονική μηχανή. 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 των εικονικών μηχανών για πρόσβαση στις συσκευές εισόδου/εξόδου, ανακατευθύνεται μέσω του ελεγκτή (VMM) στους εικονικούς οδηγούς, που βρίσκονται στον ελεγκτή επιπέδου χρήστη (User Level Monitor) και εκείνοι με τη σειρά τους την κατευθύνουν προς τους πραγματικούς οδηγούς του λειτουργικού συστήματος.

Το δεύτερο κριτήριο είναι αν ο VMM προσομοιώνει πλήρως το hardware προς τη πλευρά των εικονικών μηχανών ή όχι. Το κριτήριο αυτό δημιουργεί δύο κατηγορίες τεχνικών virtualization: η πλήρης (full virtualization) και η μερική (paravirtualization). Οι τεχνικές της πρώτης κατηγορίας, απαιτούν ελεγκτές που προσομοιώνουν πλήρως το hardware προς τη πλευρά των εικονικών μηχανών, ενώ οι τεχνικές της δεύτερης κατηγορίας απαιτούν ελεγκτές που παρουσιάζουν εικονικούς οδηγούς στη πλευρά των εικονικών μηχανών και οργανώνουν τη συνομιλία τους με τους πραγματικούς. [7]

Πριν προχωρήσουμε στην αναλυτική περιγραφή δύο βασικών ελεγκτών εικονικών μηχανών, αναφέρονται επιγραμματικά τα πλεονεκτήματα και τα μειονεκτήματα των εικονικών μηχανών. Πρώτα παρουσιάζονται τα πλεονεκτήματα:

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

Παρόλα αυτά πρέπει να αναφερθεί και το βασικό μειονέκτημα των εικονικών μηχανών, το οποίο είναι η καθυστέρηση επικοινωνίας τους με το hardware. Αυτό οφείλεται στα στρώματα που παρεμβάλλονται μεταξύ των guest λειτουργικών συστημάτων και του hardware, λόγω του virtualization και κατ’ επέκταση λόγω του ελεγκτή εικονικής μηχανής.

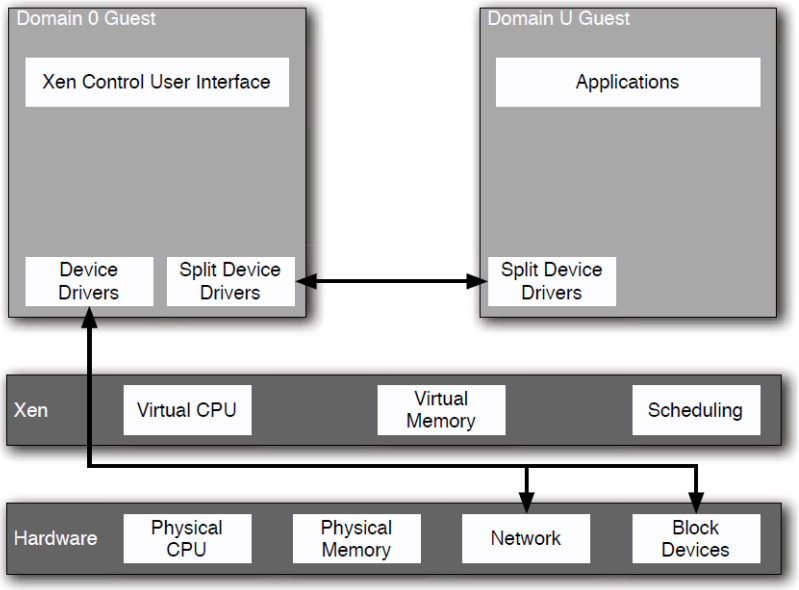
2.2.1 Xen

Το σύστημα Xen προήλθε από μία επιστημονική εργασία του πανεπιστημίου του Cambridge το 2003[7]. Είναι ένας ελεγκτής εικονικής μηχανής, για επεξεργαστή x86, που επιτρέπει σε πολλαπλά, συμβατικά λειτουργικά συστήματα να μοιράζονται ένα συμβατικό hardware με

ασφαλή και αποδοτικό τρόπο. Είναι λογισμικό ανοιχτού κώδικα και βασίζεται στη τεχνική του paravirtualization.

Αρχιτεκτονική

Στο παρακάτω σχήμα φαίνεται η αρχιτεκτονική του Xen.

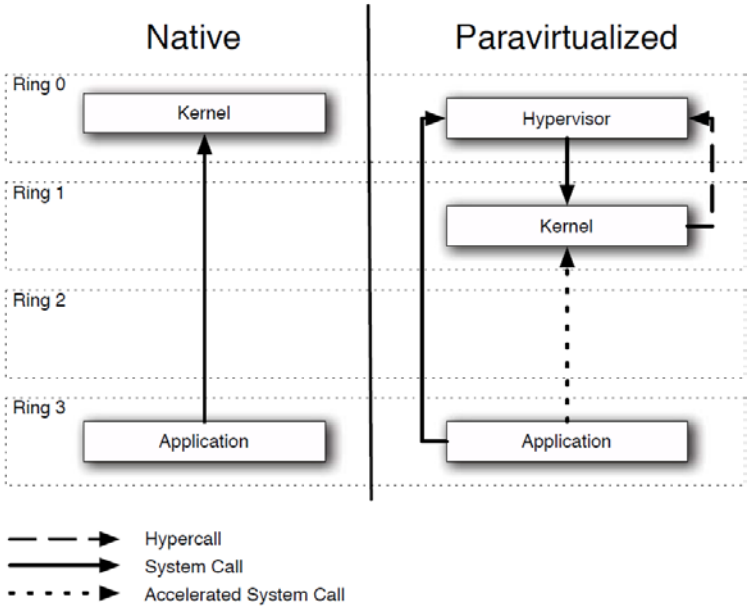


Σχήμα 2.3 : Αρχιτεκτονική του Xen

Όπως φαίνεται από το σχήμα, το Xen είναι τύπου I και πιο συγκεκριμένα, ανήκει στην υποκατηγορία "λειτουργικό σύστημα υπηρεσίας". Το ρόλο της εικονικής μηχανής υπηρεσίας τον παίζει η εικονική μηχανή Domain 0 (Dom0), ενώ η φιλοξενούμενη εικονική μηχανή (Domain U Guest ή DomU) είναι μια κοινή εικονική μηχανή, που επικοινωνεί με το Dom0 για τη προσπέλαση στο hardware. Την επικοινωνία αυτή αναλαμβάνει ένα προνομιούχο VM (το οποίο συνήθως είναι το Dom0), που ονομάζεται Driver Domain και την πραγματοποιεί μέσω του μοντέλου των διαχωρισμένων οδηγών (split driver model). Το Driver Domain περιέχει το "πίσω" μέρος του split driver (backend driver), το οποίο επικοινωνεί άμεσα με τον πραγματικό driver, ενώ τα υπόλοιπα VMs περιέχουν το "μπροστά" μέρος του split driver (frontend driver) το οποίο στέλνει αιτήματα στον backend driver. Επειδή υπάρχει περίπτωση πολλά guest VMs να ζητήσουν ταυτόχρονη πρόσβαση σε μια συσκευή του συστήματος, ο backend driver πρέπει να παρέχει δυνατότητα πολύπλεξης για τα αιτήματα [4].

Το Dom0 δημιουργείται κατά την εκκίνηση του συστήματος (boot time) και έχει δικαίωμα να χρησιμοποιεί τη διεπαφή ελέγχου του συστήματος (control interface), με την οποία μπορεί ανάμεσα στα άλλα να δημιουργεί και να τερματίζει εικονικές μηχανές και να δημιουργεί εικονικές διεπαφές δικτύου. Σε κάθε εικονική μηχανή που δημιουργείται ανατίθεται ένας αριθμός που την προσδιορίζει, ο οποίος ονομάζεται domid. Όσον αφορά τα επίπεδα δικαιωμάτων, ο Xen VMM εκτελείται στο επίπεδο 0, το Dom0 καθώς και τα DomUs εκτελούνται στο επίπεδο 1 και οι εφαρμογές χρήστη στο επίπεδο 3. [4][9]

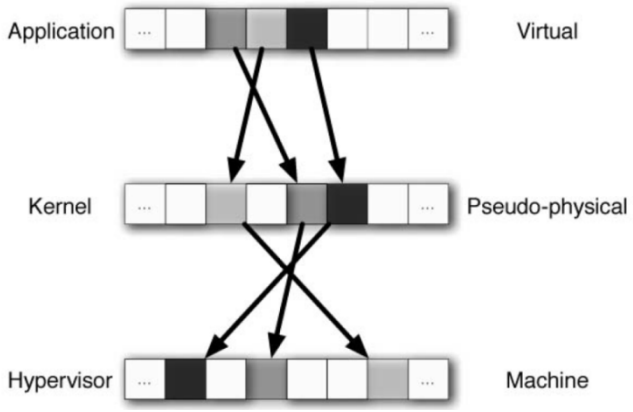
Όταν τα φιλοξενούμενα λειτουργικά συστήματα επιθυμούν να εκτελέσουν μια εντολή με αυξημένα δικαιώματα, επικοινωνούν με το Xen μέσω μιας υπερκλήσης (Hypercall) και εκτελείται η εντολή από τη πλευρά του Xen. Τα hypercalls λειτουργούν σχεδόν με τον ίδιο τρόπο που λειτουργούν και τα system calls σε ένα μηχάνημα χωρίς εικονικές μηχανές. Στην περίπτωση των hypercalls, όταν συμβεί η διακοπή (η οποία έχει διαφορετικό αριθμό στο Xen από αυτόν του συμβατικού πυρήνα του Linux), την εκτέλεση αναλαμβάνει ο ελεγκτής, και όχι ο πυρήνας, ο οποίος στη συνέχεια επιστρέφει τον έλεγχο στο guest λειτουργικό. Η διαφορά αυτή φαίνεται καλύτερα στο παρακάτω σχήμα [4].



Σχήμα 2.4 : System calls σε native και σε paravirtualized περιβάλλοντα

Διαχείριση μνήμης στο Xen

Σε ένα συμβατικό λειτουργικό σύστημα (χωρίς εικονικές μηχανές) κάθε διεργασία έχει το δικό της χώρο μνήμης (address space). Από την δικιά της οπτική γωνία, έχει πρόσβαση σε ένα συνεχόμενο χώρο της φυσικής μνήμης του συστήματος, ενώ στην πραγματικότητα έχει πρόσβαση στην εικονική της μνήμη, της οποίας οι σελίδες χαρτογραφούνται (mapping) από την Μονάδα Διαχείρισης Μνήμης (MMU) στη φυσική μνήμη. [8] Κάτι παρόμοιο πρέπει να κάνει και ένας VMM για τα guest VMs. Αυτό σημαίνει ότι σε ένα εικονικό περιβάλλον πρέπει να προστεθεί ένα αφαιρετικό στρώμα μνήμης, στο οποίο έχει πρόσβαση ο πυρήνας του κάθε VM και ονομάζεται ψευδο-φυσική (pseudo-physical) μνήμη. Η ιεραρχία μνήμης στο περιβάλλον Xen παρουσιάζεται στο Σχήμα [2.5]:



Σχήμα 2.5 : Τα τρία στρώματα της μνήμης στο Xen

Στο Σχήμα [2.5] παρατηρούμε ότι κάθε διεργασία έχει πρόσβαση στην εικονική της (virtual) μνήμη, ο πυρήνας του κάθε VM στην ψευδο-φυσική του (pseudo-physical) μνήμη και τέλος ο VMM έχει πρόσβαση στην φυσική μνήμη του συστήματος.

