REPUBLIKA E SHQIPËRISË

UNIVERSITETI POLITEKNIK I TIRANËS

FAKULTETI I TEKNOLOGJISË SË INFORMACIONIT

DEPARTAMENTI I INXHINIERISË INFORMATIKE

IGLI TAFA

Për marrjen e gradës

“ Doktor”

Në Inxhinieri Informatike

DISERTACION

KRIJIMI I NJË TEKNIKE EFIÇENTE PËR RRITJEN E PERFORMANCES `LIVE MIGRATION` NË MAKINAT VIRTUALE, IMPLEMENTUAR MBI NJË SISTEM ME BAZË TË DHËNASH `ORACLE‘”

Udhëheqës shkencor:

Prof. Dr. Aleksandër Xhuvani

TIRANË, 2013

“Krijimi i një teknike efiçente për rritjen e performances ‘live migration‘ në makinat virtuale, implementuar mbi një sistem me bazë të dhënash ‘Oracle‘”

i paraqitur në Universitetin Politeknik të Tiranës

për marrjen e gradës

“ Doktor”

në

Inxhinieri Informatike

nga
Z. Igli Tafa
2013

Disertacioni i shkruar nga

Z. Igli Tafa

Master Shkencor, Universiteti Politeknik i Tiranës, Shqipëri, 2007

DIEE, Universiteti Politeknik i Tiranës, Shqipëri, 2002

I aprovuar nga

Prof.Asoc. Elinda Meçe Kajo	Kryetar i Komisionit të Disertacionit të Doktoraturës
Prof.Asoc. Genti Daci	Anëtar i Komisionit të Disertacionit të Doktoraturës
Prof.Dr. Dhimitër Tole	Anëtar i Komisionit të Disertacionit të Doktoraturës
Prof.Dr. Kozeta Sevrani	Anëtar i Komisionit të Disertacionit të Doktoraturës
Prof.Dr Luan Karçanaj	Anëtar i Komisionit të Disertacionit të Doktoraturës

I pranuar nga

Prof.Dr. Rozeta Miho Mitrushi	Dekan , Fakulteti i Teknologjisë së Informacionit

ABSTRAKT:

Teknikat e Migrimit janë një ndër sfidat më të mëdha të teknologjisë moderne. Migrim është dërgimi i proçeseve nga një makinë fizike në një tjetër. Nëse ky migrim ndodh në kohë reale merr emrin Migrim Live. Ajo që është arritur në këtë punim është ngritja e një metode në hapsirën përdorues e cila do të bëjë të mundur përmirësimin e teknikës së migrimit Live, sidomos në rastet kur nuk ka një lajmërim për fikjen e makinës përkatëse, si p.sh. rasti i një katastrofe, zjarri apo termeti. Këtë rast e kemi quajtur "Migrim Pa Lajmërim". Metoda e ngritur ul shumë ndjeshëm numrin e faqeve të modifikuara të pa shkruar ende në Disk. Këto faqe pranojmë se pastrohen në të. Me uljen e faqeve të modifikueshme bëhet e mundur dhe ulja e cikleve të përsëritura pre-copy nga burimi në destinacion. Faqet e modifikuara kontrollohen nga një thread i metodës përkatëse, duke verifikuar bitin e modifikimit, të quajtur "Dirty Bit" në rekordet e Tabelës së Faqeve të lokalizuar në mënyrë inverse në Memorjen Fizike. Nëse ky bit ka vlerën 1 atëhere faqja përkatëse pastrohet në disk. Rendi i faqeve të pa pastruara është me dy shifra, shumë herë më pak se rasti klasik. Me anë të kësaj teknike në migrimin live do të ulet së tepërmi, koha totale e migrimit nga një Qendër të Dhënash në një tjetër, koha e bllokimit apo trafiku që shoqëron faqet që po ekzekutohen në makinën burim. Kjo do të bëjë që një përdorues i cili po ekzekuton nje aplikacion, psh një lojë grafike në kompjuterin e tij, të mos ndjejë asnjë shqetësim pavarësisht se qendra e të dhënave të Serverit që mban aplikacionin është e dëmtuar.

