ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

YASMIN: Αποδοτικός μηχανισμός ενδοεπικοινωνίας εικονικών μηχανών με τη χρήση Sockets

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Μιχάλη Ροζή

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2017

Εθνικο Μετσοβιο Πολυτεχνειο
ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

YASMIN: Αποδοτικός μηχανισμός ενδοεπικοινωνίας εικονικών μηχανών με τη χρήση Sockets

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Μιχάλη Ροζή

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Ιουλίου 2017.

.....	.....	.....
Νεκτάριος Κοζύρης	Γεώργιος Γκούμας	Νικόλαος Παπασπύρου
Καθηγητής Ε.Μ.Π	Επ.Καθηγητής Ε.Μ.Π	Αν.Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2017

.....

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

Περίληψη

Σήμερα, οι εικονικές μηχανές γίνονται ολοένα και πιο διαδεδομένες και το εύρος των εφαρμογών τους περιλαμβάνει ένα μεγάλο αριθμό από ερευνητικά πεδία. Από το HPC έως τις Υποδομές-ως-Υπηρεσία (Infrastructure as a Service - IaaS), η επικοινωνία μεταξύ εικονικών μηχανών που λειτουργούν στο ίδιο φυσικό μηχάνημα είναι σημαντικός παράγοντας απόδοσης. Στην παρούσα διπλωματική, εξετάζονται τρόποι επικοινωνίας μεταξύ εικονικών μηχανών που στεγάζονται στο ίδιο φυσικό μηχάνημα, βελτιώνοντας το χρόνο επικοινωνίας χωρίς να θυσιάζεται η συμβατότητα των εφαρμογών. Παρουσιάζεται το YASMIN (*Yet Another Shared Memory Implementation for Intra-Node*), ένας συμβατός με Sockets μηχανισμός εκτέλεσης, για ενδοεπικοινωνία εικονικών μηχανών στον hypervisor Xen. Υλοποιείται πάνω στη λογική της Vchan, μιας βιβλιοθήκης του Xen για ενδοεπικοινωνία VM και χρησιμοποιεί τους μηχανισμούς του Xen για διαμοιρασμό σελίδων και αποστολή σημάτων με σκοπό τη δημιουργία ενός αποδοτικού καναλιού επικοινωνίας. Το βασικό σημείο της υλοποίησης είναι το επίπεδο μεταφοράς το οποίο βρίσκεται κάτω από το πρωτόκολλο `AF_VSOCK` και εισάγεται δυναμικά στον πυρήνα. Επιτυγχάνεται βελτίωση κατά 4.4 φορές σε σύγκριση με το κλασσικό τρόπο επικοινωνίας ως προς το ρυθμό μετάδοσης και μείωση του latency κατά 65%, χωρίς την επαναμεταγλώττιση και συγγραφή κώδικα.

Abstract

Nowadays, virtual machines are becoming widely used and their range of applications include a large number of scientific fields. From HPC to IaaS, communication between co-located VMs is a critical factor of efficiency. In our paper, we examine communication methods between VMs located in the same physical node, optimizing communication cost without sacrificing upper-layer API compatibility. We present *YASMIN (Yet Another Shared Memory Implementation for Intra-Node)*, a generic socket-compliant framework for intra-node communication in the Xen hypervisor. We build on the concept of Vchan, a Xen library for intra-node communication between different VMs and we use Xen granting and signaling mechanisms to provide an efficient communication framework. The key of our design is the transport layer which runs underneath the `AF_VSOCK` protocol family, implemented as a dynamically inserted module. We are able to achieve 4.4x higher bandwidth rate and 65% lower latency without the need of application binary recompilation.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω το Εργαστήριο Υπολογιστικών Συστημάτων που μου έδωσε τη δυνατότητα να εκπονήσω τη διπλωματική μου και ιδιαιτέρως τον διδακτορικό φοιτητή Στέφανο Γεράγγελο για την υπομονή, το χρόνο και πολύτιμη βοήθεια που μου παρείχε καθ΄ όλη τη διάρκεια εκπόνησης της παρούσας διπλωματικής.

Περιεχόμενα

1 Εισαγωγή

Η ανάπτυξη της επιστήμης και της τεχνολογίας οδήγησε στην εμφάνιση chips με ολοένα και περισσότερα τρανζίστορ και μεγαλύτερη πυκνότητα. Ο περιορισμός όμως της τάσης για κατασκευή ταχύτερων και πολυπλοκότερων μεμονωμένων υπολογιστικών πόρων [1] οδήγησε με τη σειρά του στην εμφάνιση πολυπύρηνων επεξεργαστών και παράλληλων υπολογιστικών υποδομών.

Ταυτόχρονα, έγινε δυνατή η αξιοποίηση επιταχυντών (accelerators), όπως GPUs, FPGAs, για την επεξεργασία μεγάλου όγκου δεδομένων. Η δυνατότητα επαναδιαμόρφωσης των FPGAs, σε επίπεδο διασυνδέσεων λογικών πυλών, ανάλογα με τις ανάγκες των εφαρμογών και η χαμηλή κατανάλωση ενέργειας, καθιστούν τέτοιες συσκευές ιδανικές για αξιοποίηση ως coprocessors σε διάφορες εφαρμογές, όπως φαίνεται στον Πίνακα 1. Στην ίδια κατεύθυνση, οι σύγχρονες GPUs προσφέρουν τη δυνατότητα παράλληλης εκτέλεσης απλών εντολών σε μαζικές δομές δεδομένων [2]. Σήμερα, οι συσκευές GPUs διαθέτουν χιλιάδες πυρήνες και επεξεργάζονται δεδομένα με ρυθμό της τάξης των Teraflops. Ταυτόχρονα όμως με την ανάπτυξη της χρήσης επιταχυντών σε συγκεντρωμένες υπολογιστικές υποδομές, αναπτύχθηκαν και προγραμματιστικά μοντέλα (CUDA [3], OpenCL [4]) τα οποία δίνουν τη δυνατότητα στους προγραμματιστές να τροποποιούν τον κώδικά τους και να αξιοποιούν τις *GPUs γενικού σκοπού (GPGPUs)* για τη βελτίωση της απόδοσης των εφαρμογών.

Έτσι, η ανάγκη αποδοτικότερης διαχείρισης των συγκεντρωμένων υπολογιστικών πόρων (επεξεργαστές, μνήμες, αποθηκευτικά μέσα, δικτυακά μέσα) οδήγησε με τη σειρά της στην ανάπτυξη μεθόδων διαχείρισης και εικονοποίησης (memory virtualization, binary translation), ειδικού software διαχείρισης, τεχνικών εικονοποίησης και τελικά στην εμφάνιση των σύγχρονων εικονικών μηχανών.

Οι Εικονικές Μηχανές – ΕΜ (virtual machines - *VMs*)[1] έχουν εξελιχθεί σε σημαντικό κομμάτι των σύγχρονων κέντρων δεδομένων (data center), των συστημάτων υψηλής υπολογιστικής ισχύος (High-Performance Computing - *HPC*) και των εταιρικών πλατφόρμων εφαρμογών.

Οι βασικοί λόγοι που κάνουν τις εικονικές μηχανές ένα τόσο απαραίτητο στοιχείο των σύγχρονων υπολογιστικών συστημάτων είναι η δυνατότητα εκτέλεσης "βαρέων" υπολογιστικά εφαρμογών και υπηρεσιών παρέχοντας ένα ασφαλές, απομονωμένο περιβάλλον εκτέλεσης, βελτιώνοντας το βαθμό χρησιμοποίησης του συστήματος αλλά και μειώνοντας το κόστος επικοινωνίας μεταξύ των εφαρμογών. Για παράδειγμα, μια εταιρία η οποία διαθέτει ένα μηχάνημα ως web server, ένα μηχάνημα ως βάση δεδομένων, και ένα ως εξυπηρετητή (server) εταιρικών εφαρμογών πάνω από τοπικό εταιρικό δίκτυο, μπορεί να κάνει καταλληλότερη αξιοποίηση πόρων και να βελτιώσει την ασφάλεια μεταξύ των εφαρμογών με την χρησιμοποίηση εικονικών μηχανών, είτε εκτελώντας κάθε εφαρμογή σε ξεχωριστή ΕΜ μέσα στο ίδιο φυσικό μηχάνημα, είτε αγοράζοντας εικονικές μηχανές ως υπηρεσία από πάροχο τέτοιων υπηρεσιών.

Παράλληλα, η ενεργειακή κατανάλωση των data centers αποτελεί ολοένα και μεγα-

[1]Για συντομία, για την απόδοση του όρου Εικονική Μηχανή, θα χρησιμοποιούνται συχνά οι συντομογραφίες ΕΜ και VM σε όλη την έκταση του κειμένου.

Application	Processor Only	FPGA	Acceleration Factor
Hough and Inverse Hough Processing (1)	12 minutes processing time Pentium 4-3 GHz	2 seconds of processing time at 20 MHz	370X
Spatial Statistics (Two-Point Angular Correlation Cosmology) (2)	3,397 CPU hours with 2.8-GHz Pentium (approximate solution)	36 Hours (exact solution)	96X
Black-Scholes (Financial Application (single precision FP 2M points)) (2)	2.3M experiments/sec with a 2.8-GHz Processor	299M experiments/sec	130X
Smith Waterman SS search 34 from FASTA (1)	6461 sec processing time (Opteron)	100-sec FPGA processing	64X
Prewitt Edge Detection (compute intensive video and image processing) (3)	327M clocks (1-GHz processing power)	131K clocks at 0.33 MHz	83X
Monte Carlo Radiative Heat Transfer (1)	60-ns processing time (3-GHz processor)	6.12 ns of processing time	10X
BJM Financial Analysis (5M paths) (1)	6300 sec processing time (Pentium 4-1.5-GHz)	242 sec of processing at 61 MHz	26X

Πίνακας 1: Επιτάχυνση αλγορίθμων σε FPGA. *Πηγή: [5]*

λύτερο τμήμα του συνολικού κόστους συντήρησης τους[2] [6], και έτσι, η δυνατότητα που παρέχουν τα εικονικά περιβάλλοντα για βελτίωση της χρησιμοποίησης του συστήματος, άρα μείωση του ενεργειακού κόστους[3], καθιστά τις ΕΜ απαραίτητη επιλογή για τη διαχείριση τέτοιων συστημάτων.

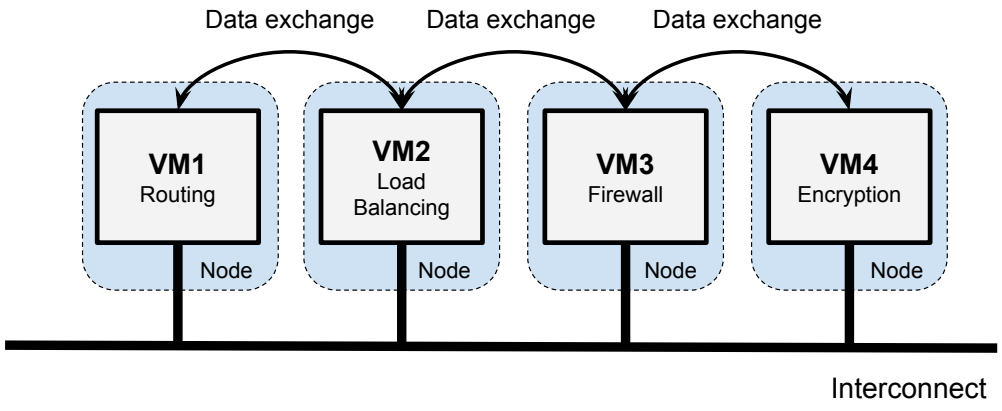
Τα πλεονεκτήματα που παρέχουν οι τεχνικές εικονοποίησης, όπως είναι φυσικό, αξιοποιούνται όχι μόνο σε συστήματα εκτέλεσης εφαρμογών αλλά επίσης και σε συστήματα διασύνδεσης δικτύων από παρόχους τηλεπικοινωνιών. Σήμερα, με τη ραγδαία ανταλλαγή δεδομένων μέσα από μεγάλες και συνεχώς μεταβαλλόμενες δικτυακές υ-

[2]Για το 2016 υπολογίζεται ότι η παγκόσμια κατανάλωση ενέργειας για τα data centers ανέρχεται σε 416,2 teraWatt το οποίο αντιστοιχεί στο 3% της παγκόσμιας παραγωγής ηλεκτρικής ενέργειας.

[3]Βέβαια το κριτήριο δεν είναι η *προστασία του περιβάλλοντος* όπως διαφημίζεται αλλά η μείωση του οικονομικού κόστους, άρα και ο ανταγωνισμός από πλεονεκτική θέση σε σχέση με άλλες εταιρίες του κλάδου.

ποδομές, τα middleboxes, δηλαδή οι συσκευές διαχείρισης δικτύων (δρομολογητές, firewalls, load balancers) έχουν γίνει συστατικό κομμάτι των μοντέρνων δικτύων τηλεπικοινωνιών. Η εμφάνιση νέων υπηρεσιών συχνά απαιτεί αναδιαμόρφωση δικτύων και εγκατάσταση νέων συσκευών το οποίο με τη σειρά του απαιτεί επιπλέον χώρο, ηλεκτρική ισχύ και έμπειρο προσωπικό. Έτσι, για να συμβαδίζουν τα σύγχρονα δίκτυα με τη ραγδαία ανάπτυξη καινοτομιών, απαιτείται μεγαλύτερη ευελιξία απ΄ όσο παρέχουν οι φυσικές διαδικτυακές συσκευές. Τέτοια δίκτυα, με συσκευές που υποστηρίζουν μεμονωμένες λειτουργίες είναι δύσχρηστα στη συντήρηση, αναπτύσσονται με αργούς ρυθμούς και δεν επιτρέπουν στους τηλεπικοινωνιακούς παρόχους να ανταπεξέρχονται στις δυναμικές αλλαγές των συστημάτων τους.

Παρόμοια με τα δυναμικά και πλήρως αυτοματοποιημένα εικονικά περιβάλλοντα τα οποία παρέχουν τεράστια οφέλη στην εκτέλεση εφαρμογών, και οι *Εικονικές Λειτουργίες Δικτύων* (Virtualized Network Functions - *VNFs*), όπως δρομολόγηση IP, firewalls, ανιχνευτές κακόβουλων επιθέσεων, load balancing κ.α, επιτρέπουν στα δίκτυα να είναι ευέλικτα και ικανά να ανταποκρίνονται με αυτοματοποιημένο τρόπο στις έντονες αλλαγές στην ανταλλαγή δεδομένων και στις υπηρεσίες που τρέχουν από "πάνω" [7][8]. Τα VNFs είναι τμήματα των σύγχρονων *υποδομών εικονοποίησης λειτουργιών δικτύων* (Network Functions Virtualisation infrastructure - *NFVi*) όπου οι ΕΜ λειτουργούν πάνω από δικτυακές συσκευές και αναλαμβάνουν την παροχή υπηρεσιών δικτύωσης [9], όπως φαίνεται και στο Σχ.1. Στην ουσία, οι τεχνικές εικονοποίησης εφαρμόζονται σε μεγάλες δικτυακές υποδομές με σκοπό τον περιορισμό του κόστους χρήσης, εγκατάστασης και διαμόρφωσης των hardware middleboxes.



Σχήμα 1: Διάφορες λειτουργίες δικτύου εκτελούνται σε εικονικά περιβάλλοντα, αντικαθιστώντας τα παραδοσιακά hardware middleboxes (routers, switches).

Πρόσθετα με τις παραπάνω εφαρμογές που υιοθετούν τεχνικές εικονοποίησης, οι ΕΜ χρησιμοποιούνται επίσης σε κατανεμημένα περιβάλλοντα εκτέλεσης, όπως το Hadoop MapReduce [10]. Πρόκειται για ένα περιβάλλον που χρησιμοποιείται κατά κόρον

σε εφαρμογές που απαιτούν επεξεργασία μεγάλου όγκου δεδομένων [11]. Οι εικονικές μηχανές γίνονται ολοένα και πιο δημοφιλή περιβάλλοντα εκτέλεσης εφαρμογών MapReduce τα οποία απαιτούν γρήγορη επικοινωνία μεταξύ παράλληλων διεργασιών. Για παράδειγμα, υπηρεσίες που βασίζονται σε υποδομές "σύννεφου" (υπηρεσίες *cloud*), όπως ο Amazon EC2, βασίζονται στις ΕΜ για να επεξεργαστούν μεγάλο όγκο δεδομένων εκκινώντας παράλληλες εργασίες υπολογισμού σε ξεχωριστά VMs.

Προκύπτει, λοιπόν, ότι τα εικονικά περιβάλλοντα εκτέλεσης είναι μια βασική κατεύθυνση στις σύγχρονες επενδύσεις υποδομής πληροφορικής και ως αποτέλεσμα, η επιστημονική έρευνα γύρω από το συγκεκριμένο κλάδο βρίσκεται σε άνθιση. Η έρευνα για τη βελτίωση της λειτουργίας των ΕΜ περιλαμβάνει ένα μεγάλο αριθμό από επιμέρους πεδία, από τη βελτίωση του χρονοπρογραμματιστή (scheduler) του hypervisor [12] έως την αποδοτικότερη διαμόρφωση των ΕΜ [13][14].