Το τριπλό, αυτό, στρώμα μνήμης θα μπορούσε να μη χρησιμοποιηθεί. Θα ήταν δυνατό να υπάρχουν όλοι οι guest πυρήνες στη φυσική μνήμη και απλά να αποτρέπονται από το να προσπελαύνουν το χώρο μνήμης των άλλων πυρήνων. Αυτό δεν υιοθετήθηκε για δύο βασικούς λόγους. Ο πρώτος είναι ότι τα περισσότερα λειτουργικά συστήματα λειτουργούν με την υπόθεση ότι έχουν έναν συνεχόμενο, ενιαίο χώρο μνήμης. Παρόλα αυτά, η φυσική μνήμη υποστηρίζει κενά ανάμεσα στις διευθύνσεις, έτσι ώστε να μπορεί να παραβλέψει κάποια προβληματικά τμήματά της. Η χρήση τέτοιων, σποραδικών, χώρων μνήμης από τα guest VMs είτε θα ήταν πολύ

αργή είτε θα χρειαζόταν σημαντικές τροποποιήσεις στον πυρήνα τους, επιπτώσεις που δεν είναι ιδανικές. Ο δεύτερος λόγος που χρησιμοποιείται αυτό το μοντέλο στρωμάτων μνήμης, σχετίζεται με τον κύκλο ζωής μιας εικονικής μηχανής. Το πιο πιθανό για μια εικονική μηχανή είναι να μην περάσει όλο τον κύκλο ζωής της στον ίδιο χώρο μνήμης. Υπάρχει περίπτωση να κατασταλεί και μετά να συνεχιστεί ξανά η λειτουργία της. Μετά την συνέχισή της, δεν είναι εγγυημένο ότι ο ίδιος χώρος στη φυσική μνήμη θα είναι διαθέσιμος, ειδικά αν η εικονική μηχανή συνεχιστεί σε διαφορετικό μηχάνημα από αυτό που είχε κατασταλεί, οπότε όλες οι διευθύνσεις φυσικής μνήμης που ήταν αποθηκευμένες από τον πυρήνα της θα πρέπει να χαρτογραφηθούν από την αρχή. [4]

Οι σελίδες μνήμης στο εικονικό περιβάλλον του Xen χωρίζονται στις εξής κατηγορίες: MFN (machine frame number), PFN (page frame number), GMFN (guest machine frame number), GPFN (guest page frame number). Η πιο γενική από όλες είναι η PFN της οποίας η σημασία διαφέρει ανάλογα με το πλαίσιο στο οποίο βρίσκεται. Ένας MFN είναι ο αριθμός μιας σελίδας στη φυσική μνήμη του μηχανήματος. Ένας GPFN είναι ο αριθμός μιας σελίδας που βρίσκεται στο χώρο μνήμης του guest VM, δηλαδή στο pseudo-physical στρώμα. Τέλος, η δυσκολότερη κατηγορία για να οριστεί είναι η GMFN, η οποία αναφέρεται είτε σε έναν MFN είτε σε έναν GPFN. Όταν διαμοιράζονται σελίδες μεταξύ των domains στο Xen, οι GPFNs του κάθε πυρήνα μεταφράζονται σε MFNs, έτσι ώστε τα domains που διαμοιράζονται στις σελίδες να έχουν πρόσβαση στα σωστά δεδομένα [4].

Μηχανισμοί επικοινωνίας στο Xen

Όπως αναφέρθηκε παραπάνω, τα VMs δεν μπορούν να είναι απομονωμένα. Χρειάζεται, πολλές φορές να επικοινωνούν μεταξύ τους για πολλούς λόγους, όπως για την πρόσβαση στο hardware. Στην ενότητα ”Αρχιτεκτονική” του Xen παρουσιάστηκε ο βασικός μηχανισμός επικοινωνίας μεταξύ των domains, το μοντέλο των split drivers. Εδώ αναλύονται κάποιοι μηχανισμοί επικοινωνίας υψηλότερου επιπέδου, οι οποίοι είναι βασισμένοι στο split driver model.

- *Μηχανισμός των Grants*:
  Ο διαμοιρασμός μνήμης (shared memory) αποτελεί τον πιο εύκολο μηχανισμό για την επικοινωνία μεταξύ των διεργασιών (interprocess communication ή IPC) ενός απλού υπολογιστή. Επειδή η μνήμη κάθε διεργασίας είναι ένα υποσύνολο ενός μεγαλύτερου, κοινού χώρου μνήμης, η μνήμη μπορεί να διαμοιραστεί απλά, δημιουργώντας

επικαλυπτόμενους χώρους διευθύνσεων. Έτσι ο Xen VMM παρέχει έναν μηχανισμό διαμοιρασμού σελίδων μνήμης και τα domains επικοινωνούν μεταξύ τους με πολιτικές βασισμένες σε αυτόν. Ονομάζεται μηχανισμός των grants και είναι χαμηλού επιπέδου, που σημαίνει ότι επιτρέπει διαμοιρασμό μνήμης σε επίπεδο σελίδων. [4] Τα grant request, τα οποία αποτελούν την αναφορά σε μια διαμοιραζόμενη σελίδα που αιτείται το απομακρυσμένο domain, αναγνωρίζονται από έναν ακέραιο που ονομάζεται grant reference (gref) και αποθηκεύεται σε μία δομή δεδομένων που ονομάζεται grant table. Κάθε domain έχει το δικό του grant table και τον διαμοιράζεται με τον Xen hypervisor. Ένα grant reference περιλαμβάνει και τις πληροφορίες της διαμοιραζόμενης σελίδας, που αντιπροσωπεύει, καθιστώντας αχρείαστη τη γνώση του mfn της σελίδας από το εκάστοτε domain.

Χρησιμοποιώντας το grant table, μπορούν να εκτελεστούν δύο λειτουργίες, το mapping μιας σελίδας ή η μεταφορά της. Οι λειτουργίες αυτές είναι εννοιολογικά παρόμοιες. Και οι δυο συνεπάγονται την εισαγωγή της φυσικής σελίδας προς ή από τον χώρο διευθύνσεων αυτού που καλεί την λειτουργία. Η διαφορά τους είναι ότι το mapping αφήνει τη σελίδα και στο χώρο διευθύνσεων του αρχικού domain, ενώ η μεταφορά αφαιρεί την αρχική αναφορά της σελίδας. Το mapping χρησιμοποιείται για τη δημιουργία διαμοιραζόμενης μνήμης, ενώ η μεταφορά για την κίνηση δεδομένων από το ένα domain σε ένα άλλο. [4][9]

Όλο αυτό το υποσύστημα επικοινωνίας μπορεί να χρησιμοποιηθεί για τη δημιουργία καταχωρητών δαχτυλιδιού (ring buffers) μεταξύ των domains. Ειδικότερα, μια διαμοιραζόμενη σελίδα μπορεί να αρχικοποιηθεί ως μια δομή δαχτυλιδιού (ring structure) και να επιτρέπει την εκπομπή ή την λήψη αιτημάτων και απαντήσεων μεταξύ των επικοινωνούντων domains. [5]

- *Κανάλια Γεγονότων (Event Channels)*

  Τα event channels είναι ο θεμελιώδης μηχανισμός για την αποστολή ασύγχρονων ειδοποιήσεων από τον Xen hypervisor στα guest domains ή μεταξύ των guests. Χρησιμοποιούνται σε συνδυασμό με τα ring buffers και παρέχουν έναν αποδοτικό μηχανισμό περάσματος μηνυμάτων για την επικοινωνία μεταξύ των frontend και backend του split device driver. [5] Το domain, που στέλνει το event, ειδοποιεί τον παραλήπτη ότι έχει τοποθετήσει ένα αίτημα ή μια απάντηση μέσα στο κοινό τους ring. Εννοιολογικά, είναι παρόμοια με τα (συμβατικά) UNIX σήματα. Κάθε event μεταφέρει ένα bit πληροφορίας, δηλαδή ότι το event έχει συμβεί. Ένα event μπορεί να αποσταλεί

την ίδια στιγμή που ένα άλλο επεξεργάζεται. Γι' αυτό είναι συχνό φαινόμενο να απενεργοποιείται η αποστολή events κατά τη διάρκεια επεξεργασίας κάποιου άλλου. Παρόλα αυτά, τα events, που λαμβάνουν χώρα ενώ η αποστολή είναι απενεργοποιημένη, δε χάνονται. Μπορεί να γίνει έλεγχος για υπάρχοντα events, ελέγχοντας την αντίστοιχη δομή του εικονικού CPU (VCPU structure).

Όταν αναφερόμαστε στα event channels, είναι σημαντικό να οριστούν δύο όροι, το "κανάλι" (channel) και η "θύρα" (port). Τεχνικά, το channel είναι η αφηρημένη έννοια της σύνδεσης μεταξύ των δύο καταληκτικών σημείων, ενώ το port είναι ένα αναγνωριστικό, που χρησιμοποιείται για την κατάδειξη του καταληκτικού σημείου στο οποίο το channel είναι συνδεδεμένο. Όταν αρχικοποιείται ένα interdomain event channel, συνδέεται σε μια θύρα στο domain που δημιουργείται. Όταν ένα απομακρυσμένο domain δεσμεύει το εν λόγω event channel, και τα δύο άκρα έχουν μία θύρα και το κανάλι μπορεί να χρησιμοποιηθεί. Από τη σκοπιά του κάθε domain, η τοπική θύρα είναι το κανάλι, καθώς δεν έχει άλλο μέσο προσδιορισμού του. Είναι σημαντικό να γνωρίζουμε τη διάκριση μεταξύ των καναλιών και των θυρών, αν και στην πράξη είναι ασφαλές να αγνοηθεί τις περισσότερες φορές. [4]

- *Xenstore*

  Το XenStore είναι ένα σύστημα αποθήκευσης, το οποίο διαμοιράζεται μεταξύ των Xen guest domains. Είναι ένα απλό ιεραρχικό σύστημα αποθήκευσης, διατηρείται σε ένα daemon, που εκτελείται στο Dom0, και είναι προσβάσιμο μέσω μιας διαμοιραζόμενης σελίδας μνήμης και ενός event channel.

  Η βασική διασύνδεση με το Xenstore αποτελείται από δύο ring buffers, ένα για κάθε κατεύθυνση. Οι αιτήσεις για την ενημέρωση του Xenstore ή για πληροφορίες σχετικά με τα τρέχοντα περιεχόμενά του τοποθετούνται σε ένα δαχτυλίδι. Οι απαντήσεις και οι ασύγχρονες ειδοποιήσεις για τυχόν αλλαγές τοποθετούνται στο άλλο δαχτυλίδι. Το πρώτο ring γράφεται από τα domU guests και διαβάζεται από το dom0, ενώ το δεύτερο γράφεται από το dom0 και διαβάζεται από τα guests. [4]

  Το XenStore αποτελείται από directories, τα οποία μπορεί να περιέχουν άλλα directories ή κλειδιά (keys). Κάθε κλειδί έχει και μια τιμή (value). Το Xenstore μπορεί επίσης να θεωρηθεί ως μια βάση δεδομένων, με δομή δέντρου, της οποίας τα φύλλα είναι ζευγάρια {κλειδί , τιμή}. [5] Η δομή του είναι πολύ παρόμοια με ένα σύστημα αρχείων. Η χρήση του είναι κάπως διαφορετική, ωστόσο. Δεν προορίζεται για την αποθήκευση ή τη

μεταφορά μεγάλου όγκου δεδομένων. Χρησιμοποιείται κυρίως ως έκτατη μέθοδο μετάδοσης μικρής ποσότητας πληροφοριών μεταξύ των domains.

Τέλος το Xenstore χρησιμοποιείται για την παροχή πληροφοριών σχετικά με τα domains που λειτουργούν, σε μία εύκολα αναγνώσιμη μορφή. Μπορεί να προσπελαστεί από τα διαχειριστικά εργαλεία (administrative tools) για την παροχή πληροφοριών σε ένα διαχειριστή (administrator), και να αποτελέσει ένα μόνιμο μέρος για να αποθηκεύουν τα domains τα δικά τους στοιχεία. [4]

Input/Output Rings

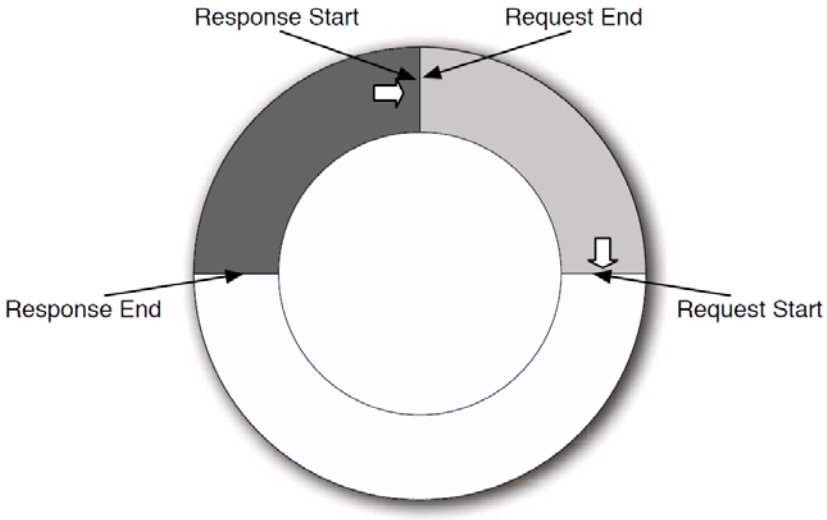
Μία από τις κύριες χρήσεις των διαμοιραζόμενων σελίδων μνήμης είναι ο ορισμός των I/O rings για την επικοινωνία μεταξύ των τμημάτων των paravirtualized οδηγών συσκευών. Αυτά παρέχουν έναν απλό, αφαιρετικό μηχανισμό ανταλλαγής μηνυμάτων (message passing) βασισμένο στην διαμοιραζόμενη μνήμη που παρέχεται από το Xen. Τα I/O rings παρέχουν μία μέθοδο για την ασύγχρονη επικοινωνία μεταξύ των domains. Ένα domain τοποθετεί ένα αίτημα (request) στο ring, ενώ ένα άλλο το αφαιρεί και εισάγει μια απάντηση (response). Επειδή τα αιτήματα και οι απαντήσεις παράγονται σε (σχεδόν) ίδιο ρυθμό, ένα ring μπορεί να φιλοξενήσει και τα δύο, ταυτόχρονα. [5]

Οι μακροεντολές που καθορίζουν τη δημιουργία και τη χρήση των ring buffers βρίσκονται στο αρχείο `/xen/include/public/io/ring.h`. Για να οριστεί ένας νέος ring buffer, χρειάζονται οι δομές (structures) που χρησιμοποιούνται για τις αιτήσεις και απαντήσεις, καθώς και ένα όνομα για τη νέα δομή του ring.