Një nga avantazhet kryesore të kësaj metode është se mund të komunikojë me çdo tip Hypervisori, pasi nuk ngrihet në Hapsirën Kernel. Tjetër avantazh është mundësia për tu zgjeruar, kjo falë instalimit të saj në gjuhën C në hapsirën përdorues. Metoda në kete tezë quhet Live Improve.

Qëllimi i kësaj metode është mundësia për tu implementuar si pjesë e një aplikacioni në Sisteme Operative të ndryshme, Windows, Linux, Mac etj. Një qëllim tjetër i saj është ulja e mundësisë së humbjeve të dokumentave gjatë punës, p.sh shkrimi i një dokumenti në MS_Word, si pasojë e një gabimi të pritur apo të pa pritur (fatal error).

Metoda Live Improve, përmirësohet vazhdimisht duke implementuar në të teknika shtesë që kërkojnë në disa raste edhe modifikimin e kernelit, si psh krijimi i tabelave Hash në nivel përmbajtje të faqes së memorjes, apo ndarja e faqeve aktive në lista aktive dhe pasive etj. Është vënë re që me anë të këtyre metodave të kombinuara parametrat e migrimit përmirësohen ndjeshëm.

Fjalët Kyçe: *Live Improve, Bitet e Modifikimit, Metoda Pre-Copy, Migrimi*

TABELA E PËRMBAJTJES

LISTA E FIGURAVE

LISTA E TABELAVE

FALENDERIME

Fillimisht dua të falenderoj përzemërsisht udhëheqësin tim të doktoraturës, Prof.Dr Aleksander Xhuvani. Ndihma e tij e çmuar dhe e pakursyer, më ka orientuar në kërkimin tim shkencor, në thellimin e njohurive përkatëse, në suksesin e prezantimeve të punimeve të mia nëpër konferenca kombëtare dhe ndërkombëtare, në botimin e artikujve të ndryshëm në revista kudo nëpër botë dhe së fundmi në organizimin, strukturimin dhe mbylljen me sukses të këtij punimi.

Falenderoj përzemërsisht dekanen e Fakultetit, Prof.Dr Rozeta Miho, për mbeshtetjen e vazhdueshme të saj ndaj të gjitha sfidave dhe problemeve gjatë ciklit tre vjeçar të doktoraturës.

Falenderoj përzemërsisht Shefen e Departamentit të Inxhinjerisë Informatike, Prof.Asoc Elinda Kajo, për ndihmën e madhe që më ka dhënë, ndaj të gjitha problemeve që shfaqeshin në punën time si kërkues shkencorë. Gjithashtu dua të nënvizoj bashkëpunimin e saj, në shumë artikuj dhe konferenca të përbashkëta kombëtare dhe ndërkombëtare duke më dhënë një ndihmesë të madhe në përballimin me sukses të tyre.

Falenderoj përzemërsisht Rektorin e Fakultetit, Akademik Jorgaq Kaçani si një person gjithmonë përkrahës dhe mbështetës për çdo situatë të vështirë të shfaqur gjatë ciklit të Doktoraturës

Falenderoj të gjithë koleget e mi, me të cilët kam shkëmbyer eksperienca të cilat më kane vlejtur shumë gjatë punimit tim

Dhe në fund dua të falenderoj të gjithë studentët e Fakultetit të Teknologjisë së Informacionit.

…Faleminderit

KAPITULLI I PARË

HYRJE

Virtualizimi është sot një nga çështjet më të nxehta në teknologjinë e informacionit. Rritja e shpejtësisë dhe e aftësive të procesorëve x86 në ditët e sotme, kanë bërë të mundur realizimin e virtualizimit për produkte hardware. Nga ana tjeter virtualizimi siguron një mënyrë të këndshme të shfrytëzimit maksimal të këtij hardware-i.