Μια από αυτές τις προκλήσεις, τόσο για εφαρμογές HPC όσο και για cloud σχετίζεται με την αποδοτικότερη επικοινωνία μεταξύ εικονικών μηχανών. Οι εικονικές μηχανές μπορούν να "στεγάζονται" στο ίδιο φυσικό μηχάνημα[4] είτε σε διαφορετικά. Η κατάλληλη τοποθέτηση ή μετανάστευσή[5] τους είναι ένας βασικός παράγοντας που καθορίζει την αποδοτικότερη επικοινωνία μεταξύ τους με μειωμένο χρόνο απόκρισης (latency) και αυξημένο throughput. Αυτό συμβαίνει διότι οι εφαρμογές HPC ή cloud μπορούν να αξιοποιήσουν την εκτέλεσή τους σε κοντινότερους κόμβους ή ακόμα και στους ίδιους κόμβους για τη μείωση της απόστασης επικοινωνίας, για την αποδοτικότερη χρήση των κοινών πόρων, ελαχιστοποίηση του χρόνου εκτέλεσης κ.α. Για παράδειγμα, μια εικονική μηχανή στην οποία εκτελείται ένα VNF δρομολόγησης πακέτων, ανταλλάσσει μεγάλο όγκο πακέτων στη μονάδα του χρόνου με μια εικονική μηχανή στην οποία εκτελείται ένα VNF εξισορρόπησης φόρτου δικτύου (load balancing). Η επικοινωνία μεταξύ των ΕΜ απαιτεί μικρό χρόνο απόκρισης και μεγάλο εύρος διαύλου (bandwidth) οπότε η κατάλληλη τοποθέτηση των ΕΜ στο ίδιο φυσικό μηχάνημα μπορεί να βελτιώσει της συνολική απόδοσης του συστήματος.

Η παρούσα διπλωματική εργασία εξετάζει μηχανισμούς επικοινωνίας μεταξύ εικονικών μηχανών που λειτουργούν μέσα στο ίδιο φυσικό μηχάνημα με σκοπό τη βελτίωση του συνολικού κόστους επικοινωνίας μεταξύ τους. Παρουσιάζεται και αναλύεται ο μηχανισμός YASMIN (*Yet-Another-Shared-Memory-Intra-Node* framework),ο οποίος αξιοποιεί τα POSIX Sockets για την παροχή αξιόπιστης ανταλλαγής δεδομένων μεταξύ ΕΜ που λειτουργούν πάνω από το ίδιο φυσικό μηχάνημα σε Xen hypervisor. Παρόλο που η συγκεκριμένος μηχανισμός υλοποιείται με τη χρήση των μηχανισμών του Xen, ο βασικός σχεδιασμός μπορεί να εφαρμοστεί και σε άλλους hypervisors. Το YASMIN χρησιμοποιεί τους μηχανισμούς *grant table* και *event-channel* του Xen με σκοπό την κοινή χρήση σελίδων μεταξύ ΕΜ του ίδιου φυσικού μηχανήματος και την ανταλλαγή σημάτων αντίστοιχα. Τελικά, δημιουργείται ένα κανάλι επικοινωνίας μεταξύ των εικονικών μηχανών, παρακάμπτοντας τόσο τη διαμεσολάβηση της ΕΜ ελέγχου

[4]Περαιτέρω ανάπτυξη του συγκεκριμένου όρου θα γίνει στην επόμενη ενότητα.

[5]Δηλαδή η μεταφορά εκτέλεσής τους από ένα κόμβο ενός cluster σε έναν άλλον.

όσο και το πρωτόκολλο TCP/IP.

2 Θεωρητικό υπόβαθρο

Στις παρακάτω υποενότητες γίνεται μια προσπάθεια να αναλυθούν βασικές έννοιες των σύγχρονων συστοιχίων υπερυπολογιστών καθώς και τα βασικά συστήματα και λειτουργίες που διέπουν τη χρήση εικονικών μηχανών. Η ανάλυση αυτή, γίνεται με σκοπό την καλύτερη κατανόηση της ανάγκης που οδήγησε στη δημιουργία και ανάπτυξη εικονικών περιβάλλοντων εκτέλεσης αλλά και στην κατανόηση των εμποδίων που προκύπτουν από τη χρήση τους, τι πρόβλημα καλείται να επιλύσει η παρούσα διπλωματική εργασία και με ποιο τρόπο το επιτυγχάνει αυτό.

2.1 High-Performance Computing

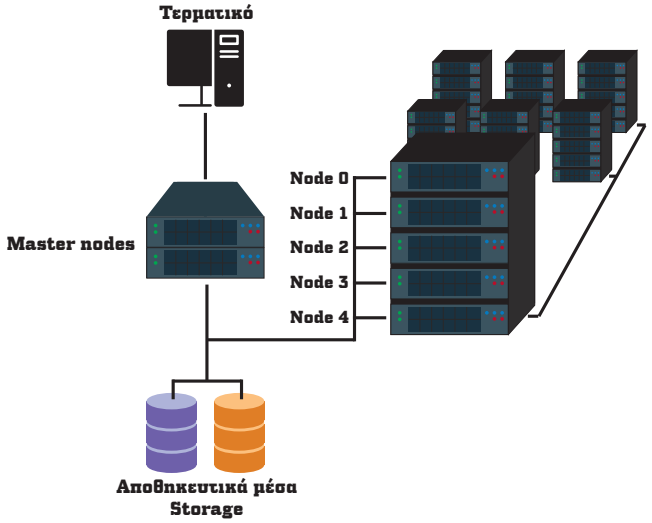
Η αντικειμενική ανάπτυξη των παραγωγικών δυνάμεων, της επιστήμης και της τεχνολογίας, η ανάγκη για αποδοτικότερη επεξεργασία όλο και μεγαλύτερου όγκου δεδομένων οδήγησαν στην εμφάνιση υπερυπολογιστών με πολυπύρηνους επεξεργαστές, επιταχυντές (GPU accelerators), μεγάλο μέγεθος μνήμης και αποθηκευτικό χώρο κλπ. Τέτοιοι υπερυπολογιστές (κόμβοι - *n*odes) διασυνδέονται μεταξύ τους και σχηματίζουν μια ομάδα κόμβων (*cluster*). Τα σύγχρονα συστήματα HPC περιλαμβάνουν εκατοντάδες χιλιάδες πυρήνες και καταναλώνουν ισχύ της τάξης των MWatt [15]. Στο Σχήμα 2 απεικονίζεται η αρχιτεκτονική των σύγχρονων υποδομών HPC.

Η ολοένα και αυξανόμενη ανάγκη για αποδοτικά υπολογιστικά συστήματα και τα πλεονεκτήματα που προσφέρει η συγκέντρωση υπολογιστικών πόρων σε clusters οδήγησε στην εμφάνιση ενός νέου ερευνητικού πεδίου, του High-Performance Computing (HPC). Ένα σημαντικό και συνεχώς αυξανόμενο πλήθος εφαρμογών εντάσσονται στον τομέα του *HPC* διότι παρέχει τη δυνατότητα αξιοποίησης συστημάτων υψηλής υπολογιστικής ισχύος. Έτσι ένα πλήθος από κερδοφόρους κλάδους της οικονομίας [16] εκμεταλλεύονται τέτοιες υπολογιστικές δυνατότητες και επενδύουν στην έρευνα γύρω από τομείς όπως:

- Κέντρα Δεδομένων (data centers) τα οποία αναλαμβάνουν την καταγραφή, αποθήκευση και επεξεργασία τεράστιου όγκου δεδομένων πελατών, χρηστών ή άλλων πολιτών[17], χρηματιστηριακές προβλέψεις και επενδύσεις μετοχικών εταιρειών[18].

- Κρυπτογραφία, για την ασφαλή αποθήκευση και μετάδοση ευαίσθητων δεδομένων.

- Προσομοιώσεις και μοντελοποίηση σε θέματα όπως καταναλωτικά μοτίβα και συμπεριφορές, αεροδυναμική, υδροδυναμική, πυρηνικές δοκιμές, κοσμολογία, μετεωρολογικές προβλέψεις, επίλυση σύνθετων προβλημάτων κβαντικής μηχανικής. Για παράδειγμα, η προσομοίωση με τη χρήση HPC αξιοποιείται από τις αεροναυπηγικές και αυτοκινητιστικές βιομηχανίες για μείωση του χρόνου παραγωγής νέων μοντέλων και του κόστους δοκιμής και ελέγχου των συστημάτων [19].



Σχήμα 2: Βασική αρχιτεκτονική ενός HPC cluster. Οι κόμβοι διασυνδέονται μεταξύ τους αλλά και με τα άλλα στοιχεία της υποδομής (GPUs, network switches, master nodes) με δίκτυα υψηλής ταχύτητας.

- Μηχανική υλικών για την βελτίωση και αξιοποίηση καταλληλότερων και φθηνότερων πρώτων υλών (μοντελοποίηση πολυμερών, κρυστάλλων, μοριακή μοντελοποίηση, μοντελοποίηση χημικών ενώσεων).

- Υποδομές-ως-υπηρεσίες (Infrastructure As A Service - *IaaS*) δηλαδή την παροχή hardware ως υπηρεσία σε εταιρικές εφαρμογές.

Τα οφέλη από την αξιοποίηση υποδομών HPC στην παραγωγή είναι τόσο μεγάλα που οι μεγάλοι μονοπωλιακοί όμιλοι, *μέσω των κρατών των χωρών τους*, ανταγωνίζονται ως προς την επίτευξη του *exascale* HPC, δηλαδή την υλοποίηση υποδομών με δυνατότητες εκτέλεσης πάνω από 10^{18} εντολών το δευτερόλεπτο. Για παράδειγμα, ένα ειδικό συμβούλιο έχει συσταθεί στις ΗΠΑ [20] το οποίο επιδιώκει την εγκαθίδρυση των ΗΠΑ ως πρώτη δύναμη παγκοσμίως στην παραγωγή και κατοχή υπερυπολογιστών και εξετάζει τα οφέλη στην *ανταγωνιστικότητα* των αμερικάνικων μονοπωλίων στη διεθνή αγορά, στην ενίσχυση της αμυντικής βιομηχανίας και στο σχεδιασμό σύγχρονων οπλικών συστημάτων [21]. Στην παγκόσμια κατάταξη των υπερυπολογιστών, τις πρώτες δυο θέσεις κατέχει η Κίνα, με δυνατότητα εκτέλεσης μέχρι 12,54 PFlops/s (1PFlop/s = 10^{15} εντολών κινητής υποδιαστολής ανά δευτερόλεπτο) και ακολουθούν η ΗΠΑ, η Ιαπωνία, Ελβετία κ.λ.π.

Η αξιοποίηση του τομέα του HPC σε κοινωνικά αναγκαίους αλλά λιγότερο κερδοφόρους τομείς (αντισεισμική πρόβλεψη, μελέτη και προσομοίωση, έρευνα στη σύγχρονη ιατρική και φαρμακολογία, συμβολή στη μείωση του εργάσιμου χρόνου) μπορεί, με τις αλματώδεις κατακτήσεις αυτού του κλάδου σήμερα, να συμβάλει καθοριστικά στην αποτελεσματικότερη ενίσχυση της συνολικής κοινωνικής παραγωγής για την ικανο-

ποίηση των αναγκών της κοινωνίας[6]. Οι σύγχρονες υποδομές HPC παρέχουν στην κοινωνία τη δυνατότητα για ολοκληρωμένη καταγραφή των σύγχρονων κοινωνικών αναγκών και παραγωγικών δυνατοτήτων ώστε να εκπονείται συγκεκριμένο σχέδιο παραγωγής. Αυτό το σχέδιο παραγωγής θα συνδυάζει τις δυνατότητες και τις ανάγκες της κοινωνίας, εξαλείφοντας την άναρχη και ανταγωνιστική παραγωγή, η οποία οδηγεί αντικειμενικά είτε στην υποπαραγωγή είτε στην υπερπαραγωγή, στη συνεχόμενη σπατάλη παραγωγικών δυνάμεων (ανεργία, απούλητα εμπορεύματα, πατεντάρισμα κοινωνικής γνώσης κ.α) είτε τελικά στις πιο ακραίες μορφές όπως τις περιοδικές κρίσεις υπερπαραγωγής [22] (π.χ καπιταλιστική κρίση 2008)

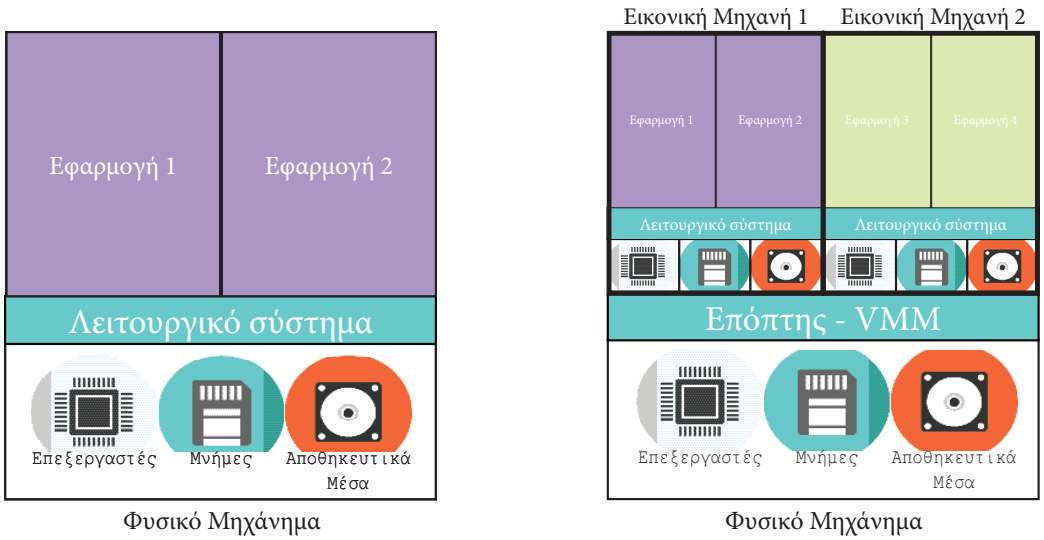
Απαραίτητη προϋπόθεση για την εφαρμογή αυτού του σχεδιασμένου τρόπου παραγωγής και την πλήρη αξιοποίηση των δυνατοτήτων του HPC αλλά και γενικά της επιστήμης, είναι η απαλλαγή της κοινωνίας από τις δυνάμεις που συγκρατούν την εξέλιξη, δηλαδή τους ιδιοκτήτες των συγκεντρωμένων παραγωγικών μονάδων.

2.2 Εικονικές Μηχανές - Virtual Machines

Η εμφάνιση των συστημάτων HPC καθώς και η ανάγκη για αποδοτικότερη διαχείριση των υπολογιστικών πόρων είχαν σαν αποτέλεσμα την ανάπτυξη των εικονικών μηχανών (Virtual Machines). Μία Εικονική Μηχανή (ΕΜ) είναι στην ουσία ένας εξομοιωτής ενός υπολογιστικού συστήματος. Οι ΕΜ βασίζονται στην αρχιτεκτονική υπολογιστών και παρέχουν λειτουργικότητα ενός φυσικού μηχανήματος. Δομική μονάδα των εικονικών μηχανών είναι ο Virtual Machine Monitor (VMM) ή hypervisor[7]. Πρόκειται για ένα στρώμα λογισμικού το οποίο αναλαμβάνει την δημιουργία και διαχείριση των VMs, και είναι υπεύθυνο για την ανταλλαγή εντολών μεταξύ των εφαρμογών και των φυσικών υπολογιστικών πόρων. Οι εφαρμογές παραδίδουν τις εντολές προς εκτέλεση στην ΕΜ, η οποία τις παραδίδει στον hypervisor και αυτός συνεργάζεται με το φυσικό μηχάνημα για την εκτέλεσή τους. Στη συνέχεια λαμβάνει τα αποτελέσματα των εντολών τα οποία τα παραδίδει πίσω στην εικονική μηχανή. Ένας υπολογιστής πάνω στον οποίο λειτουργεί ο hypervisor λέγεται μηχάνημα οικοδεσπότης (host machine) ενώ κάθε εικονική μηχανή που διαχειρίζεται ο hypervisor λέγεται μηχάνημα φιλοξενούμενος (guest machine). Στο Σχ. 3 δίνεται η λειτουργία των ΕΜ σε συνεργασία με τον hypervisor συγκριτικά με ένα φυσικό μηχάνημα το οποία το διαχειρίζεται το Λειτουργικό Σύστημα. Οι hypervisors διακρίνονται σε δυο είδη, τους Τύπου–1 ή ιθαγενείς (native) ή μετάλλου (bare-metal) καθώς και τους Τύπου–2 ή στεγασμένους (hosted). Στην πρώτη κατηγορία ανήκουν αυτοί οι οποίοι λειτουργούν ακριβώς πάνω από τους φυσικούς υπολογιστικούς πόρους και ελέγχουν το υλικό του φυσικού μηχανήματος. Στη δεύτερη κατηγορία ανήκουν οι hypervisors που λειτουργούν μέσα σε

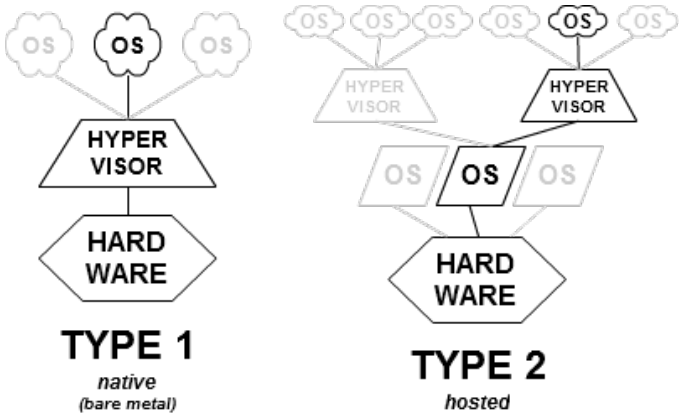
[6]Οι οποίες αν δεν βρίσκονται στην κατεύθυνση του μέγιστου κέρδους των επιχειρήσεων που καθορίζουν την παραγωγή και την έρευνα, τότε παραμελούνται.