Κάθε I/O ring έχει πέντε κύρια μέρη: τους δείκτες έναρξης και λήξης για τον παραγωγό και τον καταναλωτή, καθώς και το ίδιο το ring buffer. Τα μεγέθη όλων των ring buffers είναι δύναμη του δύο. Αυτό γίνεται για βελτιστοποίηση της απόδοσης. Σημαίνει ότι οι δείκτες μπορούν απλώς να αυξάνονται, και στη συνέχεια να εφαρμοστεί σε αυτούς μια “μάσκα” (masked) για να δώσουν τη θέση εντός του ring. Εφ 'όσον τα τμήματα των παραγωγών και των καταναλωτών του ring δεν συμπίπτουν, η βελτιστοποίηση αυτή λειτουργεί χωρίς προβλήματα. Ο έλεγχος για την, τυχόν, επικάλυψη στηρίζεται σε δύο χαρακτηριστικά του συστήματος των rings. Το πρώτο είναι το γεγονός ότι το τμήμα απάντησης (response part) του ring ξεκινά πάντα στο τέλος του τμήματος αίτησης (request part). Αυτό σημαίνει ότι η μόνη δυνατή επικάλυψη προκαλείται όταν το τμήμα αίτησης πλησιάζει το τμήμα απάντησης. Αυτό μπορεί να ελεγχθεί, μέσω της απόστασης μεταξύ των δεικτών του παραγωγού αιτημάτων (request producer) και του καταναλωτή απαντήσεων

(response consumer). Αν η απόσταση αυτή είναι ίση με το μέγεθος του ring, τότε είναι γεμάτο. Ο έλεγχος, για διαθέσιμο χώρο, στο τμήμα απαντήσεων του ring είναι ευκολότερος, επειδή ένα αίτημα που απομακρύνεται από το ring μπορεί να έχει ένα αντίστοιχο αίτημα που εισάγεται σε αυτό. [4][9]

Για να γίνουν περισσότερο κατανοητά όλα αυτά, παρουσιάζεται το παρακάτω σχήμα το οποίο δείχνει τη δομή ενός I/O ring, όπου οι buffers γεμίζουν δεξιόστροφα.



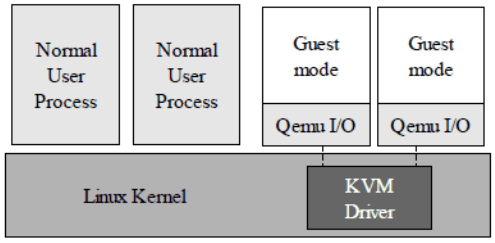
Όταν γίνεται μια αίτηση, ο δείκτης λήξης του τμήματος απαντήσεων ελέγχεται για να διαπιστώσουμε αν υπάρχει αρκετός χώρος μπροστά από την έναρξη του τμήματος αιτημάτων. Αν υπάρχει, η αίτηση γράφεται στο ring και ο δείκτης έναρξης αιτημάτων αυξάνεται. Το άλλο άκρο διαβάζει τις αιτήσεις από το πίσω μέρος, αυξάνοντας τον δείκτη αφού τελειώσει. Μετά την επεξεργασία του αιτήματος, γράφει το αποτέλεσμα στο μπροστινό μέρος του τμήματος απαντήσεων και αυξάνει το δείκτη έναρξης των απαντήσεων. Στη συνέχεια ο καλών αφαιρεί την απάντηση από τον buffer και αυξάνει το δείκτη λήξης των απαντήσεων. Να σημειωθεί ότι οι εγγραφές στο ring είναι σταθερού μεγέθους, το οποίο ορίζεται από τη μέγιστη της μεγαλύτερης αίτησης ή απάντησης που μπορεί να εισαχθεί στη δομή. [4][9]

Τα I/O rings που χρησιμοποιούνται για τη μεταφορά πολλών δεδομένων ενδέχεται να ελέγχονται συνέχεια για αλλαγές (polling) στα δύο άκρα. Αυτά που χρησιμοποιούνται πιο συχνά χρησιμοποιούν τον μηχανισμό των event channels για να ειδοποιήσουν ότι τα δεδομένα είναι διαθέσιμα. [4]

2.2.2 KVM

Ένας άλλος πολύ διαδεδομένος ελεγκτής εικονικών μηχανών είναι ο KVM (Kernel-based Virtual Machine), η χρήση του οποίου είναι η πρώτη μέθοδος virtualization που ενσωματώθηκε στον πυρήνα του λειτουργικού συστήματος Linux. Έχει αρκετές διαφορές με το Xen, οι οποίες επισημαίνονται κατά τη διάρκεια της σύντομης περιγραφής του. Η πρώτη σημαντική διαφορά είναι ότι το KVM χρησιμοποιεί τη τεχνική του full virtualization για την εκτέλεση των εικονικών μηχανών, σε αντίθεση με το Xen που χρησιμοποιεί paravirtualized guests.

Στο σύστημα του KVM, τον ρόλο του ελεγκτή εικονικής μηχανής τον παίζει το ίδιο το λειτουργικό Linux. Οι λειτουργίες, που αποτελούν το KVM, προσαρτώνται στον πυρήνα του λειτουργικού μέσω μιας λειτουργικής μονάδας (kernel module) και έτσι το λειτουργικό σύστημα Linux μετατρέπεται σε ελεγκτή εικονικής μηχανής, της οποίας η μορφή φαίνεται στο Σχήμα [2.6].



Σχήμα 2.6 : Αρχιτεκτονική του KVM

Παρατηρούμε άλλη μια βασική διαφορά μεταξύ των δύο VMMs, η οποία σχετίζεται με τον τύπο του καθενός. Αναφέραμε παραπάνω ότι το Xen είναι τύπου 1 και ειδικότερα είναι λειτουργικό σύστημα υπηρεσίας (service OS). Αντίθετα το KVM εκτελείται πάνω σε ένα ήδη εγκατεστημένο λειτουργικό σύστημα, δηλαδή είναι τύπου 2 (Host OS).

Μια ακόμη πρόσθετη λειτουργία του περιβάλλοντος του KVM, σε σχέση με το Xen και τα συμβατικά συστήματα, είναι η ύπαρξη ενός τρίτου CPU mode (οι δύο πρώτοι είναι οι user mode και kernel mode), που ονομάζεται λειτουργία φιλοξενούμενου (guest-mode) και είναι αυτή στην οποία εκτελούνται τα guest VMs.

Όσο αφορά τη μνήμη, το KVM εκμεταλλεύεται τη διαχείριση μνήμης που υπάρχει ήδη στα Linux και αποθηκεύει τις αντιστοιχίσεις εικονικών-φυσικών διευθύνσεων των εικονικών μηχανών σε ειδικές δομές που ονομάζονται shadow page tables, ένα σύστημα εντελώς

διαφορετικό από το πρόσθετο στρώμα μνήμης (pseudo-physical) που αναφέραμε για το Xen. Τέλος, το KVM υλοποιεί κάθε εικονική μηχανή ως μία διεργασία και βασίζεται στον υπάρχοντα χρονοδρομολογητή του Linux για την ανάθεση υπολογιστικής ισχύος στις εικονικές μηχανές. [6]

Κεφάλαιο 3

libvchan Evaluation

3.1 Βιβλιοθήκη libvchan : Επικοινωνία Εικονικών Μηχανών

Στο υποκεφάλαιο αυτό παρουσιάζεται αναλυτικά η βιβλιοθήκη που χρησιμοποιήθηκε στη διπλωματική, η οποία παρέχει τα απαραίτητα εργαλεία για την επικοινωνία μεταξύ των domains, σε ένα εικονικό περιβάλλον Xen. Το όνομά της είναι libvchan. Ο κώδικας της είναι γραμμένος από τους Rafal Wojtczuk και Daniel De Graaf και τα αρχεία του βρίσκονται στο directory `/tools/libvchan`. Η διεπαφή χρήστη, που παρέχει η βιβλιοθήκη αυτή, είναι βασισμένη πάνω στους μηχανισμούς επικοινωνίας του Xen, που αναφέραμε στο 2^{ο} κεφάλαιο.

3.1.1 Διαμοιρασμός μνήμης

Η libvchan βασίζεται στο μοντέλο πελάτη (client) – εξυπηρετητή (server). Το ένα domain παίζει το ρόλο του server και το άλλο το ρόλο του client. Ο server αναλαμβάνει την αρχικοποίηση όλων των μηχανισμών και των δομών, που απαιτούνται για την επικοινωνία και τη μεταφορά δεδομένων, καλώντας τη συνάρτηση `libxenvchan_server_init`. Η συνάρτηση αυτή λαμβάνει ως όρισμα το domid του domain με το οποίο πρόκειται να επικοινωνήσει ο server, τον κόμβο του Xenstore, στον οποίο θα αποθηκεύσει σημαντικές πληροφορίες της επικοινωνίας, καθώς και το μέγεθος των rings. Χρησιμοποιεί τον μηχανισμό των grants για τον διαμοιρασμό σελίδων μεταξύ των δύο domains και τα I/O rings, τα οποία αρχικοποιούνται πάνω σε αυτές τις σελίδες (shared pages), για τη μεταφορά δεδομένων. Στη συνέχεια αρχικοποιείται και ο client, με την κλήση της συνάρτησης `libxenvchan_client_init`, συνδέεται με τον server και έπειτα η επικοινωνία μπορεί να ξεκινήσει. Η συνάρτηση αυτή λαμβάνει ως όρισμα το domid του server καθώς και τον κόμβο του Xenstore από τον οποίο θα αντλήσει τις πληροφορίες της επικοινωνίας.

Δομή ελέγχου “libxenvchan”

Η libvchan χρησιμοποιεί μια συγκεκριμένη δομή για τον έλεγχο της επικοινωνίας των δύο domains, που ονομάζεται libxenvchan. Οι συναρτήσεις αρχικοποίησης του server και του client, που αναφέρθηκαν παραπάνω, αρχικοποιούν και επιστρέφουν μία τέτοια δομή, η οποία είναι απαραίτητη για τη σωστή λειτουργία όλων των υπόλοιπων συναρτήσεων της βιβλιοθήκης libvchan. Η δομή αυτή περιέχει όλα τα απαραίτητα στοιχεία που χρειάζονται τα δύο domains για να επικοινωνήσουν. Αρχικά, υπάρχουν δύο handlers για τον χειρισμό του mapping της διαμοιραζόμενης σελίδας πάνω στην οποία αρχικοποιείται το ring. Ο ένας χρησιμοποιείται από τον server και αφορά στη δημιουργία των grants και στον διαμοιρασμό τους (`*gntshr`), ενώ ο άλλος χρησιμοποιείται από τον client και αφορά στην προσπέλαση του grant table για το σωστό mapping των σελίδων (`*gnttab`). Η δομή περιέχει επίσης τα αναγνωριστικά του event channel που εγκαθίσταται μεταξύ των δύο domains, δηλαδή τον pointer του καναλιού και τον αριθμό του port του καναλιού. Στοιχεία της δομής αποτελούν και τρεις μεταβλητές που χρησιμεύουν ως flags πληροφοριών και λαμβάνουν τις τιμές 0 ή 1. Η πρώτη (`is_server`) μας πληροφορεί αν βρισκόμαστε στον server (ίση με 1) ή στον client (ίση με 0), η δεύτερη (`server_persist`) αν ο server θα παραμένει ενεργός όταν ο client κλείσει (ίση με 1), έτσι ώστε να επιτραπεί επανασύνδεση και η τρίτη (`blocking`) αν οι εργασίες θα πρέπει να μπλοκάρουν αντί να επιστρέφουν 0 (ίση με 1). Τέλος η δομή libxenvchan περιέχει δύο ακόμα δομές που αντιστοιχούν στα rings της επικοινωνίας, ένα για το read και ένα για το write των δεδομένων (`read`, `write`), καθώς επίσης και τον pointer της διαμοιραζόμενης σελίδας του ring (`*ring`).

```
struct libxenvchan
{
        union {
                xc_gntshr *gntshr; /* για τον server */
                xc_gnttab *gnttab; /* για τον client */
        };
        struct vchan_interface *ring;
        xc_evtchn *event;
        uint32_t event_port;
        int is_server:1;
```

```
        int server_persist:1;
        int blocking:1;
        struct libxenvchan_ring read, write;
}
```