Teknologjia e virtualizimit ka avantazhet dhe disavantazhet e saj. Një avantazh është se virtualizimi është një teknologji e fuqishme që në të vërtetë mund të thjeshtojë infrastrukturën kompjuterike dhe të na ndihmojë të marrim më të mirën nga procesorët më të shpejtë, rrjeti dhe teknologjitë e ruajtjes së të dhënave. Gjithashtu kjo teknologji jep shumë avantazhe lidhur me koston, konsumimin e burimeve dhe energjisë, tolerancën ndaj gabimeve, izolimin ndaj sulmeve të ndryshme, ndihmon në shfrytëzimin sa më të mirë të hardware-it ekzistues, kursen kohën dhe punën e sistemit administrues, rrit disponueshmërinë e sistemit dhe të shërbimeve etj. E keqja është se si çdo gjë në botën reale, për implementimin e suksesshëm, vënien në punë, dhe mbështetjen e një infrastrukture të re IT të bazuar në virtualizim, kërkohen të njëjtat nivele planifikimi dhe projektimi të sistemit, sikurse edhe çdo infrastrukturë bazë. Virtualizimi shpesh shkakton problem në performancë si në burimet që menaxhojnë hypervisor-in ashtu edhe në menaxhimin e makinave virtuale krahasuar me ato fizike.

Virtualizimi i sistemit operativ është përdorimi i software-it për të lejuar një pjesë të hardware-it të ekzekutojë shumë sisteme operative në të njëjtën kohë. Kjo lloj teknologjie i pati fillesat e saj në mainframe disa dekada më parë, duke i lejuar

administratorët të shmangin shpërdorimin e fuqisë procesuese tepër të kushtueshme. IBM ishte kompania e parë që vuri në përdorim virtualizimin në mjediset e serverëve. VMM (Virtual Machine Monitor) u prezantua në 1969 kur IBM donte të ndante burimet në mainframet e tyre për qëllime multitasking. Me kalimin e viteve hardware është bërë më pak i kushtueshëm në performancë, gjë që e bën të arsyeshme përdorimin e virtualizimit në dobi të kompjuterëve të përdoruesve.

Në 2005 virtualizimi u përshtat shumë më shpejtë nga çfarë imagjinohej, duke përfshirë këtu edhe ekspertët. Janë tre zona kryesore të IT ku virtualizimi po kryeson, virtualizimi i rrjetit, virtualizimi i ruajtjes së të dhënave dhe virtualizimi i serverit. Gjithsesi teknologjia komode e virtualizimit është në hapat e saj të parë. Derisa teknologji si Xen si dhe mjetet e tjera shoqëruese maturohen, aplikacionet potenciale të virtualizimit do të vazhdojnë të zgjerohen. Përqasjet që ofron Xen drejt virtualizimit nuk i ofron asnjë zgjedhje tjetër virtualizimi.

Virualizimi mund të shihet si një pjesë e tendencës së përgjithshme të iniciativës IT që përfshin *automatic computing*, një skenar në të cilin mjedisi IT do të jetë i aftë të menaxhojë veten duke u bazuar në aktivitetin e perceptuar, dhe dobinë e kompjuterizimit, ku fuqia kompjuterike shihet si një dobi për të cilën klientët paguajnë për aq sa ju nevojitet. Qëllimi më i zakonshëm i virtualizimit është të qendërzojë detyrat administrative duke përmirësuar shkallëzueshmërinë dhe ngarkesën e punës.