[7]Ο όρος προκύπτει σαν εξέλιξη της λέξης supervisor ή επόπτης, η οποία χρησιμοποιείται για να περιγράψει ένα Λειτουργικό Σύστημα. Στα ελληνικά ο όρος hypervisor αποδίδεται και αυτός ως Επόπτης, αλλά επειδή δεν αποδίδει πλήρως το περιεχόμενο, θα χρησιμοποιηθεί ο αγγλικός όρος αντ΄ αυτού.



Σχήμα 3: Αριστερά φαίνεται η λειτουργία δυο εφαρμογών μέσα σε ένα φυσικό μηχάνημα ενώ δεξιά φαίνεται η λειτουργία εφαρμογών μέσα σε δυο εικονικές μηχανές οι οποίες συνεργάζονται με τον hypervisor για την επικοινωνία με το φυσικό μηχάνημα.

ένα τυπικό Λειτουργικό Σύστημα, όπως λειτουργούν οι συνήθεις εφαρμογές. Κάθε φιλοξενούμενη εικονική μηχανή λειτουργεί τότε ως μια ξεχωριστή διεργασία. Στο Σχ. 4 φαίνεται σχηματικά η διάκριση των hypervisors στις δυο κατηγορίες. Παραδείγματα τύπου-1 hypervisor είναι ο Xen και ο KVM ενώ παραδείγματα τύπου-2 είναι ο QEMU και VirtualBox.



Σχήμα 4: Hypervisor τύπου–1 και τύπου–2. Πηγή: Wikipedia [23]

2.3 Πυρήνας Λειτουργικού Συστήματος

Ο πυρήνας ενός ΛΣ είναι στην ουσία ο διαχειριστής ολόκληρου του υπολογιστικού συστήματος. Είναι το πρώτο πρόγραμμα που τρέχει κατά τη διάρκεια εκκίνησης ενός Η/Υ και είναι υπεύθυνο για το έλεγχο και το συντονισμό όλων των λειτουργιών του συστήματος. Είναι το μοναδικό πρόγραμμα που είναι εξουσιοδοτημένο να εκτελεί το σύνολο των εντολών του επεξεργαστή και να έχει πρόσβαση στο υλικό του, όπως μνήμες, αποθηκευτικά μέσα, κάρτες δικτύων κλπ. Οι εφαρμογές δεν έχουν άμεση πρόσβαση σε λειτουργίες εισόδου/εξόδου, εγγραφής στη μνήμη, χρήσης του δικτύου κ.α, αλλά αποστέλλουν ειδικά αιτήματα προς τον πυρήνα του ΛΣ, τις κλήσεις συστήματος (system calls), ο οποίος αναλαμβάνει να τα φέρει εις πέρας. Με αυτό τον τρόπο εξασφαλίζεται ότι οι εφαρμογές δεν θα έχουν πλήρη πρόσβαση στο υλικό του υπολογιστή και έτσι ενισχύεται η ασφάλεια των εφαρμογών και δημιουργείται ένα απομονωμένο περιβάλλον εκτέλεσης.

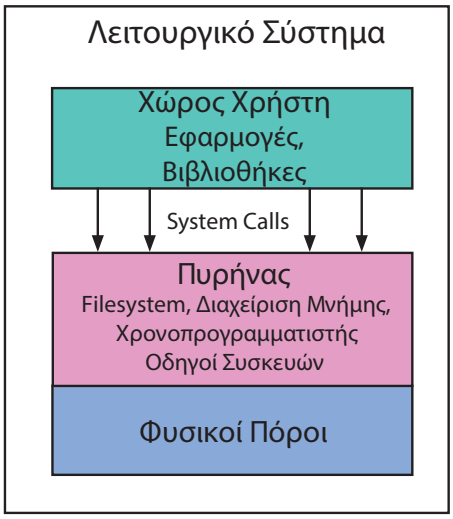
Βασικό κομμάτι του πυρήνα των Λειτουργικών Συστημάτων είναι οι οδηγοί συσκευών (device drivers). Κάθε εφαρμογή που θέλει να χρησιμοποιήσει μια συσκευή του συστήματος, στέλνει αιτήματα προς τον οδηγό της, ο οποίος εκτελείται στον πυρήνα του ΛΣ. Έτσι για παράδειγμα, μια εφαρμογή που θέλει να αποθηκεύσει ένα αρχείο στο σκληρό δίσκο του μηχανήματος, προετοιμάζει μια κλήση συστήματος προς τον οδηγό του δίσκου, ο οποίος αναλαμβάνει να πάρει τα δεδομένα από την κατάλληλη περιοχή μνήμης της εφαρμογής και να τα αποθηκεύσει στο δίσκο.

Ο βασικός πυρήνας του ΛΣ συνήθως αποτελείται από ένα ενιαίο κώδικα, μεταγλωττισμένο και συνδεδεμένο σε ένα συμπαγές εκτελέσιμο αρχείο το οποίο φορτώνεται στη μνήμη κατά την εκκίνηση. Όμως, για να μπορεί ένα λειτουργικό σύστημα να παρέχει νέες λειτουργίες και να αφαιρεί απαρχαιωμένες, να προσαρμόζεται στις ανάγκες του χρήστη, διαθέτει τα modules πυρήνα, ειδικά μεταγλωττισμένα τμήματα κώδικα τα οποία εισάγονται και αφαιρούνται δυναμικά από το χρήστη στον πυρήνα του συστήματος. Στο Σχ. 5 δίνεται σχηματικά η δομή ενός Λειτουργικού Συστήματος.

2.4 Σελιδοποίηση - Paging

Ένα ΛΣ και οι εφαρμογές που εκτελούνται έχουν τη δυνατότητα να προσπελάσουν μνήμη μεγαλύτερη από το μέγεθος της φυσικής μνήμης του συστήματος. Αυτό επιτυγχάνεται με της χρήση *εικονικής μνήμης*. Στην ουσία χρησιμοποιούνται άλλα αποθηκευτικά μέσα (σκληροί δίσκοι) για την εξομοίωση της επιπλέον μνήμης του υπολογιστή. Για παράδειγμα ένα σύστημα με δίαυλο 32bit μπορεί να προσπελάσει 2^{32} διευθύνσεις = 4 GB μνήμης. Αν το φυσικό μηχάνημα διαθέτει μόνο 1 GB μνήμης RAM, ο υπόλοιπος χώρος παρέχεται από το σκληρό δίσκο.

Το σύνολο της εικονικής μνήμης ενός συστήματος χωρίζεται σε σελίδες (pages). Πρόκειται για κομμάτια συνεχόμενης μνήμης σταθερού μεγέθους τα οποία αντιστοιχίζονται σε μια μοναδική εγγραφή στον πίνακα σελίδων του λειτουργικού συστήματος. Το μέγεθος των σελίδων αλλάζει από επεξεργαστή σε επεξεργαστή. Για παράδειγμα, για έναν x86 επεξεργαστή κάθε σελίδα έχει μέγεθος 4kB (2^{12}) άρα ο πίνακας σελίδων περιλαμβάνει 2^{20} εγγραφές.



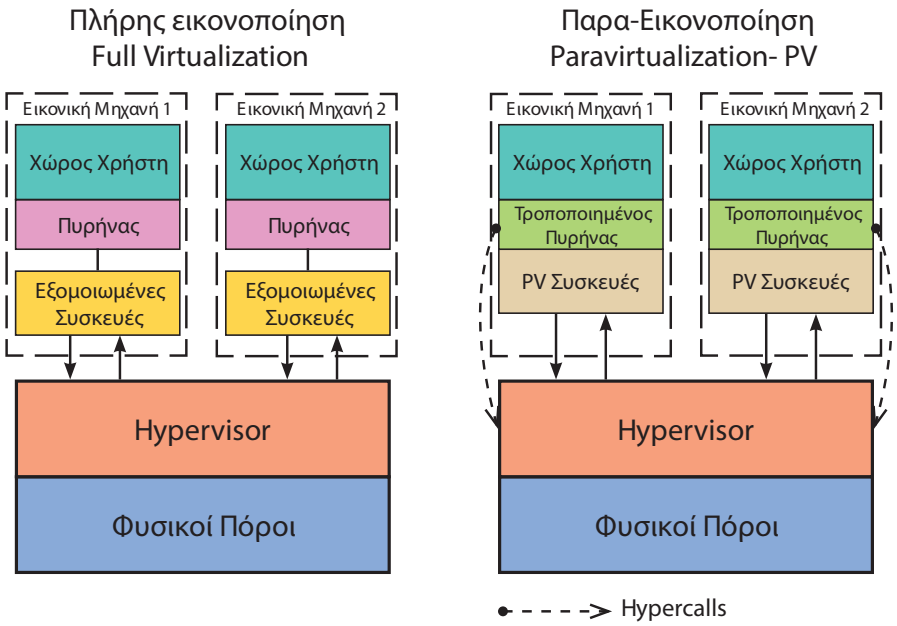
Σχήμα 5: Η βασική δομή ενός λειτουργικού συστήματος.

Κάθε εφαρμογή που εκτελείται, αποτελείται από ένα πλήθος σελίδων οι οποίες δεν φορτώνονται όλες (πρόγραμμα και δεδομένα) στη μνήμη του συστήματος. Όταν γίνεται αίτηση για πρόσβαση σε μια συγκεκριμένη σελίδα της εφαρμογής, ένα υποσύστημα είναι υπεύθυνο να ελέγχει αν η συγκεκριμένη σελίδα είναι φορτωμένη στη μνήμη RAM. Αν όχι, τότε αντικαθιστά μια σελίδα εφαρμογής που δεν εκτελείται τη δεδομένη χρονική στιγμή από τον επεξεργαστή με τη ζητούμενη σελίδα που βρίσκεται στο σκληρό δίσκο. Με αυτό τον τρόπο, οι εφαρμογές μπορούν να έχουν πρόσβαση σε μεγαλύτερο χώρο μνήμης απ΄ όσο παρέχει το φυσικό σύστημα.

2.5 Xen Hypervisor (Επόπτης Xen)

Ο Xen hypervisor[24], όπως αναφέρθηκε, είναι hypervisor τύπου–1, δηλαδή λειτουργεί ακριβώς πάνω από το υλικό του φυσικού μηχανήματος και μπορεί να δημιουργεί, εκκινεί και διαχειρίζεται παράλληλα εικονικές μηχανές. Κυρίαρχο ρόλο στη διαχείριση των εικονικών μηχανών παίζει η εικονική μηχανή ελέγχου, το domain 0. Πρόκειται για μια ειδική ΕΜ, εξουσιοδοτημένη να συνεργάζεται με τις υπόλοιπες ΕΜ, να συντονίζει και να ελέγχει την λειτουργία τους. Το domain 0 είναι στην ουσία υπεύθυνο για τη σωστή επικοινωνία των ΕΜ με τους φυσικούς πόρους.

Το Xen χρησιμοποιεί δύο τεχνικές για τη δημιουργία και διαχείριση του περιβάλλοντος εκτέλεσης των εικονικών μηχανών, την *εικονοποίηση βοηθούμενη από υλικό* (Hardware-assisted Virtualization - HVM) ή *πλήρη εικονοποίηση* και την *παραεικονοποίηση* (paravirtualization - PV). Στο Σχ. 6 απεικονίζονται συνοπτικά οι διαφορές ανάμεσα στα δυο είδη εικονικών μηχανών του Xen hypervisor.



Σχήμα 6: Σύγκριση των τεχνικών HVM και Paravirtualization. Η τεχνική PV αξιοποιεί τις ειδικές εντολές hypercalls προς τον hypervisor.

2.5.1 Τεχνική HVM

Η τεχνική HVM είναι μια μέθοδος *πλήρους εικονοποίησης (Full Virtualization)* εικονικών μηχανών. Ο hypervisor χρησιμοποιεί τις επεκτάσεις εικονοποίησης που παρέχουν οι σύγχρονοι επεξεργαστές (Intel VT, AMD-V) για να διαχειριστεί αποδοτικά τις ΕΜ. Ο Xen αξιοποιεί τον QEMU [25] ώστε να παρέχει στις ΕΜ ένα σύνολο από πλήρως εξομοιωμένες συσκευές, όπως BIOS, συσκευές δικτύου, συσκευές αποθήκευσης, USB κ.α. Επιπρόσθετα, δεν απαιτείται καμία αλλαγή στον πυρήνα κάθε φιλοξενούμενης εικονικής μηχανής. Η εξομοίωση των συσκευών επιβαρύνει το χρόνο απόκρισης και λειτουργίας των ΕΜ αλλά η χρήση των επεκτάσεων εικονοποίησης των σύγχρονων επεξεργαστών μπορεί να επιταχύνει την εκτέλεση ορισμένων εντολών και να βελτιώσει την συνολική απόδοση των ΕΜ.

2.5.2 Τεχνική PV

Με αυτή τη μέθοδο, κάθε φιλοξενούμενη ΕΜ διαθέτει τροποποιημένο πυρήνα, τέτοιον ώστε να συνεργάζεται με το domain 0. Στην ουσία, κάθε VM "γνωρίζει" ότι λειτουργεί σε εικονικό περιβάλλον και όχι πάνω από φυσικούς πόρους, και έτσι συνεργάζεται αποδοτικά με την ΕΜ ελέγχου. Οι ΕΜ στέλνουν αιτήματα στο domain 0, το οποίο τα προωθεί στο φυσικό μηχάνημα και φροντίζει για τη σωστή εξυπηρέτησή τους. Έτσι ένα σύνολο από λειτουργίες όπως δικτύωση, είσοδος-έξοδος, διακοπές, χρονιστές, πίνακες σελίδων κλπ, υλοποιούνται μέσα από τεχνικές PV. Η τεχνική αυτή

δεν απαιτεί την εξομοίωση των φυσικών συσκευών και έτσι επιτυγχάνει καλύτερη απόδοση σε σύγκριση με την τεχνική HVM. Το Xen, πέρα από το domain 0, προσφέρει και ένα σύνολο από ειδικές εντολές, από τα εικονικά μηχανήματα προς τον hypervisor, τα hypercalls. Πρόκειται για αιτήματα που, μεταξύ άλλων, περιλαμβάνουν παραχώρηση δικαιωμάτων ανάγνωσης και εγγραφής σελίδων από τη μια ΕΜ στην άλλη, ανταλλαγή και αντιγραφή σελίδων μεταξύ τους, εγκατάσταση μηχανισμού αποστολής σημάτων μεταξύ VMs κ.α.

Η παρούσα διπλωματική εργασία εξετάζει τη λειτουργία VMs σε περιβάλλον PV και η πρώτη τεχνική δεν θα αναλυθεί περισσότερο.

2.6 POSIX Sockets

Τα POSIX sockets είναι ένα ευρέως χρησιμοποιούμενο πρότυπο επικοινωνίας μεταξύ εφαρμογών. Κάθε socket περιγράφει ένα τοπικό σημείο σύνδεσης στο οποίο μια εφαρμογή μπορεί να εγκαταστήσει μια σύνδεση με μια απομακρυσμένη εφαρμογή εκτελώντας ένα σύνολο από κλήσεις συστήματος socket (*socket calls*). Τα δύο άκρα της σύνδεσης διακρίνονται σε λειτουργία server και σε client. Το άκρο που αιτείται νέας σύνδεσης με την απομακρυσμένη εφαρμογή ονομάζεται πελάτης (client) και το άκρο που αποδέχεται τη σύνδεση ονομάζεται εξυπηρετητής (server). Ακολούθως περιγράφεται η διαδικασία επιτυχούς σύνδεσης και αποστολής δεδομένων μεταξύ δυο εφαρμογών χρησιμοποιώντας POSIX sockets.