Δομή των rings

Πρέπει να σημειωθεί ότι η δομή των rings, που χρησιμοποιεί η libvchan, είναι διαφορετική από την κλασική δομή που ορίζεται στο αρχείο ring.h, που αναφέρθηκε παραπάνω. Αυτό συμβαίνει επειδή οι μακροεντολές του αρχείου ring.h ορίζουν μια ασύμμετρη διαπροσωπεία σε μια δομή διαμοιραζόμενων δεδομένων που υποθέτει ότι όλα τα rings βρίσκονται σε έναν συνεχόμενο χώρο στη μνήμη. Αυτό δεν είναι κατάλληλο για τη libvchan, η οποία χρησιμοποιεί μια συμμετρική διαπροσωπεία προς το ring. Επίσης το μέγεθος των rings, που χρησιμοποιούνται στη libvchan καθορίζεται στον χρόνο εκτέλεσης (execution time), σε αντίθεση με τα rings του αρχείου ring.h των οποίων το μέγεθος καθορίζεται κατά τον χρόνο μεταγλώττισης (compile time). Συμπεραίνουμε λοιπόν ότι οι μακροεντολές του ring.h δεν μπορούν να χρησιμοποιηθούν για την πρόσβαση στα rings της libvchan. Το μέγεθος των rings, όπως αναφέρθηκε, το λαμβάνει σαν όρισμα η συνάρτηση `libxenvchan_server_init`, η οποία αφού τα αρχικοποιήσει κατάλληλα, τα αποθηκεύει στη δομή struct libxenvchan. Η νέα δομή των rings ονομάζεται `libxenvchan_ring` και περιέχει τρείς μεταβλητές. Η πρώτη αποτελεί τον pointer της διαμοιραζόμενης σελίδας (`*shr`), η δεύτερη τον καταχωρητή στον οποίο αποθηκεύονται τα δεδομένα (`*buffer`), τα οποία μπορεί είτε να γραφτούν σε αυτόν, αν αντιστοιχεί στο producer ring είτε να διαβαστούν, αν αντιστοιχεί στο consumer και η τρίτη έναν ακέραιο (`order`), ο οποίος, αν υψωθεί ως δύναμη με βάση το 2, δίνει το μέγεθος του καταχωρητή δεδομένων.

```
struct libxenvchan_ring
{
        struct ring_shared* shr;
        void* buffer;
        int order;
}
```

Δομή vchan interface

Τέλος, αναλύεται η δομή που ορίζει την διαμοιραζόμενη σελίδα του ring που συναντήσαμε στην struct libxenvchan (`*ring`). Η δομή αυτή ονομάζεται `vchan_interface` και περιλαμβάνει τα απαραίτητα στοιχεία που πρέπει να διαμοιράζονται μεταξύ των domains. Αρχικά περιέχει την βασική διαπροσωπεία καταναλωτή – παραγωγού, δηλαδή δύο μεταβλητές (`left` και `right`) οι οποίες ορίζουν ποιος θα γράφει (παράγει) και ποιος θα διαβάζει (καταναλώνει) τα δεδομένα. Μέσα στη δομή `vchan_interface` βρίσκονται και τα μεγέθη των buffers που χρησιμοποιούνται για τη μεταφορά των δεδομένων (`left_order`, `right_order`), τα οποία πρέπει να παραμένουν σταθερά καθώς η σελίδα διαμοιράζεται. Επίσης, περιέχονται κάποιες μεταβλητές που μεταφέρουν χρήσιμες πληροφορίες, όπως ο έλεγχος τερματισμού του client ή του server (`cli_live`, `srv_live` αντίστοιχα) και οι ειδοποιήσεις ότι τα δεδομένα γράφτηκαν ή διαβάστηκαν από το απομακρυσμένο domain (`cli_notify`, `srv_notify`). Τέλος η δομή αυτή περιλαμβάνει έναν πίνακα που αποτελεί τη λίστα με τα `grants`, το μέγεθος του οποίου δεν πρέπει να επεκταθεί στο μέγεθος του ring ή μετά τα όρια του χώρου μνήμης της διαμοιραζόμενης σελίδας. Ο πίνακας αυτός είναι το σημαντικότερο στοιχείο της δομής και τα στοιχεία του πρέπει να παραμείνουν σταθερά κατά την επικοινωνία των δύο domains σε περίπτωση επανεκκίνησης του client, στην οποία θα χρειαστεί εκ νέου χαρτογράφηση των `grants`.

```
struct vchan_interface
{
        struct ring_shared left, right;
        uint16_t left_order, right_order;
        uint8_t cli_live, srv_live;
        uint8_t cli_notify, srv_notify;
        uint32_t grants[0];
}
```

3.1.2 Κανάλι επικοινωνίας

Στην προηγούμενη ενότητα, αναφέρθηκε ότι ο client, μετά την αρχικοποίηση του, συνδέεται με τον server. Η σύνδεση αυτή επιτυγχάνεται με την χρήση των μηχανισμών του Xen.

Ο server, κατά την αρχικοποίηση του, εκτός από όλα τα άλλα δημιουργεί και ένα event channel μεταξύ εκείνου και του client. Τα στοιχεία που χρειάζεται, λοιπόν, ο client, για τη σωστή επικοινωνία με τον server, είναι ο pointer στην διαμοιραζόμενη σελίδα και το port του event channel που δημιουργήθηκε, τα οποία βρίσκονται στο χώρο διευθύνσεων του server. Για να γίνουν διαθέσιμα στον client, ο server τα γράφει σε έναν κόμβο του Xenstore, αφού το Xenstore είναι προσπελάσιμο από όλα τα domains. Έπειτα, ο client συνδέεται με το port που έχει δημιουργήσει ο server και το event channel έχει εγκατασταθεί σωστά μεταξύ τους. Με το event channel εγκατεστημένο και με τα δύο domains να έχουν πρόσβαση στη (-ις) διαμοιραζόμενη (-ες) σελίδα (-ες), η επικοινωνία μπορεί να ξεκινήσει.

3.1.3 Διεπαφή χρήστη

Αφού το κανάλι επικοινωνίας έχει δημιουργηθεί, τα δύο domains πρέπει να ανταλλάξουν μηνύματα ή δεδομένα. Αυτό γίνεται εφικτό με τις συναρτήσεις που ελέγχουν την είσοδο και την έξοδο των δεδομένων, οι οποίες ορίζονται στη libvchan. Η συνάρτηση που γράφει (εισάγει) τα δεδομένα στο ring buffer είναι η `libxenvchan_write`, ενώ αυτή που τα διαβάζει (εξάγει) τα δεδομένα από αυτό είναι η `libxenvchan_read`.

Όσον αφορά την εγγραφή (write) και την ανάγνωση (read) δεδομένων, η βιβλιοθήκη libvchan προσφέρει την επιλογή μπλοκαρίσματος ή όχι (blocking , non-blocking). Όταν είναι ενεργοποιημένο το blocking, για να γίνει ένα write πρέπει να έχει προηγηθεί ένα read, αλλά και το αντίστροφο. Στην περίπτωση αυτή, το domain που διαβάζει τα δεδομένα στέλνει μία ειδοποίηση, μέσω του καναλιού, ότι ολοκλήρωσε το διάβασμα και έτσι μπορούν να γραφτούν τα νέα δεδομένα. Το ίδιο κάνει και το domain που γράφει τα δεδομένα έτσι ώστε να μη διαβάζονται τα ίδια ή λανθασμένα δεδομένα. Αντίθετα, όταν είναι απενεργοποιημένο το blocking, μετά από ένα write ή ένα read το αντίστοιχο domain συνεχίζει τις “εργασίες” του και είναι αρμοδιότητα της εφαρμογής να χειριστεί τα δεδομένα κατάλληλα έτσι ώστε να μην υπάρχουν απώλειες ή λάθη.

Συναρτήσεις εγγραφής - ανάγνωσης

Παρουσιάζεται το πρωτότυπο της κάθε συνάρτησης, η περιγραφή του κάθε ορίσματός της, μια αναπαράστασή της σε φυσική γλώσσα και η αναλυτικότερη περιγραφή της:

```
int libxenvchan_write(struct libxenvchan *ctrl, const void *data,
size_t size);
```

- ctrl: Η δομή ελέγχου της επικοινωνίας
- data: Ο buffer με τα δεδομένα που θα στείλουμε
- size: Το μέγεθος του buffer

βήμα 1) Έλεγχος για blocking επικοινωνία

βήμα 2) Έλεγχος για διαθέσιμο χώρο στο ring

βήμα 3) Αντιγραφή δεδομένων από τον πίνακα στο ring (memcpy)

βήμα 4) Αποστολή ειδοποίησης για διαθέσιμα δεδομένα

βήμα 5) Αν η επικοινωνία είναι blocking περιμένει ειδοποίηση

βήμα 6) Έλεγχος για συνδεσιμότητα στο κανάλι

βήμα 7) Επιστροφή

Το πρώτο πράγμα που κάνει η `libxenvchan_write` είναι να ελέγξει αν η επικοινωνία είναι blocking ή όχι (βήμα 1). Αν είναι blocking, τότε αφού ελέγξει το διαθέσιμο χώρο στο ring (βήμα 2), γράφει σε αυτό τα δεδομένα κάνοντας memcpy από τον πίνακα data στο ring, αντιγράφοντας δηλαδή τα δεδομένα από το χώρο μνήμης που δείχνει ο pointer data στον χώρο μνήμης που δείχνει ο pointer του shared ring (βήμα 3). Στη συνέχεια στέλνει ειδοποίηση ότι τα δεδομένα είναι διαθέσιμα μέσω του shared ring (βήμα 4) και περιμένει την αντίστοιχη ειδοποίηση ότι διαβάστηκαν (βήμα 5). Τέλος ελέγχει το κανάλι επικοινωνίας για τον πιθανό τερματισμό του server ή του client (βήμα 6) και αν κάτι τέτοιο δεν έχει συμβεί, επιστρέφει (βήμα 7). Αν η επικοινωνία είναι non-blocking τότε, αφού πάλι ελέγξει το διαθέσιμο χώρο στο ring και γράψει σε αυτό τα δεδομένα με τον ίδιο τρόπο, επιστρέφει (βήμα 6). Η συνάρτηση επιστρέφει -1 αν συμβεί κάποιο λάθος, διαφορετικά τα bytes των δεδομένων που αποστέλλονται.

```
int libxenvchan_read(struct libxenvchan *ctrl, void *data, size_t
size);
```

- ctrl: Η δομή ελέγχου της επικοινωνίας
- data: Ο buffer στον οποίο αποθηκεύουμε τα δεδομένα που διαβάσαμε
- size: Το μέγεθος του buffer

βήμα 1) Έλεγχος για διαθέσιμο χώρο στο ring

βήμα 2) Αντιγραφή δεδομένων από το ring στον πίνακα (memcpy)

βήμα 3) Αποστολή ειδοποίησης για ανάγνωση δεδομένων

βήμα 4) Έλεγχος για συνδεσιμότητα στο κανάλι

βήμα 5) Έλεγχος για blocking επικοινωνία

βήμα 6) Αν η επικοινωνία είναι blocking περιμένει ειδοποίηση

βήμα 7) Επιστροφή

Η `libxenvchan_read` πρώτα ελέγχει το διαθέσιμο χώρο στο ring (βήμα 1) και έπειτα διαβάζει από αυτό τα δεδομένα κάνοντας memcpy από το ring στον πίνακα data, αντιγράφοντας δηλαδή τα δεδομένα από το χώρο μνήμης που δείχνει ο pointer του shared ring στον χώρο μνήμης που δείχνει ο pointer data (βήμα 2). Στη συνέχεια στέλνει ειδοποίηση ότι τα δεδομένα διαβάστηκαν (βήμα 3). Έπειτα ελέγχει το κανάλι επικοινωνίας για τον πιθανό τερματισμό του server ή του client (βήμα 4) και αν δεν έχει συμβεί κάτι τέτοιο προχωράει στον έλεγχο για blocking επικοινωνία (βήμα 5). Αν η επικοινωνία είναι blocking, περιμένει την αντίστοιχη ειδοποίηση, ότι έχουν γραφτεί νέα δεδομένα (βήμα 6), για να επιστρέψει (βήμα 7). Αν η επικοινωνία είναι non-blocking τότε επιστρέφει (βήμα 7) αμέσως μετά τον έλεγχο του καναλιού. Η συνάρτηση επιστρέφει -1 αν συμβεί κάποιο λάθος, διαφορετικά τα bytes των δεδομένων που διαβάστηκαν.

Drivers gntdev και gntalloc

Οι συναρτήσεις εγγραφής και ανάγνωσης δεδομένων, που αναλύθηκαν, χρησιμοποιούν δύο drivers του Xen, ο ρόλος των οποίων είναι καθοριστικός. Οι drivers αυτοί είναι οι gntdev και gntalloc και υπάρχουν για να μπορούν τα userspace προγράμματα στο Linux να δεσμεύουν μνήμη, την οποία αργότερα θα μοιραστούν με ένα άλλο domain. Χωρίς αυτές τις συσκευές, τα userspace προγράμματα στο Linux δεν μπορούν να δημιουργήσουν grant references.