Virtualizimi hardware i kompjuterit është virtualizimi i kompjuterit ose i sistemit operativ, i cili fsheh karakteristikat fiziket të një platforme kompjuterike nga përdoruesi dhe në vend të kësaj tregon një platforme kompjuterike abstrakte. Në fillimet e tij software-i që kontrollonte virtualizimin u quajt “program kontrolli”, por në ditët e sotme ka marrë emrin “hypervisor” ose “monitor i makinës virtuale ”. Hypervisor-i është administrator dhe menaxher i burimeve të përdorura nga makinat virtual. Hypervisor-i vendoset në pjesën e sipërme të hardware dhe kjo quhet virtualizimi i plotë, ose mund të vendoset në pjesën e sipërme të sistemit operativ dhe kjo quhet virtualizimi OS.

Kompjuterët virtual që ekzekutohen në një kompjuter fizik janë tërësisht të pavarur nga njëri tjetri, dhe komunikojnë me njëri tjetrin duke përdorur protokollin IP. Avantazhi këtu qëndron në faktin se nëse njëri prej kompjuterëve sulmohet, të tjerët ngelin të pa prekur nga kjo gjë. Administratorët e sistemit reduktojnë rrezikun e prekjes së shërbimeve të tjera nga njëra duke ekzekutuar një shërbim të vetëm për makinë fizike. Nëse bëhet kjo me virtualizim prezanton një kosto të konsiderueshme në varësi të kërkesës për më shumë hardware dhe kosto e mirëmbajtjes, por duke përdorur virtualizimin shërbimet kryhen duke u ekzekutuar së bashku në një host fizik të vetëm. Hosti fizik është sërisht single point of failure, por me një sistem operativ minimal që mund të ekzekutojë makina virtuale është më i besueshmëm se sistemet operative me instalime komplekse.

Virtualizimi mund të realizohet në shumë mënyra, por më e shpeshta është të dyfishoj aplikacionet nga hardware duke shtuar një shtresë virtualizimi midis hardware-it dhe sistemit operativ.

Në punimin tim është realizuar nje software i cili është implementuar mbi Hypervisor në shtresën e përdoruesit. Ky software ka si qëllim të ulë disa parametra gjatë migrimit live të një aplikacioni nga një makinë fizike në një makinë tjetër brenda një rrjeti lokal dhe midis rrjetave të ndryshëm. Ajo që përfitohet nga implementimi i këtij software të quajtur *Live Improve*, jane koha totale e migrimit te aplikacionit nga një makinë fizike në tjetrin, koha e bllokimit dhe trafiku gjatë migrimit të proçeseve të aplikacionit.Një sërë metodash janë ngritur për të përmirësuar cilësinë e parametrave të mësipërm gjatë migrimit. Këto metoda janë futur të ndërthurura me software-in tim. Dy nga metodat më efikase janë ngritja e funksioneve dhe tabelave hash për të bërë kërkimin e fjalës në memorjen fizike dhe në disk, si edhe përcaktimi i bashkësisë së punës efiçente për të reduktuar faqet e modifikuara gjate përsëritjes në proçesin e migrimit.

Si Hypervisor janë marrë në studim tipet e mëposhtëm:

Xen, Kernel Virtual Machine, OpenVZ.

Këto tipe Hypervisor-ash jane trajtuar si Full Virtual Machine por edhe si teknika të Paravirtualizimit.

Si ambient për ruajtjen e të dhënave është përdorur Oracle Data Base.

Struktura e tezës është si më poshtë:

Në kapitullin e dytë janë përcaktuar konceptet teorike të Virtualizimit dhe llojet e tyre. Në kapitullin e tretë trajtohen avantazhet dhe disavantazhet e Virtualizimit. Më pas në kapitullin e katërt jepet një trajtim për tipe të ndryshme Hypervisor-ash si dhe diferencat mes tyre. Si shembull i konceptimit te Hypervisor-at është marrë Xen. Në kapitullin e pestë jepen disa karakteristika bazë të migrimit të cilat i kam përdorur në punimin tim. Gjithashtu një vend të vecantë në këtë kapitull zë edhe trajtimi i Sistemit të Database-it Oracle, meqënëse ky sistem është marrë si model në punimin tim. Në kapitullin e gjashtë vijohet me përshkrimin dhe analizën teorike te punimit. Po në këtë kapitull vijohet me pjesën eksperimetale, nxjerrjen e rezultateve dhe interpretimet e tyre. Në kapitullin e shtatë vijohet me konkluzionet. Në kapitullin e tetë kalohet te shtojca dhe kodet përkatëse. Në fund janë Referencat.