Server:

- `socket()`: Δημιουργεί ένα νέο σημείο σύνδεσης, αρχικοποιεί τις απαραίτητες δομές στον πυρήνα του λειτουργικού συστήματος, ορίζοντας τον τύπο σύνδεσης (TCP, UDP, Unix, κ.α).
- `bind()`: Αντιστοιχίζει μια δομή socket με μια συγκεκριμένη διεύθυνση και πόρτα.
- `listen()`: Ορίζει τη δομή socket ως ένα παθητικό σημείο σύνδεσης. Αυτό σημαίνει ότι το συγκεκριμένο socket θα δέχεται νέα αιτήματα σύνδεσης και μπαίνει σε κατάσταση όπου "ακούει" για εισερχόμενες συνδέσεις.
- `accept()`: Αποδέχεται νέα αιτήματα σύνδεσης που στάλθηκαν στο συγκεκριμένο socket και επιστρέφει πίσω στην εφαρμογή ένα νέο σημείο, συνδεδεμένο με την απομακρυσμένη εφαρμογή, όπου ο χρήστης μπορεί με επιτυχία να στείλει και να λάβει δεδομένα.

Client:

- `socket()`: Ομοίως με τη λειτουργία του server.
- `connect()`: Στέλνει ένα νέο αίτημα σύνδεσης στο απομακρυσμένο socket, χρησιμοποιώντας την διεύθυνση και πόρτα προορισμού που το χαρακτηρίζει. Σε

περίπτωση που η απομακρυσμένη εφαρμογή βρίσκεται σε κατάσταση αναμονής για νέες συνδέσεις και αποδεχτεί το αίτημα, η λειτουργία αυτή επιστρέφει με επιτυχία και ο χρήστης μπορεί να στείλει και να λάβει δεδομένα.

Για την αποστολή και λήψη δεδομένων, και τα δύο άκρα χρησιμοποιούν τις κλήσεις συστήματος `send()` και `recv()` αντίστοιχα. Μια ενεργή σύνδεση socket περιλαμβάνει ένα αμφίδρομο σημείωση επικοινωνίας, δηλαδή κάθε άκρο της σύνδεσης έχει της δυνατότητα τόσο εγγραφής όσο και ανάγνωσης.

2.7 Μοντέλο δικτύωσης TCP/IP

Στην παρούσα υποενότητα, θα δοθεί μια συνοπτική παρουσίαση του πρωτοκόλλου TCP/IP. Το Transmission Control Protocol (TCP) είναι ένα από τα βασικά πρωτόκολλα που λειτουργούν στο επίπεδο μεταφοράς [26], πάνω από το πρωτόκολλο δικτύου IP. Αναλαμβάνει τη δημιουργία μια αξιόπιστης σύνδεσης μεταξύ δυο απομακρυσμένων σημείων. Δυο εφαρμογές που λειτουργούν σε απομακρυσμένα σημεία και θέλουν να επικοινωνήσουν μεταξύ τους μπορούν να δημιουργήσουν μια νέα σύνδεση TCP. Κάθε νέα σύνδεση χαρακτηρίζεται από τη διεύθυνση του αποστολέα, τη διεύθυνση του παραλήπτη και από τις αντίστοιχες πόρτες επικοινωνίας. Για παράδειγμα, μια εφαρμογή φυλλομετρητή (web browser) και ένας εξυπηρετητής ιστοσελίδων (web server) δημιουργούν μια νέα σύνδεση μεταξύ τους με σκοπό να ανταλλάξουν δεδομένα ιστοσελίδων. Κάθε άκρο της σύνδεσης χρησιμοποιεί μια συγκεκριμένη πόρτα TCP την οποία ελέγχει για νέα δεδομένα. Ο εξυπηρετητής ιστοσελίδων λαμβάνει αιτήματα συνδέσεων στη διεύθυνση IP του εξυπηρετητή και στην πασίγνωστη (well-known) πόρτα 80.

Το πρωτόκολλο TCP αναλαμβάνει την έναρξη και τον τερματισμό νέα σύνδεσης ανάμεσα στον αποστολέα και τον παραλήπτη και την αξιόπιστη μετάδοση των δεδομένων από άκρο σε άκρο. Από την πλευρά του αποστολέα, το πρωτόκολλο είναι υπεύθυνο για τη δημιουργία κατάλληλων σε μέγεθος πακέτων, για τη μετάδοσή τους στη σωστή σειρά χωρίς λάθη, για τον έλεγχο της ροής της σύνδεσης ώστε ο αποστολέας να μην στέλνει πιο γρήγορα απ΄ όσο μπορεί να επεξεργαστεί ο δέκτης και για τον έλεγχο συμφόρησης του δικτύου. Από την πλευρά του λήπτη, το πρωτόκολλο είναι υπεύθυνο για την ορθή λήψη των πακέτων, στη σωστή σειρά, για τη συναρμολόγησή τους και παράδοσή τους στην εφαρμογή.

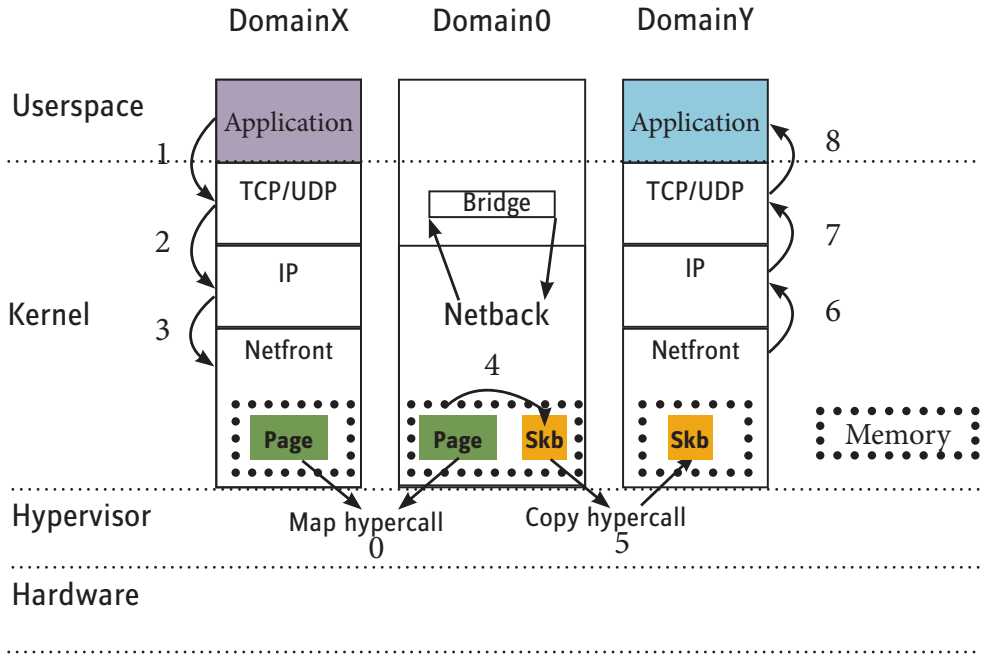
Το TCP, από την πλευρά του αποστολέα, λαμβάνει δεδομένα από την εφαρμογή και μετά την ορθή τοποθέτησή τους σε πακέτα, τα προωθεί στο πρωτόκολλο Internet - IP. Αυτό αναλαμβάνει την προώθησή τους, μέσω του δικτύου στον παραλήπτη. Στην ουσία, δεν γνωρίζει τίποτα για τις εφαρμογές που ανταλλάσσουν δεδομένα, για την κατάσταση του δικτύου, για τη σειρά των πακέτων. Εξετάζει τις διευθύνσεις IP αποστολέα και παραλήπτη των πακέτων και είναι υπεύθυνο να τα δρομολογήσει, να τα παραδώσει στο στρώμα συνδέσμου, στις δικτυακές συσκευές για αποστολή.

Τα πρωτόκολλα TCP και IP είναι συμπληρωματικά ως προς τη λειτουργικότητα που παρέχουν και πολύ συχνά αναφέρονται ως το πρωτόκολλο TCP/IP.

2.8 Μοντέλο δικτύωσης σε Xen PV Guest

Όπως αναφέρθηκε, το πρωτόκολλο TCP/IP παραδίδει τα πακέτα στο στρώμα συνδέσμου για αποστολή και αντίστοιχα, τα παραλαμβάνει από αυτό. Σε ένα φυσικό μηχάνημα, το στρώμα αυτό αντιστοιχεί σε δικτυακές συσκευές όπως κάρτες δικτύου, modems κ.α. Στις εικονικές μηχανές που λειτουργούν στο ίδιο φυσικό μηχάνημα όμως, τη λειτουργικότητα του δικτύου υλοποιεί ο hypervisor σε συνεργασία με άλλα στοιχεία. Συγκεκριμένα για paravirtualized περιβάλλοντα, ο πυρήνας κάθε φιλοξενούμενου ΛΣ είναι ειδικά τροποποιημένος ώστε να συνεργάζεται με το domain 0 και να παρέχει λειτουργίες δικτύωσης στα παραπάνω στρώματα.

Για την περίπτωση του hypervisor Xen, το σύστημα που είναι υπεύθυνο για τις λειτουργίες δικτύου ονομάζεται netback/netfront. Στο Σχ. 7 απεικονίζεται συνοπτικά η λειτουργία του netback/netfront. Η δικτύωση βασίζεται στο μοντέλο *χωριζομένων*



Σχήμα 7: Απεικόνιση της λειτουργίας του μοντέλου netback/netfront στο Xen. Οι αριθμοί αντιστοιχούν στα βήματα αποστολή ενός πακέτου TCP/IP από την EM1 στην EM2.

οδηγών (split-driver). Η EM ελέγχου (domain 0) είναι υπεύθυνη για το συντονισμό μεταξύ των δυο άκρων που βρίσκονται σε επικοινωνία. Ο οδηγός περιλαμβάνει δυο μέρη, το μπροστινό (front) που βρίσκεται στην φιλοξενούμενη EM και το πίσω μέρος (back) που βρίσκεται στην ειδική EM ελέγχου (domain 0).

Το ένα άκρο της σύνδεσης (EM1) προωθεί τα πακέτα που λαμβάνει από το πρωτόκολλο TCP/IP στο μπροστινό μέρος του οδηγού, το οποίο υλοποιείται ως οδηγός

μιας εικονικής συσκευής ethernet (netfront). Ο οδηγός στη συνέχεια αντιγράφει τα πακέτα σε μια περιοχή μνήμης πυρήνα η οποία είναι αντιστοιχισμένη (map) με περιοχή μνήμης πυρήνα του domain 0.

Αντιστοίχιση μνήμης (memory mapping) είναι η διαδικασία κατά την οποία μια περιοχή μνήμης του ενός VM παραχωρείται για χρήση και σε κάποιο άλλο VM. Έτσι κάθε εγγραφή που γίνεται σε αυτή τη μνήμη είναι ορατή ταυτόχρονα από όλες τις εικονικές μηχανές που έχουν αντιστοιχίσει τη συγκεκριμένη περιοχή μνήμης.

Στη συνέχεια, το πίσω μέρος του οδηγού, το οποίο βρίσκεται στο domain 0 (netback) διαβάζει τα πακέτα από την περιοχή μνήμης και τα αντιγράφει σε κατάλληλη δομή δεδομένων. Μια προγραμματιστική γέφυρα αναλαμβάνει να αντιγράψει τα δεδομένα απο τη δομή αυτή στον πυρήνα του άλλου άκρου της σύνδεσης (EM2), με τη χρήση ενός hypercall αντιγραφής. Στη συνέχεια ειδοποιεί τον παραλήπτη του πακέτου μέσω ενός μηχανισμού σημάτων. Το μπροστινό μέρος του οδηγού της EM2 αντιγράφει το πακέτο από τη μνήμη και το προωθεί στα ανώτερα στρώματα του πρωτοκόλλου.

Το βασικό μειονέκτημα αυτού του τρόπου επικοινωνίας μεταξύ εικονικών μηχανών είναι ότι όλη η κίνηση δεδομένων περνάει μέσω του της EM ελέγχου. Σε περιπτώσεις ταυτόχρονης ανταλλαγής δεδομένων από διαφορετικά ζευγάρια VMs ή από διαφορετικά ζευγάρια εφαρμογών μέσα στο ίδιο ζεύγος VMs, αυτό μπορεί να αποτελέσει ένα σημαντικό σημείο συμφόρησης και να επηρεάσει καθοριστικά την απόδοση του συστήματος. Άλλος ένας παράγοντας που καθιστά τη χρήση αυτού του τρόπου επικοινωνίας μη αποδοτικό είναι οι πολλαπλές λειτουργίες αντιγραφής στη μνήμη κατά τη μετάδοση ενός πακέτου από τον αποστολέα στον παραλήπτη. Συγκεκριμένα το μοντέλο netback/netfront περιλαμβάνει τέσσερις αντιγραφές, οι οποίες φαίνονται στο σχήμα στα βήματα 1,4,5 και 8.

3 Συναφείς εργασίες

Η βελτίωση της ενδο-επικοινωνίας εικονικών μηχανών παρουσιάζει μεγάλο ενδιαφέρον για τους λόγους που εξηγήθηκαν παραπάνω. Έτσι, υπάρχει μεγάλο πλήθος από εργασίες και προτάσεις στη συγκεκριμένη κατεύθυνση. Ένας σχεδιασμός που προτείνεται σε πολλές εργασίες περιλαμβάνει τη χρήση μοιραζόμενων χώρων αποθήκευσης στη φυσική μνήμη μεταξύ των ΕΜ.

Ο Diakhate et al. [27] χρησιμοποιεί τεχνικές μοιραζόμενης μνήμης στον KVM hypervisor [28] τροποποιώντας το QEMU [25]. Στο IVC [29] προτείνεται η δημιουργία ενός καναλιού μιας κατεύθυνσης (δηλαδή μόνο το ένα άκρο γράφει και το άλλο άκρο διαβάζει) χρησιμοποιώντας τεχνικές μοιραζόμενης μνήμης και δημιουργώντας μια νέα προγραμματιστική διεπαφή (API) Sockets.

Παρόμοια προσέγγιση γίνεται και στο XenSocket [30] το οποία χρησιμοποιεί μοιραζόμενες σελίδες μέσω του μηχανισμού grant table του Xen με σκοπό τη δημιουργία ενός καναλιού μιας κατεύθυνσης. Η συγκεκριμένη υλοποίηση τροποποιεί την προγραμματιστική διεπαφή BSD Sockets και παρουσιάζει μια νέα οικογένεια διευθύνσεων.

Επιπλέον, υλοποιείται το PC Calls [31] ένας μηχανισμός προώθησης κλήσεων συστήματος και εγκατάστασης καναλιού επικοινωνίας, προσανατολισμένος για χρήση σε Linux containers. Ο μηχανισμός αυτός διαφέρει ως προς την προώθηση κλήσεων συστήματος διότι περιλάμβανει ένα backend κομμάτι που λειτουργεί στην εικονική μηχανή ελέγχου.

Οι τεχνικές αυτές επιτυγχάνουν καλύτερα αποτελέσματα από πλευράς τόσο χρόνου απόκρισης όσο και εύρους ζώνης, σε σύγκριση με τον προκαθορισμένο μηχανισμό, ωστόσο απαιτούν την συγγραφή και μεταγλώττιση του κώδικα, παρουσιάζουν ελλείψεις συμβατότητας και οι εφαρμογές πρέπει να γνωρίζουν ότι τρέχουν σε εικονικό περιβάλλον και επικοινωνούν με άλλες ΕΜ που βρίσκονται στο ίδιο φυσικό μηχάνημα.

Ένας μηχανισμός που ξεπερνάει τα παραπάνω εμπόδια είναι το XenLoop [32]. Η συγκεκριμένη υλοποίηση δημιουργεί ένα αμφίπλευρο κανάλι μεταξύ ΕΜ διατηρώντας τη διαφάνεια κώδικα. Αυτό το επιτυγχάνει αξιοποιώντας το netfilter module [33] που παρέχει ο πυρήνας του Linux για τη σύλληψη των εξερχόμενων πακέτων. Το module είναι υπεύθυνο για την ανάλυση των επικεφαλίδων των πακέτων και παρέχει τη δυνατότητα κλήσεις συναρτήσεων όταν εντοπιστούν συγκεκριμένες επικεφαλίδες πακέτων. Έτσι, όταν εντοπιστεί πακέτο με διεύθυνση MAC προορισμού που βρίσκεται στο ίδιο φυσικό μηχάνημα, καλείται μια κατάλληλη συνάρτηση η οποία εγκαθιστά ένα αμφίπλευρο κανάλι και το προωθεί διαμέσω αυτού. Μια προγραμματιστική γέφυρα αναλαμβάνει την ορθή καταγραφή των διευθύνσεων MAC των ΕΜ που βρίσκονται στο ίδιο φυσικό μηχάνημα. Παρόλο που η συγκεκριμένη τεχνική μπορεί να πετύχει καλύτερα αποτελέσματα ως προς το throughput, η μείωση του χρόνου απόκρισης είναι ανεπαίσθητη, λόγω της πλήρης επεξεργασίας του πακέτου από όλη τη στοίβα TCP/IP.