Τα βασικά ioctls που χρησιμοποιεί ο οδηγός `gntalloc` είναι τα εξής:

- `gntalloc_ioctl_alloc`: Δεσμεύει μια καινούρια σελίδα και δημιουργεί ένα νέο grant reference. Για την προσθήκη του νέου gref στον grant table και τη δημοσιοποίηση του στο απομακρυσμένο domain, χρησιμοποιείται η συνάρτηση `add_grefs`.
- `gntalloc_ioctl_dealloc`: Αποδεσμεύει το grant reference, επιτρέποντας στην αντίστοιχη σελίδα να ελευθερωθεί, εφόσον δεν τη χρησιμοποιούν άλλα domains. Η συνάρτηση αυτή, αρχικά, εντοπίζει το gref προς διαγραφή, με τη χρήση της συνάρτησης `find_grefs`. Στη συνέχεια το διαγράφει από τη λίστα των grefs και ανανεώνει τους μετρητές της και τους δείκτες της.
- `ioctl_gntalloc_unmap_notify`: Εγκαθιστά έναν μηχανισμό ειδοποιήσεων πάνω στη διαμοιραζόμενη σελίδα, έτσι ώστε το απομακρυσμένο domain να μπορεί να την απελευθερώσει σε περίπτωση που το αρχικό domain τερματιστεί “βίαια”. Κάθε σελίδα υποστηρίζει μια μόνο ειδοποίηση. Όταν συμβούν πολλαπλές προσπελάσεις στην ίδια σελίδα, αυτή αποθηκεύει την ειδοποίηση της τελευταίας προσπέλασης, διαγράφοντας τις προηγούμενες.

Αντίστοιχα, τα βασικά ioctls που χρησιμοποιεί ο οδηγός `gntdev` είναι τα εξής:

- `gntdev_ioctl_map_grant_ref`: Εισάγει το grant reference στον πίνακα χαρτογραφήσεων (mapping table) του `gntdev`. Πρέπει να σημειωθεί ότι η συνάρτηση αυτή δεν πραγματοποιεί τη χαρτογράφηση (mapping) της σελίδας, η οποία αναβάλλεται μέχρι να καλεστεί η συνάρτηση `mmap()`.
- `gntdev_ioctl_unmap_grant_ref`: Διαγράφει το grant reference από τον πίνακα χαρτογραφήσεων (mapping table) του `gntdev`. Να σημειωθεί ότι πρέπει να έχει προηγηθεί η συνάρτηση αποχαρτογράφησης (`munmap()`) της εικονική διεύθυνση, στην οποία απευθύνεται το gref, διαφορετικά θα προκύψει λάθος.

- `gntdev_ioctl_get_offset_for_vaddr`: Επιστρέφει το offset στο χώρο διευθύνσεων του οδηγού που αντιστοιχεί στην εικονική διεύθυνση (`vaddr`). Αυτό μπορεί να χρησιμοποιηθεί για να εκτελέσει ένα `munmap()`, ακολουθούμενο από ένα `gntdev_ioctl_unmap_grant_ref`. Επιστρέφεται επίσης, ο αριθμός των σελίδων που δεσμεύτηκαν για τη δημιουργία της εικονικής διεύθυνσης (`vaddr`). Σε περίπτωση που περισσότερες από μία σελίδα έχει χαρτογραφηθεί σε μια συνεχόμενη περιοχή, ο δείκτης `vaddr` πρέπει να αντιστοιχεί στην αρχή της περιοχής αυτής, διαφορετικά θα προκύψει λάθος.
- `gntdev_ioctl_notify`: Εγκαθιστά έναν μηχανισμό ειδοποιήσεων πάνω στη διαμοιραζόμενη σελίδα, έτσι ώστε το απομακρυσμένο domain να μπορεί να την απελευθερώσει σε περίπτωση που το αρχικό domain τερματιστεί "βίαια". Κάθε σελίδα υποστηρίζει μια μόνο ειδοποίηση. Όταν συμβούν πολλαπλές προσπελάσεις στην ίδια σελίδα, αυτή αποθηκεύει την ειδοποίηση της τελευταίας προσπέλασης, διαγράφοντας τις προηγούμενες.

Οι drivers αυτοί λειτουργούν με τον εξής τρόπο: το ένα domain (X) κάνει grant μια σελίδα σε ένα άλλο (Y) και το δεύτερο χαρτογραφεί τη σελίδα αυτή. Ακολουθεί ο ψευδοκώδικας της διαδικασίας αυτής.

βήμα 1) Το domain X χρησιμοποιεί τη συσκευή gntalloc για να δεσμεύσει μια σελίδα της μνήμης του πυρήνα, P.

βήμα 2) Το X δημιουργεί μια καταχώρηση στον grant table που λέει ότι το Y μπορεί να έχει πρόσβαση στην P. Αυτό γίνεται χωρίς hypercall εκτός εάν ο grant table χρειάζεται επέκταση.

βήμα 3) Το X δίνει το αναγνωριστικό του grant reference, GREF, στο Y.

βήμα 4) Το Y χαρτογραφεί (maps) τη σελίδα, είτε απευθείας στη μνήμη του πυρήνα για χρήση στον backend driver, είτε μέσω της συσκευής gntdev για χαρτογράφηση μέσα στο χώρο διευθύνσεων μιας εφαρμογής που εκτελείται στο Y.

βήμα 5) Ένα πρόγραμμα στο X χαρτογραφεί στη μνήμη του (mmap) τμήμα της συσκευής gntalloc που αντιστοιχεί στην κοινόχρηστη σελίδα και μπορούν πλέον να επικοινωνούν με το Y μέσω αυτής.

Πρέπει να σημειωθεί ότι, εφόσον το Xen δεν επιτρέπει σε ένα guest VM να τερματίσει “βίαια” τη χρήση ενός grant reference, η συσκευή gntalloc μπορεί να χρησιμοποιηθεί για την κατανάλωση μνήμης του πυρήνα, αφήνοντας grant references χαρτογραφημένες από ένα άλλο domain, όταν μια εφαρμογή τερματίσει. Ως εκ τούτου, υπάρχει ένα όριο για τον αριθμό των σελίδων που μπορούν να δεσμευτούν. Όταν όλες οι αναφορές στην σελίδα αποχαρτογραφηθούν (unmapped), αυτή θα απελευθερωθεί κατά τη διάρκεια της επόμενης λειτουργίας ενός grant.

3.2 Πειραματική αποτίμηση μεταφοράς δεδομένων

Η διπλωματική αυτή ασχολείται με την μεταφορά δεδομένων μεταξύ δύο domain και το πόσο αποδοτική είναι αυτή με τη χρήση της βιβλιοθήκης libvchan. Για την αποτίμηση της επίδοσης της επικοινωνίας των domains χρησιμοποιούμε ένα προσαρμοσμένο, συνθετικό μετροπρόγραμμα. Παρουσιάζεται η δομή του μετροπρογράμματος αυτού και περιγράφεται η λειτουργία του και η χρήση της libvchan.

3.2.1 Δομή μετροπρογράμματος

Αρχικά, πρέπει να σημειωθεί ότι δημιουργούνται δύο κανάλια επικοινωνίας μεταξύ των δύο domains. Το πρώτο κανάλι είναι για την ανταλλαγή μηνυμάτων μεταξύ τους (control channel) και το δεύτερο είναι για την ανταλλαγή των δεδομένων (data channel). Το πρώτο λειτουργεί σαν ένα κανάλι γενικού ελέγχου ολόκληρης της επικοινωνίας και τα μηνύματα που ανταλλάσσονται μέσω αυτού είναι της μορφής εντολών. Υπάρχουν πέντε τέτοιες εντολές: αρχικοποίηση του server για το data channel , αρχικοποίηση του client για το data channel, τερματισμός του data channel, ειδοποίηση σε περίπτωση λάθους κατά την αρχικοποίηση του data channel client και τερματισμός της επικοινωνίας. Τόσο στο κανάλι ελέγχου όσο και στο κανάλι δεδομένων, η επικοινωνία είναι blocking. Η χρήση δύο καναλιών υιοθετήθηκε επειδή βοηθάει στην αποφυγή λαθών (αλλοίωση δεδομένων, λανθασμένος συγχρονισμός των δύο domains, λανθασμένα αποτελέσματα) που θα μπορούσαν να προκύψουν αν και ο έλεγχος και η μεταφορά δεδομένων γινόταν σε ένα κανάλι.

Οι μετρήσεις των χρόνων της επικοινωνίας γίνεται από ένα domain και συγκεκριμένα από τον control channel client. Αυτός, στην αρχή, παράλληλα με την εντολή της αρχικοποίησης του

server για το data channel, στέλνει και τα μεγέθη των δεδομένων, του ring buffer και του write granularity. Στη συνέχεια, ο server, στέλνει την εντολή αρχικοποίησης του client για το data channel και αν δεν υπάρξει κάποιο λάθος το data channel είναι έτοιμο για τη μεταφορά των δεδομένων. Έτσι, σε κάθε επανάληψη δημιουργείται ένα ξεχωριστό data channel στο οποίο γράφεται (με διαφορετικό granularity κάθε φορά) και διαβάζεται διαφορετικό μέγεθος δεδομένων με τη χρήση ring buffer, του οποίου το μέγεθος καθορίζεται κάθε φορά από τον control channel client.

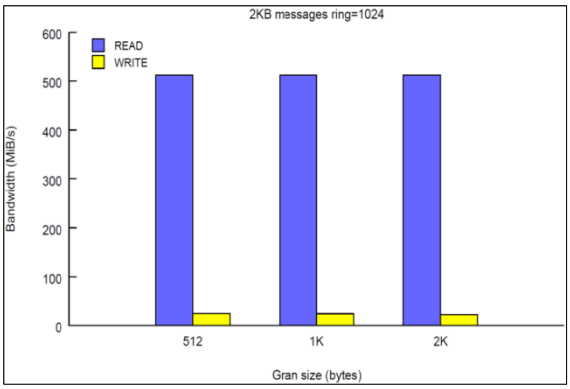
Η μεταφορά των δεδομένων, από το ένα domain στο άλλο, μπορεί να προκαλέσει κάποια απώλεια ή αλλοίωσή τους. Για το λόγο αυτό πρέπει να ελεγχθεί ότι δεν υπήρξε τέτοια περίπτωση. Ο έλεγχος αυτός πραγματοποιείται με τη μέθοδο του ping-pong. Το domain, που διαβάζει τα δεδομένα (reader), δεν μπορεί να γνωρίζει την αρχική τους κατάσταση έτσι ώστε να τα ελέγξει. Την αρχική τους κατάσταση, όμως, τη γνωρίζει το domain που τα γράφει (writer), αφού αυτό τα αρχικοποίησε. Έτσι ο reader γράφει τα δεδομένα πίσω στον writer, ο οποίος μπορεί να κάνει την επικύρωση (η μέθοδος λέγεται ping-pong επειδή τα δεδομένα μεταφέρονται από το ένα domain στο άλλο, όπως ένα μπαλάκι του ping-pong). Αυτό εξυπηρετεί και άλλο ένα ζήτημα. Ο reader είναι τώρα ικανός να μετρήσει και τον χρόνο για το read αλλά και τον χρόνο για το write, αφού γράφει τα δεδομένα πίσω στον writer, για την επικύρωσή τους. Το link που οδηγεί στον κώδικα του μετροπρογράμματος είναι το εξής: https://github.com/HPSI/libvchan-intranode

Τα αποτελέσματα παρουσιάζονται σε μορφή διαγραμμάτων στηλών. Τα διαγράμματα αυτά χωρίζονται σε δύο κατηγορίες ανάλογα με το μέγεθος που μένει σταθερό. Στη μία κατηγορία εντάσσονται αυτά που παραμένει σταθερό το μέγεθος του ring και στην άλλη κατηγορία αυτά που παραμένει σταθερό το μέγεθος του granularity.

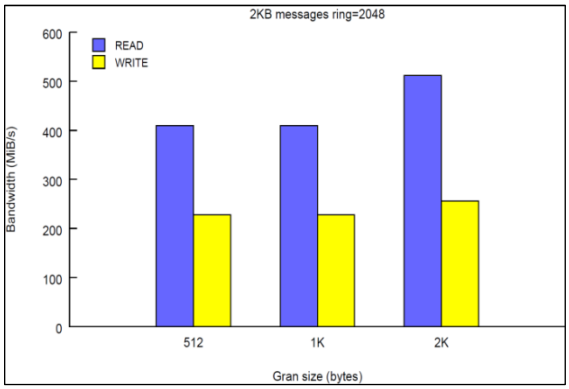
3.2.2 Επίδραση του granularity στην επίδοση

Στην υποενότητα αυτή παρουσιάζονται τα διαγράμματα με σταθερό μέγεθος ring. Το κάθε ζευγάρι στηλών (read, write) αντιστοιχεί σε μια τιμή του granularity, την επίδραση του οποίου θέλουμε να μελετήσουμε.

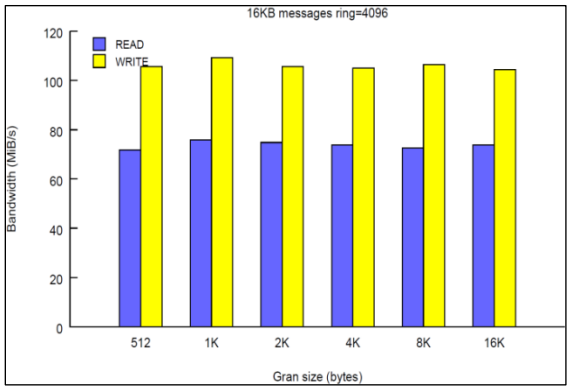
Για το κάθε μέγεθος δεδομένων (messages) που παρουσιάζεται, παρατίθενται διαγράμματα για δύο διαφορετικά μεγέθη των ring (τα οποία φαίνονται στον τίτλο του κάθε διαγράμματος).