KAPITULLI I DYTË

LLOJET E VIRTUALIZIMIT

2.1 Çfarë është virtualizimi?

Virtualizimi [1] është një ndarje logjike e kërkesës për disa shërbime nga burimet fizike që e sigurojnë shërbimin. Praktikisht, virtualizimi siguron aftësinë për të ekzekutuar aplikacione, sisteme operative, ose sisteme shërbimesh në një mjedis të veçantë logjik të pavarur nga sistemi fizik i kompjuterit. Sigurisht, që të gjitha këto duhet të ekzekutohen në një kompjuter të caktuar në një kohë të caktuar, por virtualizimi siguron një nivel logjik abstraksioni që liron aplikacionet, sistemet e shërbimeve, madje dhe sistemin operativ nga të qenët të lidhura në një pjesë specifike të hardware-it. Fokusimi i virtualizimit në mjediset operative logjike në vend të atyre fizike i bën aplikacionet, shërbimet dhe instancat e një sistemi operativ të transferueshëm nëpër sisteme fizike të ndryshme kompjuterësh.

Platforma e virtualizimit performohet në një platformë hardware të caktuar nga *hosti software* (ose programi i kontrollit), e cila krijon një kompjuter të simuluar, një makinë virtuale për *guest software-in*. Guest software nuk është i limituar për përdoruesit e aplikacioneve; shumë hoste lejojnë ekzekutimin e komplet sistemit operativ. Ai ekzekutohet njësoj sikur ti përkiste direkt pjesës fizike hardware.

Shembulli klasik i virtualizimit të cilin e njohin shumë njerëz është memoria virtuale, e cila bën të mundur që një sistem kompjuterik të ketë më shumë memorie në dukje sesa ka të instaluar fizikisht. Memoria virtuale është një teknikë menaxhimi e memories që bën që një sistem operativ të përdorë segmente të ndara të memories si një hapësirë memorieje të vetme e të vazhdueshme. Memoria virtuale është implementuar tradicionalisht në një sistem operativ nëpërmjet “paging”, që bën që sistemi operativ të

përdorë një file ose pjesë të dedikuara të disa pajisjeve të ruajtjes se të dhënave për të ruajtur faqe te memories te cilat nuk janë përdorur së fundmi. E njohur edhe si “paging file” ose “swap space”, sistemi mund të transferojë shumë shpejt faqe drejt dhe prej kësaj zone duke qenë se sistemi operativ ose një aplikim ekzekutues kërkojnë të aksesojnë përmbajtjen e këtyre faqeve. Sistemet operative si UNIX (duke përfshirë Linux, sistemet operative BSD, dhe Mac OSX) edhe Microsoft Windows përdorin disa forma të memories virtuale për te bërë të mundur që sistemi operativ dhe aplikacionet të aksesojnë më shumë të dhëna se sa mund të vendosen në memorien fizike.

Ka shumë tipa të ndryshëm të virtualizimit, të gjitha rrotullohen rreth idesë bazë të sigurimit të aksesit logjik në burimet fizike. Sot, virtualizimi është zakonisht i hasur në rrjete, sistemet e ruajtjes së të dhënave, dhe proceset e serverave, në nivel të sistemit operativ dhe atë të makinës. Xen mbështet vitrualizimin në makina-nivel duke përdorur një varietet teknikash të zgjuara dhe të fuqishme.