Μία άλλη προσέγγιση στη βελτίωση της ενδο-επικοινωνίας εικονικών μηχανών παρουσιάζεται στα V4VSockets [34]. Πρόκειται για έναν πλήρες μηχανισμό που αξιοποιεί τα BSD Sockets ο οποίος επιτυγχάνει καλύτερα αποτελέσματα ως προς το χρόνο

απόκρισης αλλά και ως προς το εύρος ζώνης. Η βασική ιδέα βασίζεται σε αντίγραφα τα οποία πραγματοποιεί ο ίδιος ο hypervisor στον παραλήπτη των πακέτων μέσω του μηχανισμού V4V [35], ο οποίος είναι επέκταση του Xen. Η υλοποίηση αυτή διατηρεί τη δυνατότητα επαναχρησιμοποίησης κώδικα και τη διαφάνειά του, ωστόσο απαιτεί την τροποποίηση του ίδιου του hypervisor το οποίο έχει ως αποτέλεσμα την περιορισμένη συμβατότητα του κώδικα. Επιπλέον, η συνολική διαδρομή δεδομένων περιλαμβάνει τρεις λειτουργίες αντιγραφής μεταξύ του αποστολέα και του παραλήπτη.

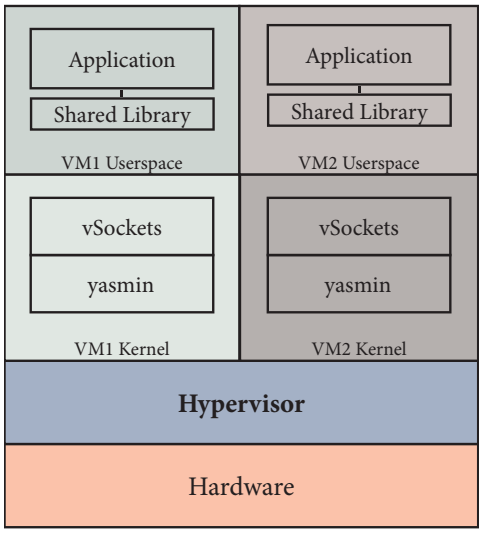
Ο μηχανισμός YASMIN ο οποίος υλοποιείται στην παρούσα διπλωματική, επιτυγχάνει την αξιοποίηση των θετικών χαρακτηριστικών των προηγούμενων υλοποιήσεων παρακάμπτωντας τόσο το domain ελέγχου όσο και το πρωτόκολλο TCP/IP. Παρέχει πλήρη διαφάνεια κώδικα, δεν απαιτεί την επανεγγραφή και μεταγλώττιση κώδικα, δεν επεμβαίνει στον hypervisor και διατηρεί τη συμβατότητα με άλλες πλατφόρμες.

4 Σχεδίαση και υλοποίηση

Για να ξεπεραστούν τα μειονεκτήματα του κλασσικού μηχανισμού επικοινωνίας VM που στεγάζονται στο ίδιο φυσικό μηχάνημα, σχεδιάστηκε και υλοποιήθηκε ο YASMIN σε Xen 4.4 χρησιμοποιώντας Linux 3.16 ως πυρήνα των φιλοξενούμενων μηχανών.

Ο σχεδιασμός του YASMIN βασίζεται στη Vchan [36], μια βιβλιοθήκη του Xen η οποία χρησιμοποιεί κλήσεις συστήματος (`open(), ioctl(), mmap()`) σε ειδικές συσκευές του Xen (`xen_gntdev, xen_gntalloc`) με σκοπό τη δημιουργία ενός καναλιού επικοινωνίας μεταξύ VM πάνω από το ίδιο φυσικό μηχάνημα, για την ανταλλαγή μηνυμάτων. Ο μηχανισμός YASMIN επεκτείνει τη βασική ιδέα της Vchan και υλοποιεί ένα επίπεδο μεταφοράς για το *vSockets* [37]. Το vSockets είναι μια προγραμματιστική διεπαφή (API) για Sockets, παρόμοια με το POSIX πρότυπο, η οποία υποστηρίζει γρήγορη και αποδοτική επικοινωνία μεταξύ φιλοξενούμενων εικονικών μηχανημάτων. Χρησιμοποιεί ένα νέο τύπο συνδέσεων, μια νέα οικογένεια διευθύνσεων, το `AF_VSOCK` για τις κοινές κλήσεις συστήματος Sockets (`socket(), bind(), connect()...`). Παρόμοια με τα POSIX Sockets, μια νέα σύνδεση μεταξύ δυο εικονικών μηχανημάτων μπορεί να εγκατασταθεί χρησιμοποιώντας τον αριθμό αναγνώρισης του VM (domain ID) και την πόρτα στην οποία απευθύνεται το αίτημα.

Το YASMIN αποτελείται από ένα module πυρήνα και μια βιβλιοθήκη που μεταβάλλει τις κλήσεις συστήματος socket. Η βιβλιοθήκη αυτή συλλέγει τα TCP/IP socket calls και τα μετατρέπει σε vSockets socket calls, χρησιμοποιώντας μια 1-1 αντιστοίχιση μεταξύ διευθύνσεων Internet και αναγνωριστικών domain ID. Μια συνοπτική απεικόνιση του βασικού κορμού του YASMIN δίνεται στο Σχ. 8.



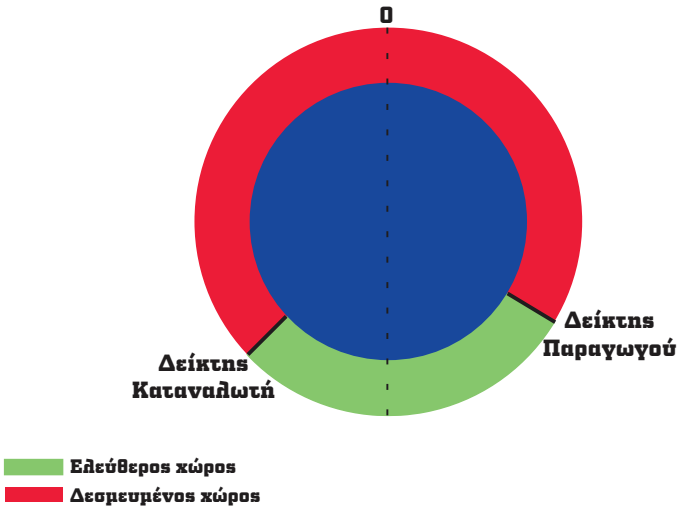
Σχήμα 8: Σχεδιαστική δομή του YASMIN

4.1 Περιγραφή υλοποίησης

Η υλοποίηση του YASMIN βασίζεται στη χρήση hypercalls, όπως περιγράφονται στην ενότητα 2.5.2. Αυτά περιλαμβάνουν την παραχώρηση δικαιωμάτων εγγραφής/ανάγνωσης σελίδων μεταξύ διαφορετικών EM, στην αντιστοίχιση σελίδων και στην αποστολή ειδικών σημάτων διακοπής μεταξύ εικονικών μηχανών. Τα βασικά hypercalls που χρησιμοποιήθηκαν στην υλοποίηση του μηχανισμού είναι:
`HYPERVISOR_grant_table_op(GNTTABOP_map_grant_ref,...)`: Mapping σελίδων.
`HYPERVISOR_grant_table_op(GNTTABOP_unmap_grant_ref,...)`: Unmapping σελίδων.
`HYPERVISOR_event_channel_op(EVTCHNOP_alloc_unbound,...)`: Δημιουργία νέου καναλιού εικονικών σημάτων.
`HYPERVISOR_event_channel_op(EVTCHNOP_bind_interdomain,...)`: Σύνδεση με απομακρυσμένο κανάλι σημάτων.
`HYPERVISOR_event_channel_op(EVTCHNOP_close,...)`: Τερματισμός καναλιού.

Επιπροσθέτως, γίνεται χρήση της δομής *μοιραζόμενου δακτυλίου παραγωγού - καταναλωτή* (*producer-consumer shared ring*). Δυο άκρα που θέλουν να ανταλλάξουν δεδομένα με τη χρήση μοιραζόμενης μνήμης, δημιουργούν μια δομή κυκλικού buffer η οποία χαρακτηρίζεται από δυο δείκτες, τη θέση εγγραφής και ανάγνωσης. Μόνο το ένα άκρο της σύνδεσης αναλαμβάνει την εγγραφή και μόνο το άλλο άκρο την ανάγνωση. Κάθε φορά που θέλει ο αποστολέας (*παραγωγός δεδομένων*) να στείλει δεδομένα στο άλλο άκρο, τα αντιγράφει στη μοιραζόμενη μνήμη και ενημερώνει το δείκτη παραγωγού στην τρέχουσα θέση εγγραφής. Αντίστοιχα, ο παραλήπτης (*καταναλωτής δεδομένων*) λαμβάνει τα δεδομένα από τη μοιραζόμενη μνήμη και ενημερώνει το δείκτη καταναλωτή. Τα δυο άκρα μπορούν να ελέγχουν αν υπάρχουν νέα δεδομένα για ανάγνωση ή αν υπάρχει θέση για νέα εγγραφή, συγκρίνοντας τους δείκτες καταναλωτή και παραγωγού. Επειδή ο χώρος αποθήκευσης είναι περιορισμένου μεγέθους, οι δείκτες επαναφέρονται κυκλικά στις αρχικές τους θέσεις, εξού και η ονομασία δακτύλιος. Βασικό πλεονέκτημα της χρήσης του producer-consumer shared ring για την ανταλλαγή δεδομένων, είναι ότι δεν απαιτείται κανένας μηχανισμός κλειδώματος της δομής μεταξύ των λειτουργιών εγγραφής και ανάγνωσης, διότι η ενημέρωση των δεικτών γίνεται με ατομικές εντολές (atomically) και ανεξάρτητα από κάθε συμβαλλόμενο μέρος.

Άλλο ένα βασικό χαρακτηριστικό του μηχανισμού ενδοεπικοινωνίας YASMIN είναι η αποδοτική ανταλλαγή δεδομένων διατηρώνας τη διαφάνεια εκτέλεσης, δηλαδή οι εφαρμογές δεν απαιτούν επανεγγραφή και διόρθωση, ούτε μεταγλώττιση. Αυτό επιτυγχάνεται παρακάμπτοντας το πρωτόκολλο TCP/IP, το οποίο εισάγει ένα επιπρόσθετο χρόνο επεξεργασίας των πακέτων και στη θέση του χρησιμοποιείται το πρωτόκολλο vSockets. Το πρωτόκολλο αυτό είναι τμήμα του πυρήνα Linux και υποστηρίζει τα εικονικά επίπεδα μεταφοράς VMCI [38] και VIRTIO [39]. Σε αυτή την κατεύθυνση, μέσα από την παρούσα διπλωματική εργασία, υλοποιείται ένα νέο εικονικό επίπεδο μεταφοράς αξιοποιώντας τους μηχανισμούς που παρέχει ο Xen hypervisor. Με αυτό τον τρόπο, όχι μόνο αποφεύγεται η υλοποίηση ενός νέου δικτυακού πρωτο-



Σχήμα 9: Δακτύλιος παραγωγού-καταναλωτή. Ο δείκτης παραγωγού δείχνει στη θέση όπου θα εγγράφονται τα νέα δεδομένα και ο δείκτης καταναλωτή στη θέση ανάγνωσης.

κόλλου από το μηδέν αλλά επίσης δίνεται η δυνατότητα στους χρήστες να επιλέγουν το επίπεδο μεταφοράς κατά την εκτέλεση των εφαρμογών. Ακολούθως δίνεται μια αρχιτεκτονική αποτύπωση του σχεδιασμού σε κάθε στρώμα επεξεργασίας:

Επίπεδο εφαρμογής: Όπως αναφέρθηκε, βασικό σημείο του μηχανισμού YASMIN είναι η συμβατότητα με τις ευρέως χρησιμοποιούμενες κλήσεις socket. Ο σχεδιασμός στοχεύει στη διατήρηση των εκτελέσιμων των εφαρμογών, χωρίς την ανάγκη για επανεγγραφή και μεταγλώττιση. Για να επιτευχθεί αυτό, υλοποιείται μια δυναμική βιβλιοθήκη εκτέλεσης η οποία συλλέγει τις κλήσεις συστήματος, φιλτράρει τις κλήσεις τύπου socket (`bind()`, `listen()`,κλπ) και τις αντικαθιστά ως εξής:
Η βιβλιοθήκη "ρωτάει" ένα αρχείο το οποίο περιλαμβάνει εγγραφές τύπου *διεύθυνση IP - αναγνωριστικό εικονικής μηχανής (domain ID)* για όλες τις EM που λειτουργούν στο ίδιο φυσικό μηχάνημα. Αν η διεύθυνση IP προορισμού των TCP/IP socket calls βρίσκεται σε εγγραφή στο αρχείο, τότε το IP socket call αντικαθίσταται από vSockets socket call με διεύθυνση προορισμού το αντίστοιχο αναγνωριστικό που βρίσκεται στη συγκεκριμένη εγγραφή στο αρχείο. Αν όχι, τότε ακολουθείται το προκαθορισμένο μονοπάτι netback/netfront.

Επίπεδο μεταφοράς-συνδέσμου: Κάθε κλήση socket τύπου `AF_VSOCK` που προωθείται στον πυρήνα, εξυπηρετείται από το πρωτόκολλο vSockets. Αυτό είναι υπεύθυνο για τον κατακερματισμό και την παράδοση των πακέτων στο στρώμα μεταφοράς. Επιπλέον, το πρωτόκολλο είναι υπεύθυνο για το μπλοκάρισμα (sleep) των λειτουργιών του αποστολέα και του παραλήπτη, σε περίπτωση που ο ring buffer είναι γεμάτος ή άδειος αντίστοιχα.

Το στρώμα μεταφοράς είναι ο πυρήνας της υλοποίησης του YASMIN και έχει τη δυνατότητα να δημιουργεί κανάλι επικοινωνίας μεταξύ εικονικών μηχανών του ίδιου φυσικού μηχανήματος, να παραδίδει μηνύματα και να ειδοποιεί τον παραλήπτη για νέα πακέτα. Είναι υλοποιημένο με τη μορφή module που εισάγεται δυναμικά στον πυρήνα κάθε φιλοξενούμενης ΕΜ. Το επίπεδο συνδέσμου είναι στην ουσία ένας ring buffer παραγωγού καταναλωτή ο οποίος βρίσκεται στην από κοινού μοιραζόμενη μνήμη από το ζευγάρι εικονικών μηχανών.

Εγκατάσταση καναλιού: Ο hypervisor Xen παρέχει το μηχανισμό *grant table* ο οποίος δίνει τη δυνατότητα διαμοιρασμού σελίδων μεταξύ διαφορετικών εικονικών μηχανών. Το ένα domain (διαμοιραστής) δεσμεύει μια σελίδα, παραχωρεί δικαιώματα εγγραφής και ανάγνωσης σελίδας στην άλλη ΕΜ καλώντας ένα ειδικό hypercall. Το hypercalls επιστρέφει ένα δείκτη στο grant table για να αναφέρεται στη συγκεκριμένη σελίδα.

```
int gnttab_grant_foreign_access(domid_t domid,
unsigned long frame,int readonly);
```

Η συνάρτηση αυτή παραχωρεί δικαιώματα εγγραφής ή/και ανάγνωσης σε απομακρυσμένο VM και επιστρέφει το δείκτη της διαμοιραζόμενης σελίδας στον πίνακα grant.

Το απέναντι VM (διαμοιραζόμενος) δεσμεύει επίσης μια σελίδα και αντιστοιχίζει αυτή τη σελίδα με τη σελίδα που του παραχώρησε ο διαμοιραστής. Αυτό γίνεται με την χρήση hypercall στο οποίο δίνεται ως παράμετρος ο δείκτης πίνακα που αναφέρθηκε πριν.

```
HYPERVISOR_grant_table_op(GNTTABOP_map_grant_ref, void *uop,
unsigned int count);
```

Το hypercall κάνει *map* (αντιστοίχιση) μια σελίδας της τοπικής ΕΜ με μια σελίδα της απομακρυσμένης ΕΜ χρησιμοποιώντας το δείκτη της σελίδας στον πίνακα grant.

Ο δακτύλιος παραγωγού-καταναλωτή είναι τμήμα του καναλιού επικοινωνίας και υλοποιείται ως ένα σύνολο από διαμοιραζόμενες σελίδες μεταξύ των επικοινωνούντων εικονικών μηχανών, χρησιμοποιώντας τον μηχανισμό που μόλις περιγράφηκε.