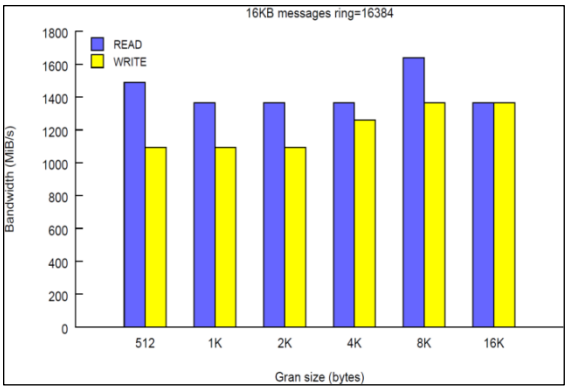
Σχήμα 3.1 Α



Σχήμα 3.1 Β

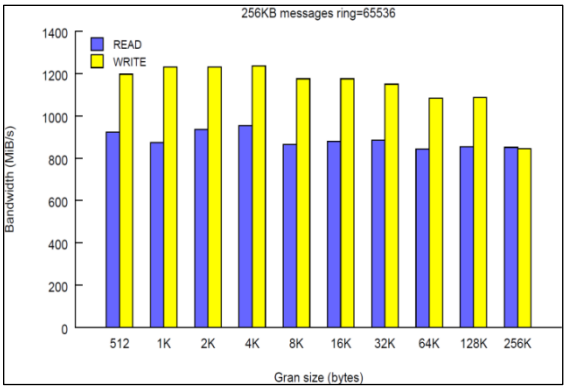


Σχήμα 3.2 Α

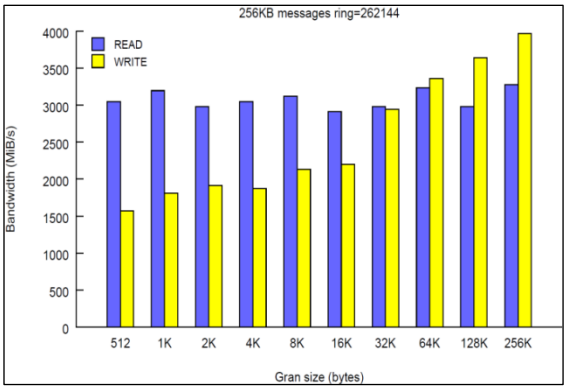


Σχήμα 3.2 Β

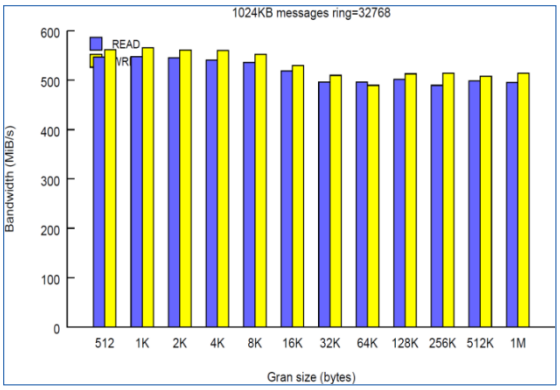
Παρατηρούμε ότι για μικρά μεγέθη δεδομένων (2KB και 16KB) το granularity (Gran size) δεν παίζει πολύ μεγάλο ρόλο στην επίδοση. Για παράδειγμα στο Σχήμα [3.1 B], το bandwidth του read είναι ίσο με 400 MiB/s για gran size 512 bytes και 1KB, ενώ ανεβαίνει στα 500 MiB/s για gran size 2KB. Αντίστοιχα, στο ίδιο Σχήμα το bandwidth του write είναι ίσο με 220 MiB/s για gran size 512 bytes και 1KB, ενώ ανεβαίνει ελάχιστα στα 250 MiB/s για gran size 2KB. Το ίδιο βλέπουμε και στα διαγράμματα των 16KB δεδομένων, όπου στο πρώτο (Σχήμα [3.2 A]) παρατηρούμε ανεπαίσθητες διαφορές στο bandwidth όσο αυξάνεται το gran size, ενώ στο δεύτερο (Σχήμα [3.2 B]) αρχίζει να φαίνεται μια μικρή επίδραση του granularity, αφού βλέπουμε διαφορές στο bandwidth μεγέθους μέχρι και 300 MiB/s.



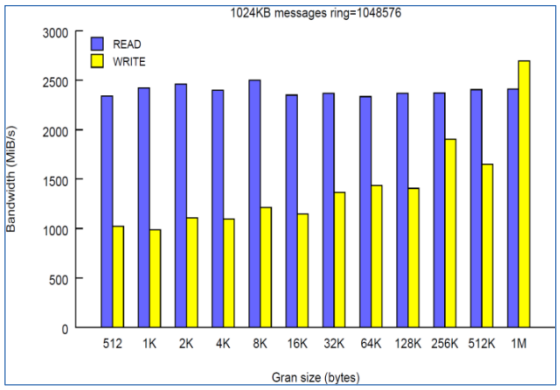
Σχήμα 3.3 Α



Σχήμα 3.3 Β

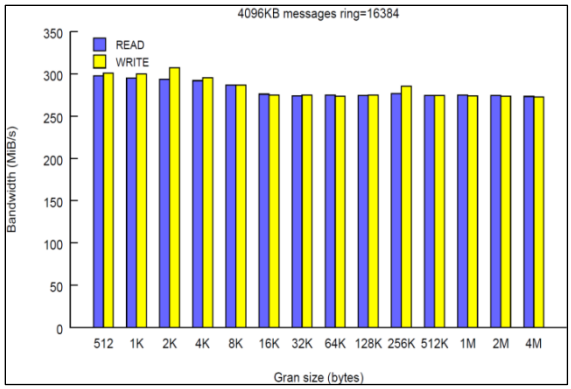


Σχήμα 3.4 Α

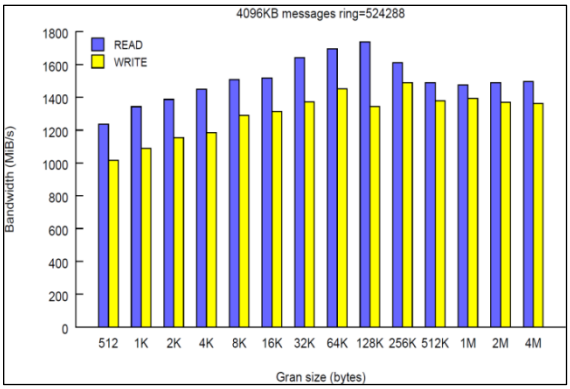


Σχήμα 3.4 Β

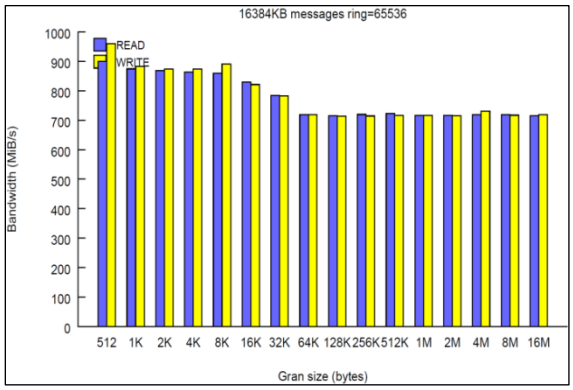
Η επίδραση του granularity ξεκινάει να γίνεται εμφανής στα μεσαία μεγέθη δεδομένων. Παρατηρώντας το πρώτο διάγραμμα για 256KB δεδομένων (Σχήμα [3.3 A]), συμπεραίνουμε ότι ούτε και σε αυτό το μέγεθος γίνεται αισθητή η επίδραση του gran size. Η εικόνα αυτή αλλάζει, όμως, στο αμέσως επόμενο διάγραμμα (Σχήμα [3.3 B]). Η διαφορά φαίνεται κυρίως στο bandwidth του write, το οποίο για 512B gran size είναι περίπου ίσο με 1500 MiB/s, ενώ στην καλύτερη περίπτωση (gran size = 256KB) είναι περίπου ίσο με 4000 MiB/s, διαφορά πολύ σημαντική στη μεταφορά των δεδομένων. Στο ίδιο διάγραμμα, το bandwidth του read είναι περίπου ίσο με 3000 MiB/s, ανεξαρτήτως του μεγέθους του granularity. Η ίδια εικόνα παρατηρείται και στα διαγράμματα του 1MB δεδομένων, στο πρώτο από τα οποία (Σχήμα [3.4 A]) έχουμε ring size ίσο με 32KB και στο δεύτερο (Σχήμα [3.4 B]) ίσο με 1MB.



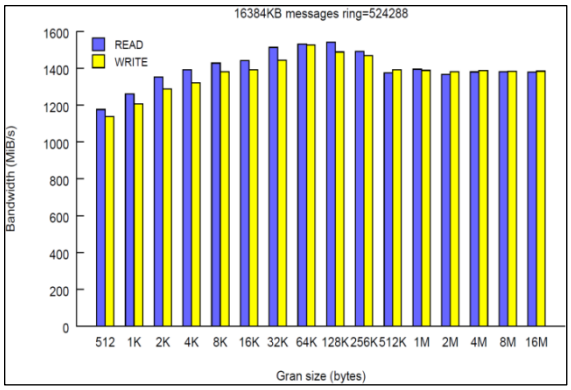
Σχήμα 3.5 Α



Σχήμα 3.5 Β



Σχήμα 3.6 Α



Σχήμα 3.6 Β

Όσο μεγαλώνει το μέγεθος των δεδομένων, η επίδραση του granularity γίνεται αισθητή και στα δύο διαγράμματα, κάτι που φαίνεται αν παρατηρήσουμε τα διαγράμματα των 16MB δεδομένων (Σχήματα [3.6 A] και [3.6 B]).

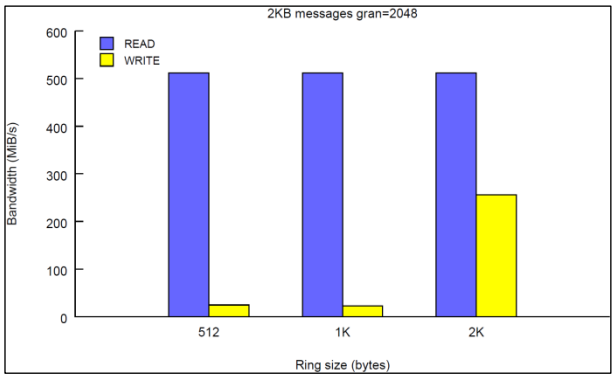
Στα διαγράμματα των μεγάλων μεγεθών δεδομένων (Σχήματα [3.5 A], [3.5 B], [3.6 A], [3.6 B]) παρατηρούμε και άλλη μια διαφοροποίηση σε σχέση με τα μικρότερα μεγέθη. Μέχρι τώρα παρατηρούμε ότι το ιδανικό μέγεθος για το granularity, δηλαδή αυτό που μας δίνει την καλύτερη επίδοση, είναι ίσο με το αντίστοιχο μέγεθος δεδομένων (για ιδανικό ring size). Αυτό φαίνεται σε μεγάλο βαθμό στο δεύτερο διάγραμμα του 1MB δεδομένων (Σχήμα [3.4 B]). Κάτι τέτοιο, όμως, δε συμβαίνει και στα τελευταία διαγράμματα. Βλέποντας τα Σχήματα [3.5 B] και [3.6 B], παρατηρούμε ότι η καλύτερη επίδοση επιτυγχάνεται με gran size ίσο με 64K, γεγονός αξιοσημείωτο, αφού θα περιμέναμε το βέλτιστο gran size να είναι ίσο με το μέγεθος του ring

(αφού το μέγιστο μέγεθος που μπορεί να εγγραφεί/αναγνωστεί κάθε φορά καθορίζεται από το μέγεθος του ring).

3.2.3 Επίδραση του ring στην επίδοση

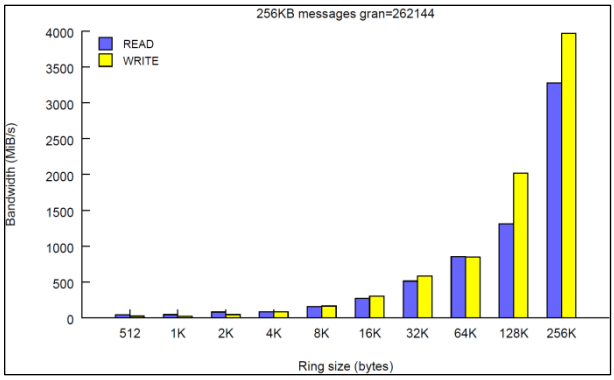
Παρατηρώντας τα δύο διαγράμματα του κάθε μεγέθους δεδομένων, της υποενότητας 3.2.2, συμπεραίνουμε ότι το μέγεθος του ring παίζει πολύ σημαντικό ρόλο στην επίδοση. Για παράδειγμα στα 4MB δεδομένων το βέλτιστο read bandwidth για μέγεθος ring ίσο με 16KB (Σχήμα [3.5 A]) είναι περίπου ίσο με 300 MiB/s, ενώ το αντίστοιχο βέλτιστο bandwidth για μέγεθος ring ίσο με 512KB (Σχήμα [3.5 B]) είναι περίπου ίσο με 1700 MiB/s!