Për grupet e marketingut të koorporatave është joshëse të abuzojnë me termin “virtualizim” që të përfitojnë më shumë vëmendje për produktet apo teknologjitë e tyre. Përdorimi i termit “virtualizim” në literaturën e marketingut të sotshëm rivalizon terma si *“Interne*t” dhe *“network-enabled”* në vitet 1990.

2.2 Virtualizimi në nivel aplikacioni

Termi “virtualizim në nivel aplikacioni ”, [2], [3] përshkruan procesin e kompilimit të aplikimit në kode byte-sh (byte code) të pavarur nga makina që mund të ekzekutohen në çdo sistem që siguron një makinë virtuale të përshtatshme si një mjedis ekzekutimi. Shembulli i mirë i këtij përafrimi të virtualizimit është kodi byte i prodhuar nga kompilatorët për gjuhën e programimit Java. Si çdo kod byte-sh, CIL siguron një bashkësi instruksionesh të pavarur nga platforma që mund të ekzekutohen në një mjedis që mbështet Framework .NET.

Një përdorim i vlefshëm i termit “virtualizim” është virtualizimi në nivel aplikacionesh sepse aplikacionet që kompilohen në kode byte-sh bëhen entitete logjike

që mund të ekzekutohen në sisteme fizike të ndryshëm me karakteristika të ndryshme, sisteme operative dhe gjithashtu arkitektura procesorësh.

2.3 Virtualizimi në nivel Desktop-i

Termi "virtualizim në nivel desktop-i"[3], përshkruan aftësinë për të shfaqur një desktop grafik nga një sistem kompjuterik në një tjetër ose ndryshe një pajisje ekrani e zgjuar. Ky term përdoret për të përshkruar software si *Virtual Network Computing* (VNC), *Microsoft Remote Desktop* dhe produkte shoqërues me terminal Server, serverat e terminaleve të Linux si projekti *Linux Terminal Server* (LTSP), *NX e NoMachine's*, dhe madje sisteme X Window dhe protokolli i tij i prezantimit XDMCP. Shumë menaxhues të window, gjithashtu sigurojnë një mbështetje të brendshme për desktop virtual të shumëfishtë që përdor përdoruesi për të shfaqur outputin e veprimtarive të ndryshme.

Si fillim në sistemet X Windows, desktop-ët virtualë u prezantuan në versionet e TWM të menaxherit Tom LeStrange, por sot janë të disponueshme në çdo menaxher *window*. Sistemi X Window suporton edhe virtualizimin në nivel desktop-i në nivel ekrani, duke bërë të mundur që menaxhuesit *window* të përdorin një zonë ekrani që është më e madhe se përmasa fizike e monitorit.

Virtualizimi në nivel desktop-i është më shumë një "lustër" e termit "virtualizim" sesa një shembull mbresëlënës i koncepteve të virtualizimit. Në të vërtetë bën një mbështetje grafike të çdo sistemi të mbështetur në një entitet logjik që mund të aksesohet dhe të përdoret ne sisteme fizike kompjuteri të ndryshëm, këtë e bën duke përdorur një software të shfaqjes standard klient/server. *Remote console* që po ekzekuton sistemi operativ në fakt ekzekutohen në një makinë të vetme fizike, thjesht shihen nga një vend tjetër.

2.4 Virtualizimi i rrjetit

Termi "virtualizim i rrjetit" [4] përshkruan aftësinë për t'ju referuar burimeve të rrjetit logjikisht sesa ti referohemi pajisjeve fizike specifike të rrjetit, konfigurimeve, ose

grupit të makinave të lidhura. Ka shumë nivele të ndryshme të virtualizimit të rrjetit, duke filluar nga një makinë e vetme, virtualizim i pajisjeve të rrjetit që bën që shumë makina virtuale të ndajnë një burim fizik të vetëm rrjeti, deri në konceptet *enterprise - level* si për shembull rrjete virtuale private dhe teknikat *edge-routing* dhe *enterprise-cor*e për krijimin e nënrrjeteve dhe segmentimin e rrjetave ekzistuese.