Τέλος, το Xen παρέχει ένα απλό μηχανισμό αποστολής σημάτων μεταξύ VMs, το μηχανισμό *event channel*, οποίος αξιοποιείται για την ενημέρωση των εικονικών μηχανών για τη λήψη νέων πακέτων. Το ένα άκρο της σύνδεσης (*καναλάρχης*) δημιουργεί ένα νέο κανάλι μηνυμάτων με το άλλο άκρο (*"θεατής"*) καλώντας ένα hypercall το οποίο επιστρέφει τον τοπικό αριθμό πόρτας:

```
port = HYPERVISOR_event_channel_op(EVTCHNOP_alloc_unbound,
&alloc_unbound);.
```

Στη συνέχεια δηλώνει ένα νέο χειριστή διακοπών (interrupt handler) ο οποίος συνδέεται με το νέο κανάλι. Ο "θεατής" εγγράφεται στο κανάλι καλώντας ένα hypercall με παράμετρο τον τοπικό αριθμό σύνδεσης του καναλάρχη:

```
port = HYPERVISOR_event_channel_op(EVTCHNOP_alloc_unbound,
&alloc_unbound);.
```

Αντίστοιχα, δηλώνει ένα νέο χειριστή διακοπών ο οποίος με τη σειρά του συνδέεται με το κανάλι. Κάθε άκρο της σύνδεσης μπορεί να καλεί ένα hypercall και να προκαλεί την αποστολή εικονικής διακοπής στο άλλο άκρο, με σκοπό να το ενημερώσει για νέα δεδομένα που είναι διαθέσιμα στο δακτύλιο:
`(void) HYPERVISOR_event_channel_op(EVTCHNOP_send, &send);`.

Στη συνέχεια δίνεται μια βηματική περιγραφή μια επιτυχημένους σύνδεσης και αποστολής δεδομένων μεταξύ ενός ζεύγους εικονικών μηχανών. Μια σχηματική απεικόνιση αυτής της διαδικασίας δίνεται στο Σχ. 10. Το ζευγάρι εικονικών μηχανών EM1 και EM2 θέλουν να ανταλλάξουν δεδομένα πάνω από TCP/IP δίκτυο . Η EM1 αντιστοιχεί στον πελάτη της σύνδεσης και η EM2 αντιστοιχεί στον εξυπηρετητή. Ο πελάτης βρίσκεται στην τοπική διεύθυνση IP 10.0.0.1 και έχει το domain ID 1 και ο εξυπηρετητής στη διεύθυνση 10.0.0.2 και έχει το domain ID 2. Εκτελούνται οι εξής ενέργειες:

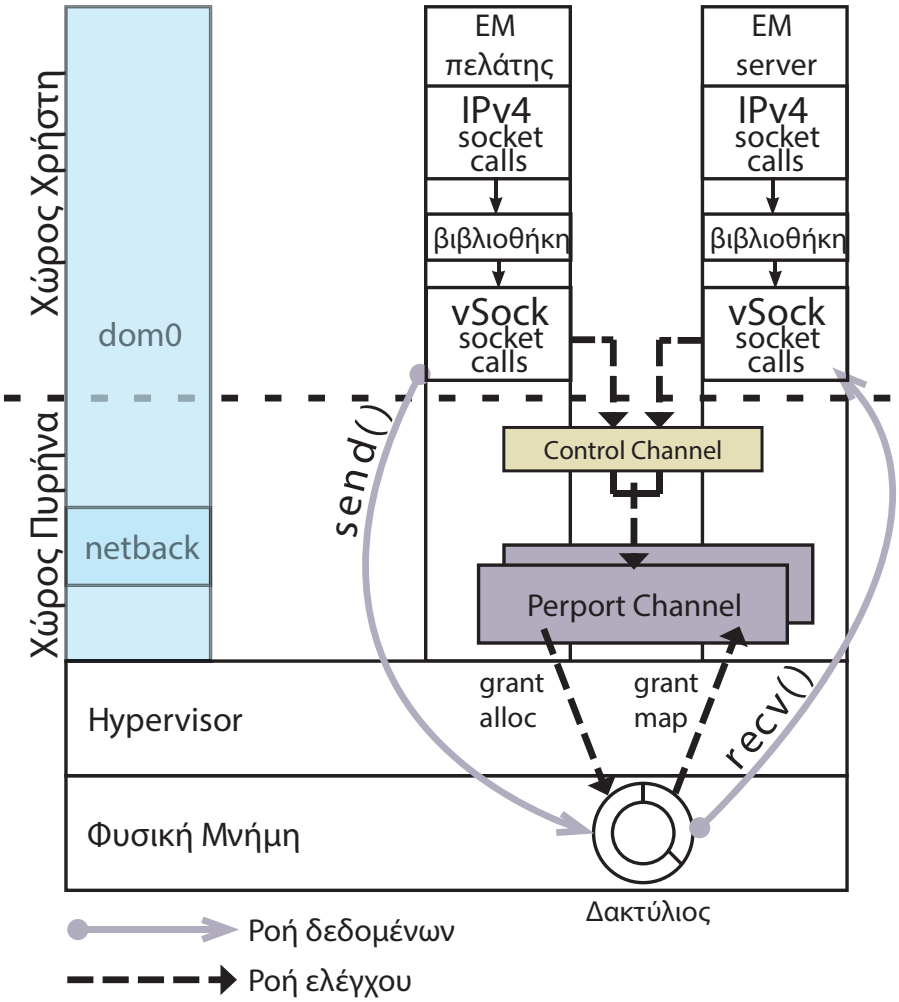
1. Οι εικονικές μηχανές εισάγουν το module yasmin στον πυρήνα τους. Με την εισαγωγή του δημιουργείται ένα καινούργιο directory σε επιλεγμένη διαδρομή του XenStore (π.χ `/local/domain/0/data/yasmin`), το οποίο παρακολουθείται για νέα αιτήματα συνδέσεων. Κάθε εγγραφή στο συγκεκριμένο κατάλογο προκαλεί την εκκίνηση μια ρουτίνας εξυπηρέτησης, η οποία είναι υπεύθυνη για την κατάλληλη επεξεργασία των αιτημάτων.

2. Η EM1 καλεί την κλήση `socket()` για τη δημιουργία ενός νέο σημείου σύνδεσης τύπου TCP/IP. Η δυναμική βιβλιοθήκη λαμβάνει την κλήση συστήματος και την τροποποιεί κατάλληλα ώστε να την προωθήσει στον πυρήνα σε μορφή `socket()` τύπου `AF_VSOCK`. Ο πυρήνας λαμβάνει το αίτημα και το προωθεί στην κατάλληλη ρουτίνα εξυπηρέτησης του module, όπου γίνεται κατανομή πόρων και αρχικοποίηση των δομών του σημείου σύνδεσης.

3. Αντίστοιχη σειρά ενεργειών `socket()` εκτελεί και η EM2 για τη δημιουργία του άλλου άκρου της σύνδεσης.

4. Ο εξυπηρετητής (10.0.0.2) εκτελεί τις κλήσεις `bind()` και `listen()` οι οποίες οριοθετούν την ίδια σημασιολογία με τα αντίστοιχα POSIX socket calls. Παρόμοια, η δυναμική βιβλιοθήκη λαμβάνει τις κλήσεις συστήματος, τις τροποποιεί και τις προωθεί κατάλληλα στον πυρήνα, όπου παραδίδονται στο πρωτόκολλο vSockets για επεξεργασία. Μετά την επιτυχή ολοκλήρωσή τους, η παραπάνω δομή socket του εξυπηρετητή βρίσκεται σε κατάσταση αναμονής για νέες συνδέσεις.

5. Στη συνέχεια, η εφαρμογή - πελάτης εκτελεί την κλήση `connect()` προς την απομακρυσμένη διεύθυνση 10.0.0.2. Η δυναμική βιβλιοθήκη λαμβάνει την κλήση και ελέγχει τη διεύθυνση προορισμού εξετάζοντας το αρχείο που περιγράφηκε παραπάνω. Επειδή η διεύθυνση 10.0.0.2, η οποία αντιστοιχεί στην EM2, βρίσκεται στο ίδιο φυσικό μηχάνημα, το αίτημα σύνδεσης TCP προς αυτή τη



Σχήμα 10: Υλοποίηση του YASMIN. Κάθε νέα σύνδεση πραγματοποιείται μέσω του *καναλιού ελέγχου* (βέλος μονοπατιού ελέγχου). Για κάθε ζεύγος συνδεδεμένων sockets, ένα νέο κανάλι perport δημιουργείται, με το δικό του δακτύλιο παραγωγού-καταναλωτή, όπου ανταλλάσσονται δεδομένα (βέλος μονοπατιού δεδομένων).

διεύθυνση μετατρέπεται σε αίτημα σύνδεσης τύπου `AF_VSOCK` προς τη διεύθυνση 2 (αναγνωριστικό εικονικής μηχανής). Η μετασχηματισμένη κλήση συστήματος προωθείται προς τον πυρήνα και εν συνεχεία προς το πρωτόκολλο vSockets και το επίπεδο μεταφοράς yasmin.

(α΄) Το πρωτόκολλο μεταφοράς λαμβάνει το αίτημα για σύνδεση προς την εικονική μηχανή με αναγνωριστικό 2 και ελέγχει ότι δεν έχει υπάρξει προηγούμενη σύνδεση από τη συγκεκριμένη EM προς την EM προορισμού.

(β΄) Στη συνέχεια δημιουργεί ένα κανάλι ελέγχου μεταξύ των δυο εικονικών μηχανών με σκοπό την ανταλλαγή μηνυμάτων διαχείρισης των συνδέσεων. Αρχικά, δεσμεύεται μια σελίδα στη μνήμη πυρήνα του EM1 και παραχωρούνται δικαιώματα στην EM2, καθώς επίσης δημιουργείται και ένα κανάλι σημάτων , όπως περιγράφεται αναλυτικά παραπάνω.

(γ΄) Το αναγνωριστικό της εγγραφής στον grant table και η τοπική πόρτα του event channel εγγράφονται στη διαδρομή XenStore που δημιούργησε η EM2 κατά την εισαγωγή του module στον πυρήνα.

(δ΄) Η ρουτίνα εξυπηρέτησης του VM2 ξυπνάει και ελέγχει τις εγγραφές στη διαδρομή. Επειδή πρόκειται για ένα νέο αίτημα δημιουργίας καναλιού ελέγχου με το VM1, ακολουθεί την αντίστοιχη διαδικασία αντιστοίχισης σελίδας (mapping) και καταχώρησης του καναλιού σημάτων και επιβεβαιώνει στον αποστολέα την ορθή δημιουργία του καναλιού. Οι εικονικές μηχανές 1 και 2 μοιράζονται πλέον ένα κανάλι ελέγχου, στο οποίο κάθε άκρο μπορεί να αποστείλει πακέτα ελέγχου (*αιτήματα νέων συνδέσεων socket, τερματισμός ενεργών συνδέσεων κλπ.*) και να ενημερώσει το άλλο άκρο για την ορθή αποστολή τους. Η ανταλλαγή πακέτων γίνεται με τη χρήση μίας διαμοιραζόμενης σελίδας στη φυσική μνήμη του μηχανήματος. Κάθε ζευγάρι από εικονικές μηχανές που επικοινωνούν δημιουργούν μόνο ένα κανάλι ελέγχου το οποίο υφίσταται για όσο βρίσκεται σε χρήση το module στους πυρήνες των EM.

(ε΄) Μετά την επιτυχή δημιουργία καναλιού ελέγχου μεταξύ των δυο εικονικών μηχανών, πραγματοποιείται η εγκατάσταση του καναλιού επικοινωνίας μεταξύ του ζεύγους των sockets. Το κανάλι αυτό (*persocket channel*) είναι υπεύθυνο για την ανταλλαγή δεδομένων μεταξύ των εφαρμογών και αντιστοιχεί σε ένα μόνο ζευγάρι συνδεδεμένα sockets. Έτσι, κάθε ανεξάρτητη εφαρμογή που λειτουργεί στην ίδια εικονική μηχανή και θέλει να επικοινωνήσει με διαφορετική εφαρμογή στην ίδια απομακρυσμένη εικονική μηχανή, εγκαθιστά ένα καινούργιο persocket κανάλι. Το module yasmin δεσμεύει ένα πλήθος από σελίδες πυρήνα και δημιουργεί ένα event channel μεταξύ των VM, ακολουθώντας την ίδια διαδικασία με τη χρήση hypercalls.

(ϛ΄) Αποστέλλει τους δείκτες του grant table που επιστράφηκαν από τα hypercalls και την τοπική πόρτα event channel στην απομακρυσμένη εικονική

μηχανή, χρησιμοποιώντας το κανάλι ελέγχου και την ενημερώνει για την άφιξη νέου πακέτων προκαλώντας εικονική διακοπή.

(ζ′) Η ρουτίνα εξυπηρέτησης της διακοπής ξυπνάει, η οποία επεξεργάζεται το πακέτο. Η πλευρά του εξυπηρετητή αντιστοιχίζει (map) τις σελίδες και καταχωρεί το κανάλι εικονικών διακοπών παρόμοια με πριν. Αποστέλλει μήνυμα επιβεβαίωσης εγκατάστασης σύνδεσης και προκαλεί εικονική διακοπή πίσω στον πελάτη.

(η′) Ο πελάτης δέχεται την διακοπή και ενημερώνεται για την επιτυχημένη εγκατάσταση του καναλιού persocket, οπότε η `connect()` επιστρέφει με επιτυχία πίσω στο χρήστη.

6. Το κανάλι ανταλλαγής δεδομένων μεταξύ των εφαρμογών είναι έτοιμο, οπότε ο πελάτης καλεί την `send()` για να στέλνει ένα μήνυμα στην απομακρυσμένη εφαρμογή. Η δυναμική βιβλιοθήκη συλλέγει την κλήση συστήματος και την προωθεί κατάλληλα στον πυρήνα. Το πρωτόκολλο vSockets λαμβάνει το αίτημα, ελέγχει το διαθέσιμο χώρο αποστολής και φροντίζει για την πλήρη παράδοση του μηνύματος. Σε περίπτωση που δεν υπάρχει αρκετός διαθέσιμος χώρος για την αποστολή των δεδομένων, η κλήση συστήματος "κοιμάται", περιμένοντας την απομακρυσμένη εφαρμογή να διαβάσει τα δεδομένα και να απελευθερώσει χώρο. Το πρωτόκολλο παραδίδει τα δεδομένα στο επίπεδο μεταφοράς yasmin το οποίο αναλαμβάνει την αντιγραφή τους στο δακτύλιο παραγωγού-καταναλωτή. Τέλος, προκαλεί εικονική διακοπή στην EM2 για να την ενημερώσει για την άφιξη δεδομένων.

7. Αντίστοιχα, η EM2 καλεί τη συνάρτηση `recv()` για την ανάγνωση δεδομένων από το socket. Παρόμοια με τη διαδικασία αποστολής, η κλήση συστήματος μπλοκάρει περιμένοντας για νέα δεδομένα, τα οποία αντιγράφει από το δακτύλιο παραγωγού-καταναλωτή.

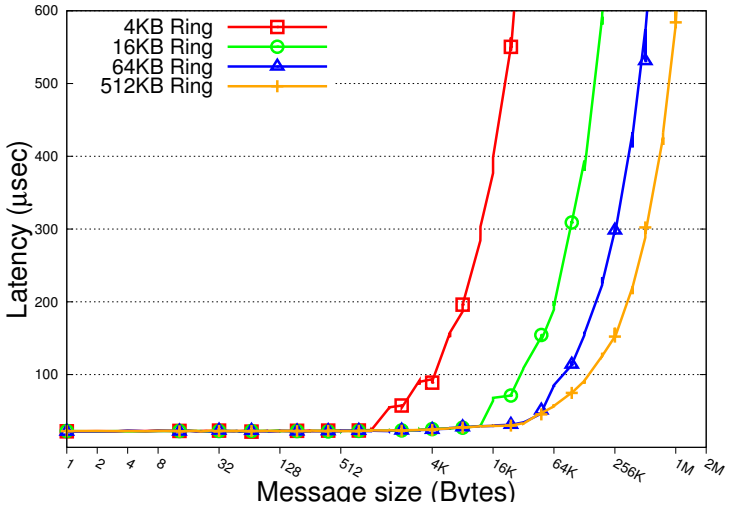
Βασικό πλεονέκτημα της παραπάνω διαδικασίας είναι η παράκαμψη του πρωτοκόλλου TCP/IP η οποία εισάγει επιπλέον χρόνο εκτέλεσης σε κάθε αποστολή πακέτου καθώς και η ελαχιστοποίηση των λειτουργικών αντιγραφής δεδομένων από τον αποστολέα στον παραλήπτη. Έτσι, η διαδικασία αποστολής και λήψης πακέτου περιλαμβάνει μόνο δυο λειτουργίες αντιγραφής: την αντιγραφή από το χώρο μνήμης της εφαρμογής-αποστολέα (userspace) στο διαμοιραζόμενη δακτύλιο (`copy_from_user()`) και την αντιγραφή από το δακτύλιο στο χώρο μνήμης της εφαρμογής-παραλήπτη (`copy_to_user()`). Τέλος, να σημειωθεί ότι η αποστολή πακέτων περιλαμβάνει τη δέσμευση χώρου στο δακτύλιο παραγωγού-καταναλωτή, την κατάλληλη αρχικοποίησή τους με λειτουργίες εγγραφής και την αντίστοιχη ανάγνωσή τους από το δακτύλιο. Έτσι δεν απαιτείται η επιπλέον δέσμευση μνήμης και οι λειτουργίες αντιγραφής μνήμης από το χώρο πυρήνα στο δακτύλιο και αντίστροφα.