Αυτή η παρατήρηση επιβάλλει περεταίρω μελέτη για την επίδραση του μεγέθους του ring στο bandwidth της μεταφοράς δεδομένων, η οποία φαίνεται στα επόμενα διαγράμματα, όπου το gran size παραμένει σταθερό και μεταβάλλεται το μέγεθος του ring. Επίσης, παρακάτω παρατίθεται ένα διάγραμμα για κάθε μέγεθος δεδομένων, το οποίο αντιστοιχεί στο βέλτιστο gran size που εξάγεται από τα προηγούμενα διαγράμματα.

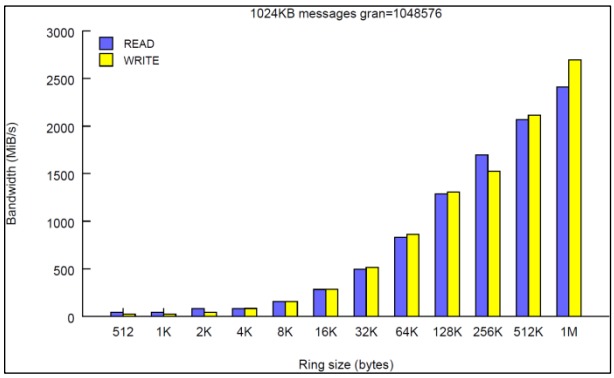


Σχήμα 3.7

Παρατηρούμε στο Σχήμα [3.7] ότι για τις δύο πρώτες τιμές του ring size οι τιμές των bandwidth του write και του read είναι ίσες. Αυτό συμβαίνει λόγω ενός περιορισμού της βιβλιοθήκης, ο οποίος αφορά το μέγεθος των ring. Όταν η συνάρτηση `libxenvchan_server_init` λάβει ως όρισμα, μέγεθος του ring μικρότερο από 2KB, τότε αρχικοποιεί αυτόματα τον read buffer του ring με μέγεθος 1KB και τον write buffer με μέγεθος 2KB. Ο περιορισμός αυτός φαίνεται και στα επόμενα διαγράμματα.



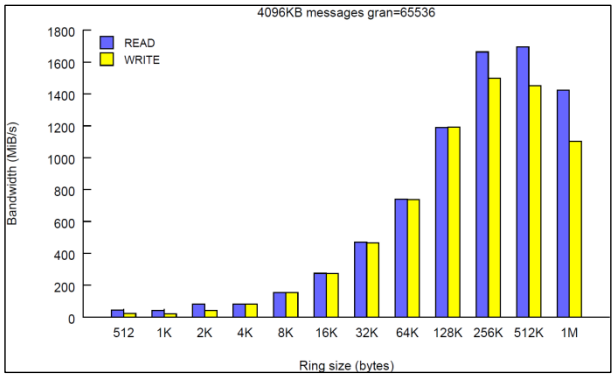
Σχήμα 3.8



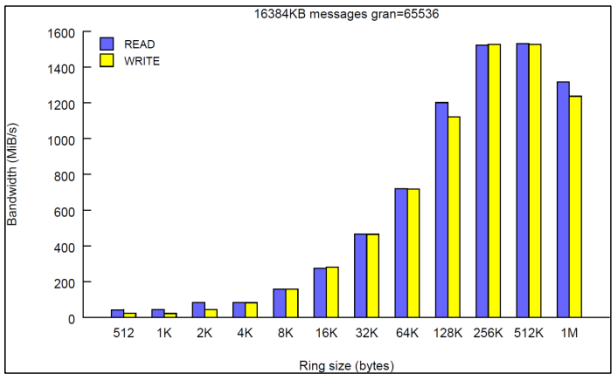
Σχήμα 3.9

Μέχρι τώρα σε όλα τα διαγράμματα παρατηρούμε ότι όσο αυξάνεται το ring size τόσο αυξάνεται και το bandwidth της επικοινωνίας, είτε αυτό αφορά το read είτε το write, το οποίο έχει τη μέγιστη τιμή του με ring size ίσο με το μέγεθος των δεδομένων.

Εδώ, πρέπει να σημειωθεί μία λεπτομέρεια για τα διαγράμματα που ακολουθούν, η οποία αφορά έναν ακόμη περιορισμό της βιβλιοθήκης. Το μέγιστο μέγεθος ring που επιτρέπει η βιβλιοθήκη είναι ίσο με 1MB και για το λόγο αυτό βλέπουμε ότι στα τρία τελευταία διαγράμματα (3.11, 3.12, 3.13) το μέγεθος του ring δε γίνεται ίσο με το μέγεθος των δεδομένων.



Σχήμα 3.10



Σχήμα 3.11

Παραπάνω αναφέρθηκε ότι όσο αυξάνεται το ring size τόσο αυξάνεται και το bandwidth της επικοινωνίας. Παρόλα αυτά, κάτι τέτοιο δε συμβαίνει στα διαγράμματα που αφορούν σε μεγέθη δεδομένων μεγαλύτερα του 1MB (Σχήματα [3.10], [3.11]). Σε αυτά βλέπουμε ότι το ιδανικό μέγεθος ring είναι ίσο με 256KB, γεγονός αξιοσημείωτο αφού θα περιμέναμε ότι όσο πιο μεγάλο είναι το ring, τόσο μεγαλύτερο θα ήταν και το bandwidth, εφόσον θα χωρούσαν περισσότερα δεδομένα μέσα στο πρώτο.

Τα διαγράμματα, της υποενότητας αυτής, αποδεικνύουν τη μεγάλη επίδραση του μεγέθους του ring στο bandwidth του read και του write των δεδομένων. Παρατηρούμε ότι σε όλα τα μεγέθη δεδομένων υπάρχουν μεγάλες διαφορές στην επίδοση ανάλογα με το μέγεθος του ring. Συμπεραίνουμε λοιπόν ότι η σωστή επιλογή του ring size, ανάλογα με το μέγεθος των δεδομένων, είναι ο σημαντικότερος παράγοντας για την αποδοτική επικοινωνία μεταξύ των domains.

Τέλος, παραθέτουμε δύο πίνακες, οι οποίοι προέρχονται από ένα πρόγραμμα που μετράει ταχύτητες μεταφοράς μνήμης σε MB/sec, για απλούς υπολογιστικούς πυρήνες γραμμένους σε C. Το πρόγραμμα αυτό ονομάζεται stream.

Function	Rate (MB/s)	Avg time	Min time	Max time
Copy	3477.7476	0.0025	0.0024	0.0028
Scale	3231.4816	0.0027	0.0026	0.0029
Add	3410.8808	0.0038	0.0037	0.0041
Triad	3561.6519	0.0036	0.0035	0.0244

Πίνακας 1: Τιμές για 4MB δεδομένων

Function	Rate (MB/s)	Avg time	Min time	Max time
Copy	3062.1068	0.0111	0.0110	0.0126
Scale	3045.4090	0.0111	0.0110	0.0126
Add	3557.2707	0.0143	0.0141	0.0151
Triad	3619.1708	0.0140	0.0139	0.0159

Πίνακας 2: Τιμές για 16MB δεδομένων

Οι παραπάνω πίνακες αποδεικνύουν ότι τα αποτελέσματα που εξάγουμε από το μετροπρόγραμμα μας και παρουσιάζονται στα διαγράμματα, συμφωνούν με τα αντίστοιχα αποτελέσματα του stream. Παρατηρώντας τα αποτελέσματα της πρώτης γραμμής των πινάκων και συγκρίνοντας τα με αυτά των σχημάτων [3.5 B], [3.6 B], [3.10], [3.11] συμπεραίνουμε ότι το bandwidth της επικοινωνίας μεταξύ των domains είναι της ίδιας τάξης μεγέθους με αυτό της αντιγραφής τους σε έναν συμβατικό πυρήνα Linux.

3.3 Βελτιστοποίηση της βιβλιοθήκης για καλύτερη επίδοση

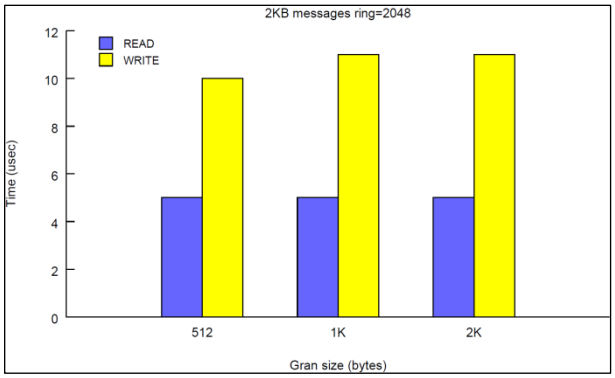
Κατά τη διάρκεια της μελέτης του κώδικα της βιβλιοθήκης libvchan, εντοπίστηκε ένα τμήμα, το οποίο είχε περιθώρια βελτιστοποίησης. Το τμήμα αυτό αφορά στο API της μεταφοράς των δεδομένων και συγκεκριμένα στη συνάρτηση `libxenvchan_write`. Στο υποκεφάλαιο 3.1.3

αναφέρθηκε ότι η συνάρτηση αυτή χρησιμοποιεί το memcpy για να αντιγράψει τα δεδομένα από τον πίνακα, που βρίσκονται αρχικά, στο ring buffer. Η αντιγραφή δεδομένων από έναν χώρο μνήμης σε έναν άλλο είναι μια χρονοβόρα διαδικασία και μάλιστα στο πρόγραμμα μας προκαλεί ένα πολύ σημαντικό overhead, αφού μπορεί να συμβεί πολλές συνεχόμενες φορές, στην περίπτωση που το ring δε χωράει όλα τα δεδομένα.

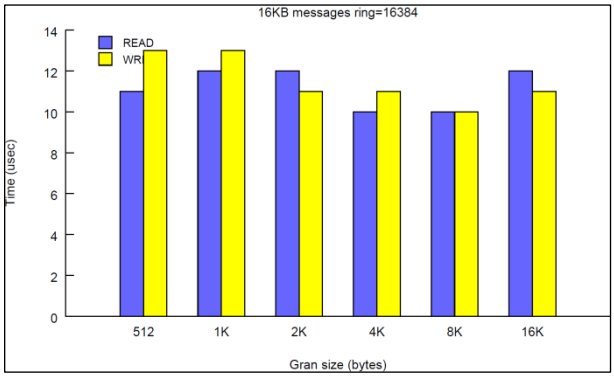
Πρέπει, λοιπόν να βρεθεί ένας τρόπος για την αποδοτική αντιγραφή των δεδομένων από τον αρχικό πίνακα στο ring επικοινωνίας, έτσι ώστε να μεταφερθούν στο απομακρυσμένο domain. Η λύση που υιοθετήθηκε είναι η εξής: δεν αρχικοποιούμε τα δεδομένα σε κάποιον πίνακα αλλά απευθείας στον χώρο μνήμης του ring επικοινωνίας, με αποτέλεσμα να μη χρειάζεται η αντιγραφή τους. Με τον τρόπο αυτό αποφεύγουμε το memcpy και έτσι παρακάμπτουμε το τρίτο βήμα του ψευδοκώδικα της `libxenvchan_write`, που παρουσιάστηκε στο υποκεφάλαιο 3.1.3. Αναλυτικότερα, ο writer (στην περίπτωσή μας συμπίπτει με τον server) αρχικοποιεί (γράφει) τα δεδομένα στον write buffer της πλευράς του server. Στη συνέχεια, ο reader (στην περίπτωσή μας συμπίπτει με τον client) διαβάζει τα δεδομένα αυτά από τον read buffer της πλευράς του client και τα αντιγράφει (memcpy) στον write buffer, της ίδιας πλευράς, έτσι ώστε να μεταφερθούν πίσω στον writer. Αυτός με τη σειρά του, τα αντιγράφει από τον read buffer, της server πλευράς, σε έναν πίνακα για την τελική επαλήθευση τους.

Η διαδικασία αυτή μας επιβάλει έναν περιορισμό, ο οποίος προέρχεται από τη σχεδίαση του μετροπρογράμματος μας. Επειδή αρχικοποιούμε τα δεδομένα μας στον χώρο διευθύνσεων του ring, αυτό πρέπει να είναι ίσου μεγέθους με το μέγεθος των δεδομένων. Αυτό, με τη σειρά του μας οδηγεί σε ακόμα έναν περιορισμό. Πρέπει να μελετήσουμε μεγέθη δεδομένων τα οποία δεν ξεπερνούν το 1MB, αφού το μέγιστο δυνατό ring size που μας επιτρέπει η βιβλιοθήκη είναι 1MB.