Xen mbështetet në virtualizimin e rrjetit përmes paketës *bridge-utils* për të bërë të mundur që makinat virtuale të duken sikur kanë adresa fizike unike (Media Access Control, ose adresa MAC) dhe adresat unike IP. Zgjedhje të tjera të virtualizimit të serverëve, si UML, përdorin pajisje rrjeti *Ethernet* (TAP) dhe *Linux virtual Point-to-Point* (TUN) për të siguruar akses të hapësirës së përdoruesit te rrjeti i host-it. Shumë *switche-e* rrjetash te avancuara dhe *routera* përdorin teknika si Routim-i Virtual dhe Forwarding (VRF), VRF-Lite, dhe multi-VRF për të veçuar trafikun e klientit në segmente të ndarë të rrjetit, dhe mbështet domain-et e routim-it virtual brenda një pjese te hardware-it të rrjetit.

2.5 Virtualizimi i serverit dhe makinave

Termi “Virtualizimi i serverëve” [5] dhe “virtualizimi i makinës” [6] përshkruan aftësinë për të ekzekutuar një makinë virtuale të plotë, duke përfshirë sistemin e saj operativ në një sistem operativ tjetër. Çdo makinë virtuale që ekzekutohet në sistemin operativ prind është logjikisht e veçuar, ka një akses tek disa ose të gjitha pjesët hardware në sistemin host, dhe mund të ekzekutojë aplikacionet e saj brenda mjedisit operativ të vet.

Virtualizimi i serverëve është një tip i teknologjisë së virtualizimit, e cila u vjen ndërmend njerëzve kur dëgjojnë termin “virtualizim”. Edhe pse termi “virtualizimi i makinës” nuk është aq i përhapur, është e dobishme për të identifikuar në mënyrë unike këtë tip virtualizimi, sepse thekson më qartë nivelin në të cilin kryhet virtualizimi. Virtualizimi i makinës është një teknikë e përdorur nga teknologjitë e virtualizimit si

KVM, Microsoft Virtual Server dhe Virtual PC, Workstation Paralele, Linux të hapësirës përdorues, Virtual Iron, VMware dhe Xen.

Në vorbullën marramendëse të termave që përfshijnë fjalën "virtual", virtualizimi i serverëve është zakonisht i ndryshëm nga termi "server virtual", i cili shpesh përdoret për të përshkruar aftësinë e sistemeve operative server si serverë e-mail dhe web për t'u shërbyer shumë domaine-ve të Internet-it, edhe teknikat në nivel sistemi të virtualizimit që përdoren për t'u shërbyer përdoruesve të ISP-ve makinën e tyre virtuale server .

Aspekti kyç i virtualizimit të serverëve ose virtualizimit të makinës është se makinat virtuale të ndryshme nuk ndajnë të njëjtin kernel dhe si rrjedhojë mund të ekzekutojnë sisteme operative të ndryshme. Kjo ndryshon nga virtualizimi në nivel sistemi, ku serverët virtual ndajnë të njëjtin kernel dhe sigurojnë një numër infrastrukturash unike, klientë, dhe mundësi biznesi. Disa nga këto janë:

Ekzekutimi i sofware-it të vjetër, në varësi të një produkti sofware që ekzekutohet vetëm në një version specifik të një sistemi operativ.

Testimi i sistemeve me anë të software-ve dhe e sigurimit të cilësisë së mjedisit në të cilat duhet testuar një software specifik në sisteme operative të ndryshme apo në versione të ndryshme.

Mjediset e zhvillimit në nivel të ulët, ku zhvilluesit mund të duan ose të kenë nevojë për të punuar me versione specifike të mjeteve, një sistem operativ kernel dhe një sistem operativ specifik. Virtualizimi i serverit e bën më të lehtë ekzekutimin e shumë sistemeve operative të ndryshme pa kërkuar hardware të dedikuar për secilin.