5 Πειραματική αποτίμηση

Για την αξιολόγηση του μηχανισμού YASMIN χρησιμοποιούνται μονοπύρηνες εικονικές μηχανές και συγκρίνονται οι αποδόσεις του YASMIN με τον προεπιλεγμένο μηχανισμό netback/netfront σε διάφορα πειράματα. Συγκεκριμένα, χρησιμοποιούνται ειδικές σουίτες αξιολόγησης επιδόσεων δικτύων για την εκτίμηση τόσο του χρόνου απόκρισης (latency) όσο και του εύρους ζώνης. Επίσης συγκρίνεται η υλοποίηση με την απόδοση του μηχανισμού επικοινωνίας Unix sockets σε φυσικό μηχάνημα, δηλαδή χωρίς τη χρήση τεχνικών εικονοποίησης, αλλά και με την ταχύτητα του διαύλου μνήμης. Τέλος αξιολογείται η απόδοση του YASMIN ως προς την κλιμακωσιμότητα (scaling) δηλαδή τη δυνατότητα παράλληλης εκτέλεσης μεγάλου αριθμού από εικονικές μηχανές. Η εκτέλεση των πειραμάτων γίνεται σε φυσικό μηχάνημα με 2x Xeon E5335, 8GB RAM, όπου ο κάθε επεξεργαστής διαθέτει 4 πυρήνες με 32KBytes L1 Cache (κρυφή μνήμη) και 4MB L2 Cache.

Χρησιμοποιήθηκε η εφαρμογή NetPIPE [40] για τη μέτρηση του χρόνου απόκρισης και της κλιμακωσιμότητας, η εφαρμογή Iperf [41] για την μέτρηση του εύρους ζώνης, το netperf [42] για τη μέτρηση της διεκπεραιωτικής ικανότητας των Unix Domain Sockets και το STREAM [43] για την απόδοση του διαύλου μνήμης.

5.1 Καθορισμός μεγέθους δακτυλίου



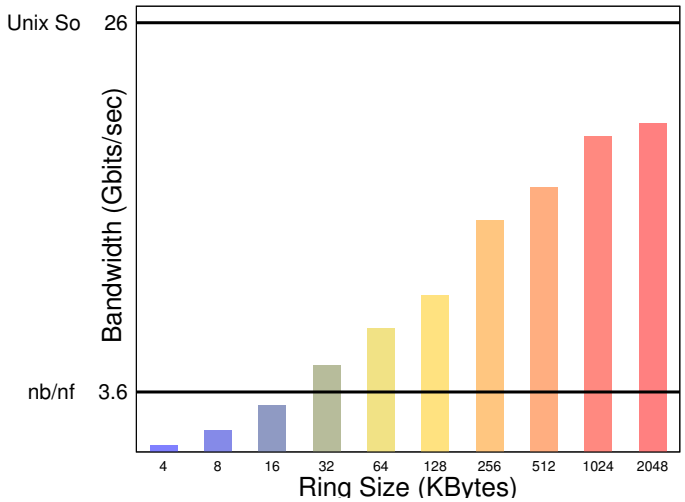
Σχήμα 11: Απεικόνιση του χρόνου απόκρισης (latency) ως προς το μέγεθος μηνύματος, για διαφορετικά μεγέθη δακτυλίου παραγωγού-καταναλωτή. Για μικρά μεγέθη μηνυμάτων ο χρόνος απόκρισης παραμένει αμετάβλητος με την αύξηση του μεγέθους του δακτυλίου.

Μια σημαντική παράμετρος που καθορίζει τη συνολική απόδοση του συστήματος είναι το μέγεθος του δακτυλίου παραγωγού-καταναλωτή. Στο Σχήμα 11 παρατηρείται

η επίδραση της επιλογής δακτυλίου στο χρόνο απόκρισης. Η αύξηση του μεγέθους του δακτυλίου δεν επηρεάζει το latency για μικρά μεγέθη μηνύματος. Αντίθετα όμως, όπως φαίνεται στο Σχήμα 12, η αύξηση της συγκεκριμένης παραμέτρου επιδρά καθοριστικά στο throughput της υλοποίησης.

Συγκριτικά, μπορούμε να παρατηρήσουμε ότι το YASMIN φτάνει έως το 76% της απόδοσης των Unix Socket σε φυσικό μηχάνημα για μέγεθος δακτυλίου στα 2MB.

Παρά την καλύτερη απόδοση για μεγέθη δακτυλίου μεγαλύτερα από 1MB, επιλέχθηκε το μέγεθος των 512kB (128 δεσμευμένες σελίδες μνήμης), ανταλλάσσοντας bandwidth με μειωμένη κατανάλωση μνήμης πυρήνα. Κάθε συνδεδεμένο socket διατηρεί δεσμευμένη μνήμη στον πυρήνα της εικονική μηχανής. Έτσι, όταν υπάρχουν πολλές ανοιχτές συνδέσεις socket από ένα VM (για παράδειγμα ένας rCUDA server), υπάρχει κίνδυνος δέσμευσης μεγάλου μεγέθους μνήμης στον πυρήνα.



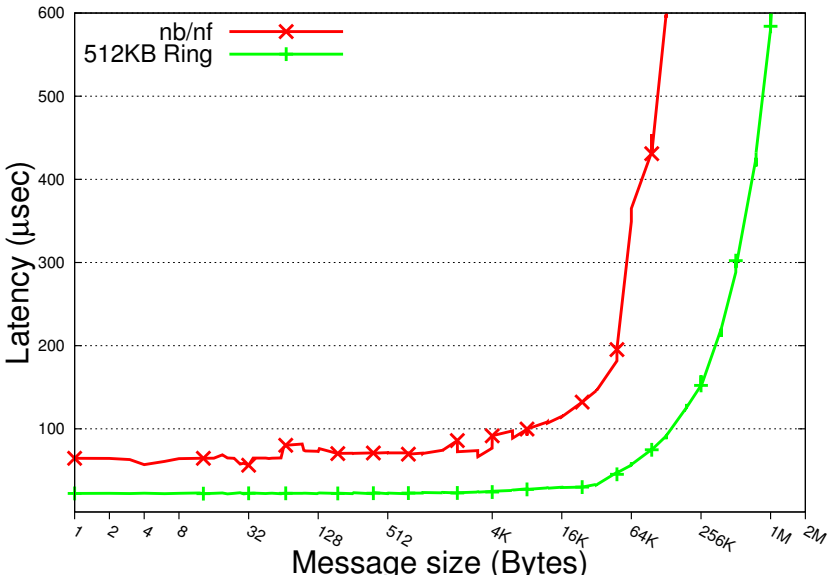
Σχήμα 12: Απεικόνιση της διεκπεραιωτικής ικανότητας του YASMIN ως προς το μέγεθος δακτυλίου. Το εύρος ζώνης του συστήματος, για μέγεθος δακτυλίου 2MB, αντιστοιχεί στο 76% της απόδοσης bare-metal Unix Domain Sockets.

5.2 Χρόνος απόκρισης

Στο Σχήμα 13 απεικονίζεται η απόδοση του YASMIN ως προς τον χρόνο απόκρισης (latency). Συγκεκριμένα, αξιολογείται ο χρόνος που κάνει ένα μήνυμα να φτάσει από την εφαρμογή-πελάτη στον εφαρμογή-εξυπηρετητή. Το διάγραμμα απεικονίζει το χρόνο απόκρισης σε συνάρτηση με το μέγεθος του μηνύματος.

Η συγκεκριμένη υλοποίηση (για δακτύλιο μεγέθους 128 σελίδων) επιτυγχάνει μειωμένο χρόνο απόκρισης σε σχέση με το μοντέλο netback/netfront κατά 65%[8].

[8]Η μείωση αυτή αντιστοιχεί σε μέγεθος μηνύματος 1Byte



Σχήμα 13: Χρόνος απόκρισης (χρόνος μεταξύ επιτυχημένης αποστολής και λήψης μηνύματος) σε σχέση με το μέγεθος του απεσταλμένου μηνύματος. Το μέγεθος του δακτυλίου έχει επιλεγεί στα 512kB.

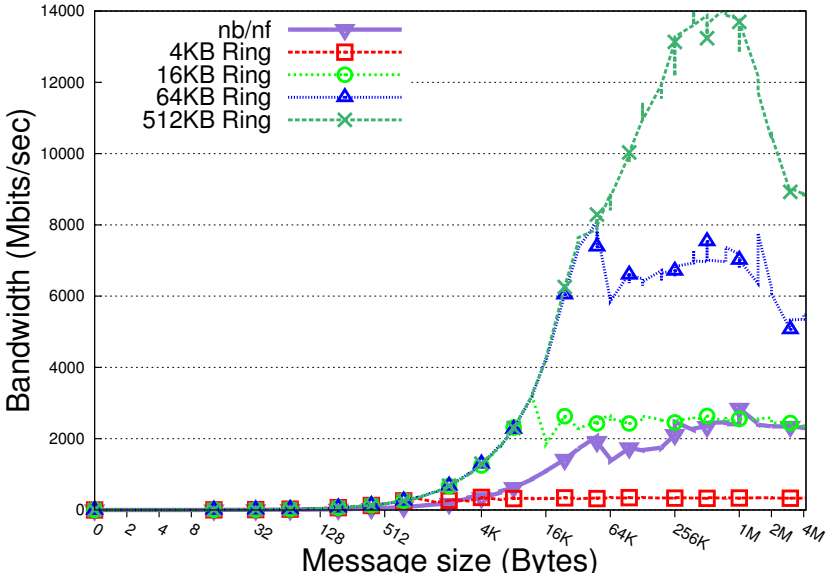
Αυτό οφείλεται στο γεγονός ότι παρακάμπτεται όλη η στοίβα TCP/IP η οποία επιβαρύνει την αποστολή κάθε πακέτου. Η χρήση του μηχανισμού YASMIN περιλαμβάνει μόνο δυο λειτουργίες αντιγραφής, όπως εξηγήθηκε παραπάνω, το οποίο με τη σειρά του, μειώνει το χρόνο αποστολής και λήψης κάθε πακέτου.

5.3 Εύρος ζώνης

Το YASMIN επιτυγχάνει βελτίωση της διεκπεραιωτικής ικανότητας του συστήματος, όπως φαίνεται και στο Σχήμα 14. Για μέγεθος δακτυλίου στα 512KB, η υλοποίηση επιτυγχάνει αύξηση κατά 4.4, σε σύγκριση με το netback/netfront όπως μετρήθηκε με τη χρήση του Iperf.

Συγκριτικά, το YASMIN επιτυγχάνει απόδοση 16Gbps (2048 MBytes/sec) ενώ η απόδοση του διαύλου της μνήμης μετρήθηκε στα 2813 MBytes/sec για 1 thread υπό εκτέλεση και στα 3770 MBytes/sec για 8 threads υπό εκτέλεση. Τα Unix Domain Sockets πετυχαίνουν εύρος ζώνης στα 3250 MBytes/sec.

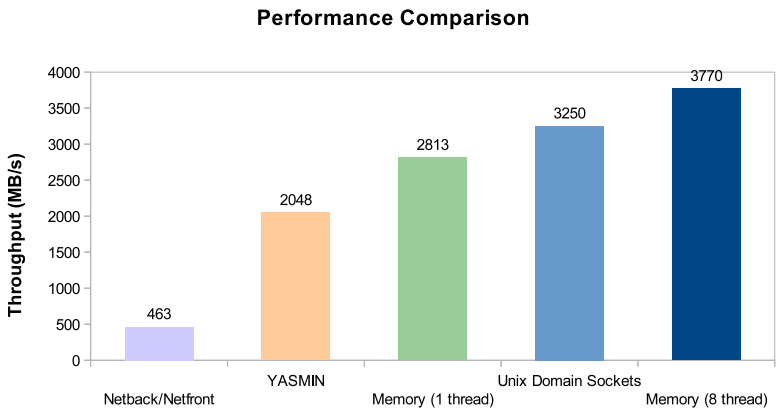
Τα Unix Domain Sockets επιτυγχάνουν καλύτερη απόδοση από την εγγενή απόδοση του διαύλου της μνήμης για 1 εκτελούμενο thread, η οποία μετρήθηκε με τη χρήση του STREAM, διότι κάνουν αποδοτικότερη χρήση των λανθάνουσων μνημών (cache memories). Πράγματι, καταγράφοντας τους Μετρητές Απόδοσης (Performance Counters) του συστήματος, βρέθηκε ότι κατά την εκτέλεση του πειράματος STREAM τα cache misses έφτασαν στο 32.6% των αντίστοιχων cache hits, ενώ κατά τη μέτρηση της απόδοσης των Unix Domain Sockets, τα cache misses ήταν μόνο το 0,02% των



Σχήμα 14: Στο Σχήμα απεικονίζεται η επίδραση του εύρους μηνύματος στην απόδοση (throughput) του μηχανισμού YASMIN. Για μεγέθη μηνύματος που πλησιάζουν στο μέγεθος μνήμης cache (4MB) παρατηρείται σημαντική πτώση της απόδοσης, η οποία εκτιμάται ότι οφείλεται σε κορεσμό της λανθάνουσας μνήμης (4MB).

αντίστοιχων cache hits.

Στο Σχήμα 15 φαίνεται η σύγκριση του εύρους ζώνης της υλοποίησης YASMIN με την απόδοση του netback/netfront, Unix Domain Sockets καθώς και την εγγενή απόδοσης του διαύλου της μνήμης.

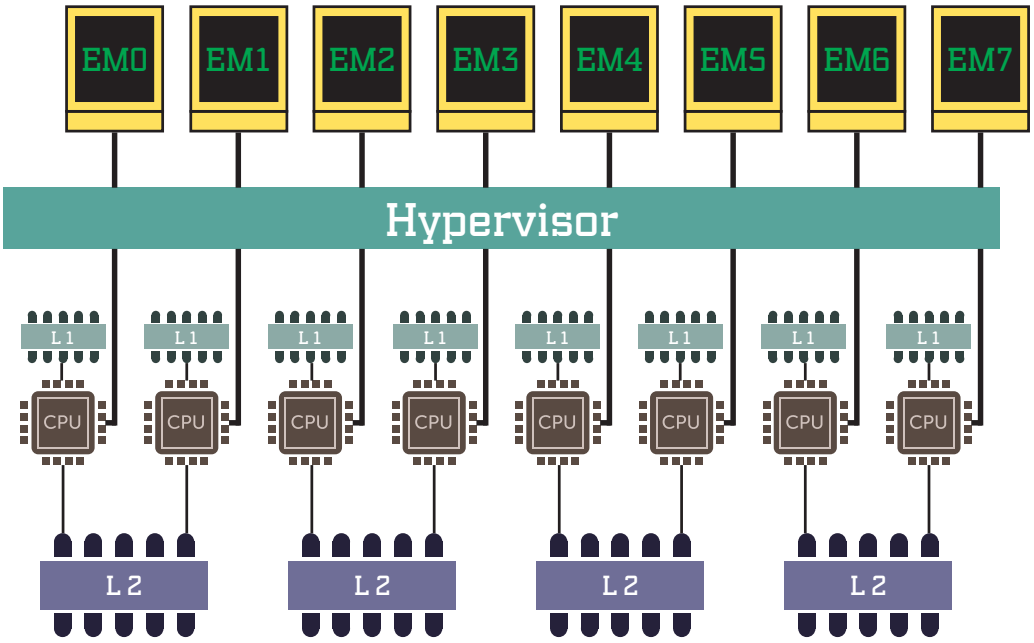


Σχήμα 15: Συγκριτική αποτίμηση της υλοποίησης YASMIN.

5.4 Κλιμακωσιμότητα - Scalability

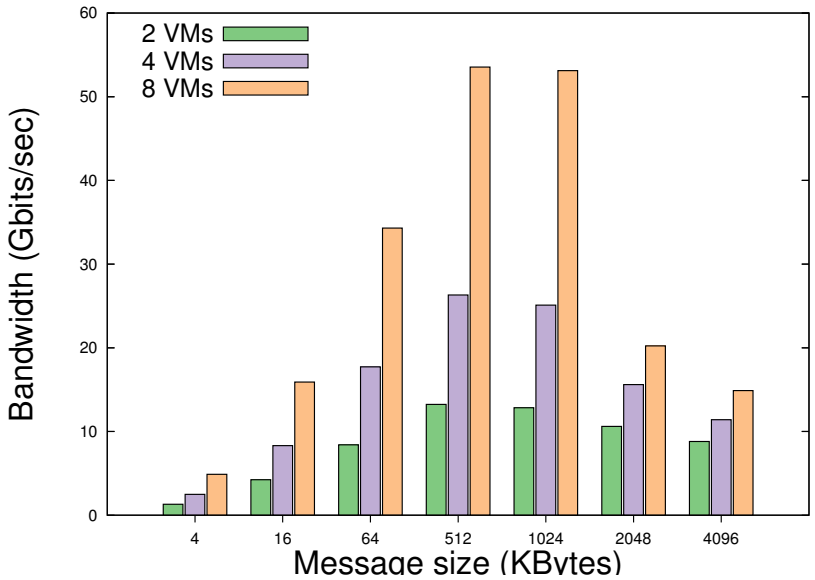
Μια βασική παράμετρος αξιολόγησης μηχανισμών σε εικονικά περιβάλλοντα είναι η κλιμακωσιμότητά τους, δηλαδή η δυνατότητα παράλληλης εκτέλεσης του μηχανισμού σε πολλά εικονικά μηχανήματα.