Τα διαγράμματα που ακολουθούν, παρουσιάζουν τα αποτελέσματα της βελτιστοποίησης αυτής. Εφόσον το μέγεθος του ring είναι σταθερό (ίσο με το μέγεθος των δεδομένων), στον οριζόντιο άξονα μεταβάλλεται το granularity. Στον κατακόρυφο άξονα έχουμε τον χρόνο της επικοινωνίας, σε αντίθεση με τα προηγούμενα διαγράμματα στα οποία είχαμε το bandwidth αυτής.

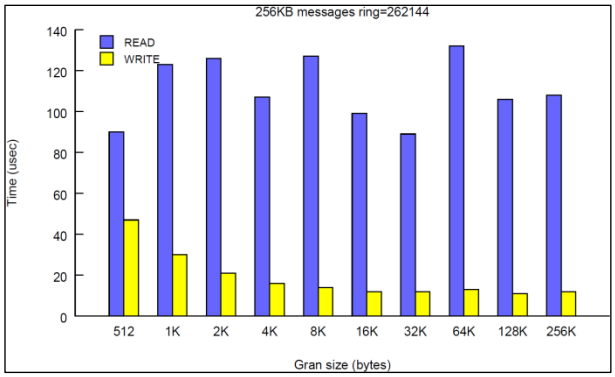


Σχήμα 3.12

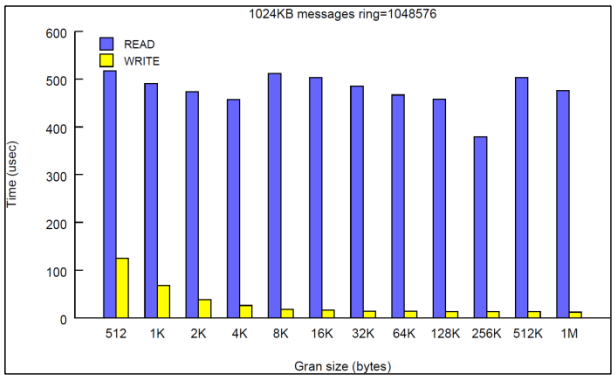


Σχήμα 3.13

Παρατηρώντας τα δύο πρώτα διαγράμματα, δε σημειώνεται καμία σημαντική διαφορά σε σχέση με τα αντίστοιχα διαγράμματα (Σχήματα [3.1 Β], [3.2 Β]) της υποενότητας 3.2.2. Για παράδειγμα στο Σχήμα [3.12] βλέπουμε ότι ο χρόνος του read είναι μικρότερος από αυτόν του write, κάτι που βρίσκεται σε αντιστοιχία με το Σχήμα [3.1 Β], όπου bandwidth του read είναι μεγαλύτερο από αυτό του write. Αυτό συμβαίνει επειδή τα μεγέθη των δεδομένων είναι μικρά, με αποτέλεσμα η αντιγραφή τους (memcpy) να μην επιβάλλει σημαντικό overhead στην επικοινωνία. Ο χρόνος που παρουσιάζεται, λοιπόν, σε αυτά τα διαγράμματα δεν αντιπροσωπεύει το memcpy των δεδομένων, αλλά το overhead που επιβάλλει η αρχικοποίηση της επικοινωνίας. Για το λόγο αυτό παρατηρούμε ότι παραμένει σχεδόν σταθερός για οποιοδήποτε gran size, αφού ουσιαστικά αποτελεί τον χρόνο που απαιτείται για τη μεταφορά των δεδομένων και όχι για την αντιγραφή τους, καθώς επίσης και το χρόνο αποστολής και λήψης των ειδοποιήσεων της blocking επικοινωνίας. Η εικόνα αυτή αλλάζει στα επόμενα διαγράμματα.



Σχήμα 3.15



Σχήμα 3.14

Στα διαγράμματα αυτά φαίνεται η διαφορά, μετά τη βελτιστοποίηση. Ο χρόνος που χρειάζεται το optimized write είναι πολλές φορές μικρότερος από τον αντίστοιχο χρόνο του read. Για παράδειγμα στο Σχήμα [3.15], αν παρατηρήσουμε τις μπάρες που αφορούν στα 64K gran size, βλέπουμε ότι το write time είναι περίπου δέκα φορές μικρότερο από το αντίστοιχο read time. Επομένως, καταφέραμε να δημιουργήσουμε μια διεπαφή, η οποία προσφέρει πολύ γρήγορη εγγραφή των δεδομένων από ένα domain σε ένα άλλο.

Κεφάλαιο 4

Επίλογος

4.1 Σύνοψη και Συμπεράσματα

Σκοπός της παρούσας διπλωματικής εργασίας ήταν η πειραματική αποτίμηση της επικοινωνίας δύο εικονικών μηχανών και η προσπάθεια βελτιστοποίησης της επικοινωνίας αυτής. Το εικονικό περιβάλλον δημιουργήθηκε από τον ελεγκτή εικονικής μηχανής Xen. Η αποτίμηση έγινε με τη χρήση ενός κατάλληλα προσαρμοσμένου μετροπρογράμματος, του οποίου τα αποτελέσματα παρουσιάστηκαν σε διαγράμματα στηλών.

Αρχικά παρουσιάστηκε στον αναγνώστη το θεωρητικό υπόβαθρο που θα έπρεπε να κατανοήσει έτσι ώστε να ακολουθήσει τη διαδικασία σχεδιασμού και υλοποίησης του συστήματος. Παρουσιάστηκε η έννοια της εικονικής μηχανής και αναφέρθηκαν εργαλεία του λειτουργικού συστήματος Linux που χρησιμοποιήθηκαν. Αναλύθηκε η αρχιτεκτονική του Xen καθώς επίσης και οι μηχανισμοί επικοινωνίας που υιοθετεί. Στη συνέχεια, παρουσιάστηκε η βιβλιοθήκη libvchan, την οποία χρησιμοποιεί το μετροπρόγραμμα. Αναλύθηκε ο τρόπος, με τον οποίο, η libvchan διαμοιράζει τη μνήμη στις δύο εικονικές μηχανές, ο μηχανισμός με τον οποίο εγκαθιστά την επικοινωνία καθώς επίσης και οι συναρτήσεις εγγραφής και ανάγνωσης των δεδομένων και οι οδηγοί του Xen, που χρησιμοποιούνται από αυτές. Ακολούθησε η περιγραφή του μετροπρογράμματος και της δομής του και η παρουσίαση των αποτελεσμάτων που εξήχθησαν από αυτό. Μελετήθηκε η επίδραση του ρυθμού εγγραφής των δεδομένων (granularity) και του μεγέθους του ring (ring size) στην επικοινωνία. Τέλος, απαλείψαμε, από τη συνάρτηση εγγραφής των δεδομένων, μια χρονοβόρα εντολή αντιγραφής των δεδομένων από μια θέση μνήμης σε μια άλλη (memcpy) και παρουσιάστηκαν τα αποτελέσματα της βελτιωμένης έκδοσης.

Από τη μελέτη των αποτελεσμάτων, συμπεραίνουμε ότι ο βασικός παράγοντας που επηρεάζει την επίδοση της επικοινωνίας είναι το ring size. Αυτό συμβαίνει επειδή το ring size καθορίζει το μέγιστο μέγεθος των δεδομένων που μπορούν να γραφτούν ή να διαβαστούν κάθε φορά. Έτσι όσο μικρότερο είναι το ring size, τόσες περισσότερες επαναλήψεις θα χρειαστούν για τη

μεταφορά όλων των δεδομένων, με αποτέλεσμα ο χρόνος της μεταφοράς αυτής να αυξάνεται. Επίσης, είναι αξιοσημείωτη η διαφορά στην επίδοση, που παρατηρείται μετά την βελτιστοποίηση της βιβλιοθήκης. Από τα διαγράμματα της τελευταίας ενότητας, συμπεραίνουμε πόσο επιβαρύνει την επικοινωνία η αντιγραφή των δεδομένων από ή προς το ring buffer.

Η χρήση της βιβλιοθήκης libvchan παρέχει στον χρήστη μια διαπροσωπεία για την ευκολότερη δημιουργία ενός καναλιού επικοινωνίας μεταξύ των εικονικών μηχανών, στο εικονικό περιβάλλον του Xen. Η μελέτη της είναι πολύ σημαντική και η βελτιστοποίηση που πραγματοποιήσαμε μπορεί να αυξήσει την επίδοση σε πολλές απαιτητικές εφαρμογές στο χώρο του HPC.

4.2 Μελλοντικές Επεκτάσεις

Από το σχεδιασμό του μετροπρογράμματος και την αναλυτική περιγραφή της μελέτης που πραγματοποιήθηκε, συμπεραίνουμε ότι η παρούσα εργασία επιδέχεται βελτιώσεις και επεκτάσεις, τις οποίες αναφέρουμε παρακάτω:

- Βελτιστοποίηση της ανάγνωσης. Στην ενότητα 3.3 παρουσιάστηκε η βελτιστοποίηση της συνάρτησης libvchan. Η βελτιστοποίηση αυτή αφορούσε μόνο την εγγραφή των δεδομένων, από την οποία αφαιρέσαμε τη διαδικασία αντιγραφής τους από τον αρχικό πίνακα προς στο ring (memcpy). Παρόλα αυτά, αντιγραφή των δεδομένων πραγματοποιείται και στην ανάγνωση των δεδομένων. Η επίδοση της επικοινωνίας μπορεί να αυξηθεί ακόμα περισσότερο αν βελτιστοποιηθεί και η ανάγνωση με τον ίδιο τρόπο. Αυτό, βέβαια συνεπάγεται ότι τα δεδομένα που διαβάζονται θα πρέπει να επεξεργαστούν πριν το επόμενο iteration της επικοινωνίας, επειδή θα υπερκαλυφθούν από τα επόμενα δεδομένα προς ανάγνωση.
- Προσθήκη TCP/IP semantics και σύγκριση με v4v. Εκτός από τη libvchan, υπάρχει και άλλη μια βιβλιοθήκη η οποία επιτρέπει στα user space προγράμματα να χρησιμοποιήσουν τους μηχανισμούς του Xen, για τη δημιουργία καναλιού επικοινωνίας μεταξύ εικονικών μηχανών. Η βιβλιοθήκη αυτή ονομάζεται v4v. Μια πιθανή επέκταση της διπλωματικής είναι η τροποποίηση του μετροπρογράμματος, έτσι ώστε να χρησιμοποιεί την v4v και η σύγκριση των αποτελεσμάτων που προκύπτουν από τις δύο βιβλιοθήκες.
- Ενσωμάτωση με πρωτόκολλα υψηλής απόδοσης και δημιουργία split driver για MPI. Η βιβλιοθήκη του Message Passing Interface επιτρέπει σε μια πολυνηματική εφαρμογή να

χρησιμοποιήσει με αποδοτικό τρόπο το hardware του συστήματος, μοιράζοντας τις επιμέρους διεργασίες της στα διαφορετικά νήματα (πυρήνες). Μια πιθανή επέκταση της παρούσας διπλωματικής θα ήταν η τροποποίηση της βιβλιοθήκης MPI, έτσι ώστε μια πολυνηματική εφαρμογή να διαμοιράζει τις επιμέρους διεργασίες της σε διαφορετικές εικονικές μηχανές, αντί για διαφορετικά νήματα. Παράλληλα θα πρέπει να δημιουργηθεί ένας split driver για το περιβάλλον του Xen, ο οποίος θα επιτρέπει την επικοινωνία των εικονικών μηχανών για την ανταλλαγή των δεδομένων που αφορούν στην πολυνηματική αυτή εφαρμογή.

Βιβλιογραφία

[1] Wikipedia, “Virtual machine,” June 2013.

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

[3] C. D. Encyclopedia, “virtual machine monitor,” 2010.

[4] David Chisnall. 2007. “The Definitive Guide to the Xen Hypervisor” (Prentice Hall Open Source Software Development Series). Prentice Hall PTR, Upper Saddle River, NJ, USA.

[5] Anastassios Nanos and Nectarios Koziris, “Xen2MX: High-performance communication in Virtualized Environments”

[6] Kivity, Avi. “KVM: the Linux Virtual Machine Monitor”. In OLS ’07: □e 2007 Ottawa Linux Symposium, pages 225–230, July 2007.

[7] Ε. Κοζύρη, “Ένταξη Σημασιολογίας Δικτύων Διασύνδεσης Υψηλής Επίδοσης σε Εικονικές Μηχανές”, 2010

[8] Wikipedia, “Virtual Memory”, June 2013

[9] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. “Xen and the art of virtualization”. In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03). ACM, New York, NY, USA, 164-177. http://doi.acm.org/10.1145/945445.945462

[10] Mark Mitchell and Alex Samuel. 2001. “Advanced Linux Programming”. New Riders Publishing, Thousand Oaks, CA, USA.

[11] Wikipedia, “Ring (computer security)”, 2013

[12] http://www.linfo.org/kernel_mode.html, “Kernel Mode Definition”, The Linux Information Project, 2004 - 2006