Για την εκτίμηση της κλιμακωσιμότητας του YASMIN, χρησιμοποιούνται 8 μονοπύρηνα εικονικά μηχανήματα τα οποία παράλληλα ανταλλάσσουν δεδομένα κατά ζεύγη (το EM0 με το EM1, το EM2 με το EM3 κ.ο.κ.). Κάθε EM εκτελείται σε ένα πυρήνα και μοιράζεται τη L2 cache με την απομακρυσμένη EM. Για παράδειγμα, όταν η EM1 και η EM2 ανταλλάσσουν δεδομένα, η EM1 εκτελείται στον πυρήνα CPU0 και η EM2 στον CPU1, οι οποίοι μοιράζονται τη L2 cache μνήμη μεγέθους 4MBytes. Η παραπάνω τοπολογία φαίνεται στο Σχ. 16.



Σχήμα 16: Απεικόνιση της διάταξης του συστήματος κατά την εκτέλεση των πειραμάτων κλιμακωσιμότητας. Η L1 cache μνήμη έχει μέγεθος 32KB και η L2 μέγεθος 4MB

Τα αποτελέσματα του παραπάνω πειράματος φαίνονται στο Σχήμα 17. Τα αθροιστικό εύρος ζώνης, δηλαδή το άθροισμα του εύρους ζώνης κάθε ζεύγους εικονικών μηχανών, αυξάνεται ανάλογα με τον αριθμό των ζεύγων EM που ανταλλάσσουν δεδομένα. Για παράδειγμα, δυο VMs ανταλλάσσουν μήνυμα μεγέθους 512KB με ταχύτητα 13.2Gbps, ενώ 8 VMs ανταλλάσσουν μηνύματα ίδιου μεγέθους με τετραπλάσια αθροιστική ταχύτητα (53Gbps ή 6625MBytes/sec). Σε σύγκριση με τα παραπάνω, η ταχύτητα του διαύλου μνήμης του συστήματος φτάνει στα 3784MBytes/sec.



Σχήμα 17: Αθροιστικό εύρος ζώνης της παράλληλης ανταλλαγής μηνυμάτων μεταξύ 4 ζεύγων VM σε σχέση με το μέγεθος μηνύματος. Για μηνύματα μέχρι 1MB, ο διπλασιασμός των παράλληλων EM προκαλεί διπλασιασμό του αθροιστικού bandwidth.

Στο Σχήμα 17 παρατηρείται ότι το αθροιστικό bandwidth του πειράματος σημειώνει σημαντική μείωση για μεγέθη μηνυμάτων μετά τα 2MB. Αυτό συμβαίνει διότι για τέτοια μεγέθη, επιδρούν καθοριστικά φαινόμενα που σχετίζονται με τη χρήση μνημών cache. Τα EM που ανταλλάσσουν δεδομένα μοιράζονται 4MB L2 cache μνήμες, οπότε όσο το μέγεθος μηνύματος πλησιάζει τη συγκεκριμένη τιμή, τόσο μειώνεται η επαναχρησιμοποίηση δεδομένων και αυτά πρέπει να μεταφερθούν από την κύρια μνήμη του συστήματος.

6 Σύνοψη και Συμπεράσματα

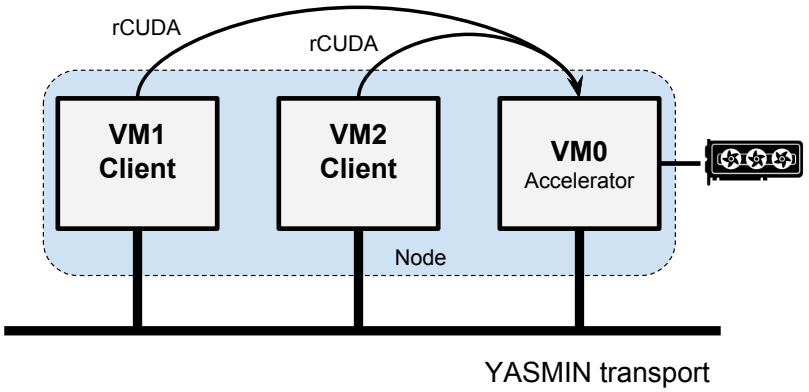
Ο μηχανισμός YASMIN βασίζεται στην κοινή χρήση σελίδων μνήμης μεταξύ των εικονικών μηχανών που βρίσκονται σε επικοινωνία και εγκαθιστά ένα αξιόπιστο κανάλι ανταλλαγής μηνυμάτων. Ο συγκεκριμένος σχεδιασμός μπορεί να εφαρμοστεί σε κάθε hypervisor που προσφέρει τεχνικές paravirtualization αλλά στην παρούσα διπλωματική εργασία υλοποιείται στον Xen hypervisor.

Παρόλο που η ιδέα για το διαμοιρασμό σελίδων μνήμης μεταξύ των ΕΜ έχει εξεταστεί και σε προηγούμενες εργασίες, όπως περιγράφονται στην Ενότητα 3, ο συγκεκριμένος μηχανισμός επιτυγχάνει βελτίωση της επικοινωνίας εξασφαλίζοντας διαφάνεια και επαναχρησιμοποίηση κώδικα. Αυτό είναι ένα πολύ βασικό πλεονέκτημα σε σχέση με τις προηγούμενες υλοποιήσεις που απαιτούν επανασχεδιασμό και επανεγγραφή κώδικα και δεν εξασφαλίζουν συμβατότητα με πρόσφατες εκδόσεις του πυρήνα του Linux.

Ο YASMIN βελτιώνει το χρόνο απόκρισης (latency) της επικοινωνίας κατά 64% και το throughput ανταλλαγής μηνυμάτων (throughput) κατά 4.4 φορές, σε σύγκριση με το προεπιλεγμένο μοντέλο netback/netfront. Η ανταλλαγή δεδομένων από τον αποστολέα στον παραλήπτη περιλαμβάνει μόνο δυο λειτουργίες αντιγραφής, η πρώτη από το χώρο χρήστη του αποστολέα προς τον διαμοιραζόμενο δακτύλιο στο χώρο πυρήνα και η δεύτερη από το δακτύλιο στο χώρο χρήστη του παραλήπτη. Η ανταλλαγή δεδομένων δεν απαιτεί τη διαμεσολάβηση της εικονικής μηχανής ελέγχου dom0, παρά μόνο κατά την εγκατάσταση του control channel μεταξύ δυο ΕΜ. Επιπλέον, η υλοποίηση YASMIN ανταποκρίνεται επιτυχώς στις ανάγκες κλιμακωσιμότητας (scaling) των εφαρμογών, όπως φαίνεται και στο Σχήμα 17.

Από την παρούσα διπλωματική εργασία προκύπτει το συμπέρασμα ότι σε ένα μεγάλο εύρος εφαρμογών όπου ο χρόνος επικοινωνίας είναι ένας κρίσιμος παράγοντας απόδοσης, η κατάλληλη εκτέλεσή τους σε εικονικές μηχανές που βρίσκονται στο ίδιο φυσικό μηχάνημα έχει καθοριστική επίδραση στην απόδοση. Μπορούν να αξιοποιηθούν τεχνικές βελτιστοποίησης της επικοινωνίας τους κάτι το οποίο καθιστά τον YASMIN ένα επωφελή μηχανισμό επικοινωνίας εφαρμογών σε εικονικά περιβάλλοντα. Για παράδειγμα, δίνεται η δυνατότητα σύνδεσης ενός επιταχυντή (π.χ GPU) σε ένα κόμβο και αποδοτικότερης αξιοποίησής του από όλες τις εικονικές μηχανές του κόμβου για την εκτέλεση πυρήνων CUDA μέσω του μηχανισμού rCUDA [44], όπως φαίνεται στο Σχ.18.

Ο YASMIN είναι μηχανισμός ανοιχτού κώδικα και βρίσκεται στο
`https://github.com/mrozis/YASMIN.git`.



Σχήμα 18: Τα VMs εκτελούν πυρήνες CUDA στον συνδεδεμένο επιταχυντή, χρησιμοποιώντας τον μηχανισμό rCUDA και το επίπεδο μεταφοράς YASMIN. Το server VM αναλαμβάνει την επικοινωνία με τη συσκευή GPU.

7 Περαιτέρω υλοποίηση

Ως μελλοντική δουλειά, για την περαιτέρω μελέτη και ανάπτυξη του μηχανισμού YASMIN, αρχικά προτείνεται η καλύτερη αξιολόγηση των φαινομένων που προκαλούνται εξαιτίας του κορεσμού της λανθάνουσας μνήμης (cache memory), όπως παρατηρούνται στα Σχήματα 14 και 17 και ερμηνεύτηκαν παραπάνω. Η αποστολή μηνυμάτων κοντά στο μέγεθος της L2 cache (4MB) προκαλεί την αύξηση των cache misses, τα οποία επιδρούν δραματικά στο χρόνο αντιγραφής δεδομένων από/προς τον επεξεργαστή. Προς αυτή την κατεύθυνση προτείνεται η αξιοποίηση των virtual Performance Counters σε συνδυασμό με τον Xen hypervisor για την καλύτερη τεκμηρίωση των cache effects.

Επιπλέον, προτείνεται η βελτίωση του μηχανισμού ενημέρωσης του αρχείου `hosts`. Προς το παρόν το αρχείο αυτό, στο οποίο αντιστοιχίζονται οι διευθύνσεις IPv4 με local domain IDs, ενημερώνεται χειροκίνητα από τον διαχειριστή του κόμβου. Προτείνεται λοιπόν η επέκταση του μηχανισμού YASMIN ώστε η ενημέρωση του αρχείου αυτού να γίνεται αυτοματοποιημένα και να ενημερώνεται on-line ώστε να μπορεί να ανταποκρίνεται στις δυναμικές μεταναστεύσεις εικονικών μηχανών. Μια πιθανή προσθήκη θα ήταν η υλοποίηση ενός daemon προγράμματος το οποίο θα λειτουργεί στην εικονική μηχανή ελέγχου, το οποίο θα παρακολουθεί για την εκκίνηση, απενεργοποίηση ή μετανάστευση των εικονικών μηχανών και θα ενημερώνει κατάλληλα το αρχείο hosts που βρίσκεται σε κάθε EM.

Τέλος, προτείνεται η σύγκριση του μηχανισμού YASMIN με τις υλοποιήσεις που αναφέρθηκαν στην Ενότητα 3, για την καλύτερη αξιολόγηση της υλοποίησης και την τεκμηρίωση των πλεονεκτημάτων που προσφέρει σε σύγκριση με τις προηγούμενες.

8 Βιβλιογραφία

Αναφορές

[1] Laszlo B. Kish, End of Moore's law: thermal (noise) death of integration in micro and nano electronics, Texas A&M University, Department of Electrical Engineering, College Station, TX 77843-3128, USA

[2] What is GPU Computing?, `http://www.nvidia.com/object/what-is-gpu-computing.html`

[3] CUDA Parallel Computing Platform, `http://www.nvidia.com/object/cuda_home_new.html`

[4] OpenCL, The open standard for parallel programming of heterogeneous systems, `https://www.khronos.org/opencl/`

[5] Accelerating High-Performance Computing With FPGAs, White Paper, Altera Corporation.

[6] Shehabi, A., Smith, S.J., Horner, N., Azevedo, I., Brown, R., Koomey, J., Masanet, E., Sartor, D., Herrlin, M., Lintner, W. 2016. United States Data Center Energy Usage Report. Lawrence Berkeley National Laboratory, Berkeley, California. LBNL-1005775.

[7] Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action, `https://portal.etsi.org/nfv/nfv_white_paper.pdf`

[8] John DiGiglio, Davide Ricci, High Performance, Open Standard Virtualization with NFV and SDN, A Joint Hardware and Software Platform for Next-Generation NFV and SDN Deployments.

[9] Network Function Virtualisation (FV); Use Cases, ETSI GS NFV 001, `http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf`

[10] Apache Hadoop, `https://wiki.apache.org/hadoop`

[11] Λίστα με ιδρύματα που χρησιμοποιούν το Apache Hadoop για εκπαιδευτικούς και εμπορικούς σκοπούς , `https://wiki.apache.org/hadoop/PoweredBy`, `https://wiki.apache.org/hadoop/DistributionsandCommercialSupport`

[12] Hui Kang, Yao Chen, Jennifer Wong, Radu Sion, Jason Wu. Enhancement of Xen's Scheduler for MapReduce Workloads, HPDC'11, June 8–11, 2011, San Jose, California, USA.

[13] Jongse Park, Daewoo Lee, Bokyeong Kim, Jaehyuk Huh, Seungryoul Maeng. Locality-Aware Dynamic VM Reconfiguration on MapReduce Clouds, HPDC'12, June 18–22, 2012, Delft, The Netherlands.

[14] Shadi Ibrahim, Hai Jin, Lu Lu, Bingsheng He, Song Wu. Adaptive Disk I/O Scheduling for MapReduce in Virtualized Environment, 2011 International Conference on Parallel Processing.

[15] Top500 SuperComputers, https://www.top500.org.

[16] The Vital Importance of High-Performance-Computing to U.S Competitiveness, http://www2.itif.org/2016-high-performance-computing.pdf.

[17] Amid spying scandal, billion-dollar NSA data center may secretly open, https://www.rt.com/usa/nsa-data-open-utah-408/

[18] The computers that run the stock market, http://money.cnn.com/2013/07/08/investing/stock-market-citadel/

[19] Nicole Hemsoth, "The Supercomputing Strategy That Makes Airbus Soar,The Next Platform", July 22, 2015, https://www.nextplatform.com/2015/07/22/the-supercomputing-strategy-that-makes-airbus-soar/.

[20] National Strategic Computing Initiative, https://www.nitrd.gov/nsci/.

[21] The Vital Importance of High-Performance-Computing to U.S Competitiveness, http://www2.itif.org/2016-high-performance-computing.pdf.

[22] Marx, Karl. Capital: A Critique of Political Economy, Vol. I. The Process of Capitalist Production. Frederick Engels, Ernest Untermann, eds. Samuel Moore, Edward Aveling, trans. 1906.

[23] Hypervisor, Virtual Machine Monitor (VMM), https://en.wikipedia.org/wiki/Hypervisor

[24] Xen Project Hypervisor, https://wiki.xen.org/wiki/Xen_Project_Software_Overview.

[25] QEMU - The FAST! processor emulator, www.qemu.org.

[26] Open Systems Inteconnection (OSI) model.

[27] François Diakhaté, Marc Pérache, Raymond Namyst, Hervé Jourdren. Efficient shared memory message passing for inter-VM communications. 2008.

[28] Kernel Virtual Machine, https://www.linux-kvm.org/page/Main_Page.

[29] W. Huang, M. J. Koop, Q. Gao, and D. 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.

[30] X. Zhang, S. McIntosh, P. Rohatgi, and J. Griffin. Xensocket: A high-throughput interdomain transport for virtual machines. In R. Cerqueira and R. Campbell, editors, Middleware 2007, Lecture Notes in Computer Science. 2007.

[31] PV Calls: a new paravirtualized protocol for POSIX syscalls.

[32] J. Wang, K.-L. Wright, and K. Gopalan. XenLoop: a transparent high performance inter-vm network loopback. In HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing.

[33] Netfilter, `https://www.netfilter.org/`.

[34] Anastassios Nanos, Stefanos Gerangelos, Ioanna Alifieraki and Nectarios Koziris. V4VSockets: low-overhead intra-node communication in Xen. In CloudDP'15, April 21-24, 2015, Bordeaux, France.

[35] V4V implementation, `http://lists.xen.org/archives/html/xen-devel/2013-05/msg02711.html`.

[36] Vchan Xen Library, `https://github.com/mirage/xen/tree/master/tools/libvchan`

[37] VMware vSockets, `https://pubs.vmware.com/vsphere-65/index.jsp\#com.vmware.vmci.pg.doc/vsockAbout.3.2.html\#1023121`.

[38] Virtual Machine Communication Interface, `https://pubs.vmware.com/vmci-sdk/`.

[39] Virtio - IO Virtualization in KVM, `http://www.linux-kvm.org/page/Virtio`

[40] NetPIPE – Network Protocol Independent Performance Evaluator, `https://linux.die.net/man/1/netpipe`.

[41] iPerf Benchmark, `https://iperf.fr/`.

[42] netperf - a network performance benchmark, `https://linux.die.net/man/1/netperf`.

[43] McCalpin, John D., 1995: "Memory Bandwidth and Machine Balance in Current High Performance Computers", IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, December 1995

[44] José Duato, Francisco D. Igual, Rafael Mayo, Antonio J. Peña, Enrique S. Quintana-Ortí, and Federico Silla. An efficient implementation of GPU virtualization in high performance clusters. In Euro-Par 2009, Parallel Processing – Workshops, volume 6043 of Lecture Notes in Computer Science, pages 385–394. Springer-Verlag, 2010.