Teknologjia e virtualizimit të serverit dhe makinës punojnë në mënyra të ndryshme. Ndryshimet midis përqasjeve të virtualizimit të server-it ose makinës mund të jenë të mprehta por janë gjithmonë të rëndësishme në kapacitetin qe ato sigurojnë dhe kërkesat hardware dhe software për sistemin më poshtë.

Përqasjet me të përhapura të virtualizimit të serverëve dhe makinave sot janë:

Makina virtuale paralele: Disa sisteme fizike ose virtual organizohen në një makinë virtuale të vetme duke përdorur *software clustering* [7] si një Makine Virtuale Paralele (PVM). Cluster-i rezultues është i aftë të ekzekutoj CPU komplekse dhe llogaritje me të dhëna intensive në mënyrë bashkëpunuese.

Para-virtualizimi: Është metodë e përdorur nga XEN, Denali dhe VMware [8], [9]. Është i bazuar në hypervisor, por pajisjet nuk emulohen. Gjithashtu procesorët duhet të kenë karakteristika të njëjta. Në këtë mënyrë, komunikimi midis aplikacioneve në makinat virtuale dhe pajisjeve I/O realizohet përmes driver-ave virtual I/O që ngrihen në pjesën e sipërme të hypervisor-it. Kjo gjë rrit shpejtësinë e komunikimit midis aplikacioneve dhe pajisjeve I/O. Vijat e zeza në figurën 2.1 tregojnë hypervisor-in i cili fut një overhead krahasuar me virtualizimin e plotë pra ulet fleksibiliteti. GuestOS ekzekutohen në DomainU në paralel me HostOS qe po ekzekutohen në Domain0. Doamin0 është më i privilegjuari që do të thotë se sistemet operative që ekzekutohen në të mund të administrojnë sistemet e tjera operative. Paravirtualizimi mund të sigurojë përmirësime të performancës krahasuar me përafrime të tjera të virtualizimit të serverit dhe të makinës sepse modifikimet e sistemit operativ bëjnë të mundur që ai të komunikojë direkt me hypervisor-in, dhe kështu nuk pëson ndonjë nga mbingarkimet e shoqëruar me emulacionin e kërkuar për makinat e tjera të bazuara në hypervisor.

Paravirtualizimi ndryshon nga virtualizimi i plotë sepse bën të mundur që SO i virtualizuar të "shoh" si burimet reale ashtu dhe ato virtuale. Kjo gjë rrit performancën dukshëm dhe mban disi dhe fleksibilitetin. E meta është se guest kernel duhet të përmirësohet që të siguroj thirrje të reja për sistemin për shërbime të reja.

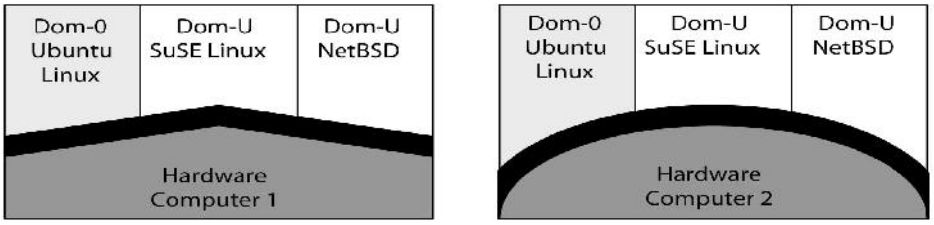


Figura 2.1- Diagrami logjik për para-virtualizimin

Virtualizimi i plotë: Nga [10] makinat virtuale janë shumë fleksibël. Është shumë i ngjashëm me paravirtualizimin, virtualizimi i plote gjithashtu përdor një hypervisor por ndërfut kod në hypervisor i cili emulon hardware-in e poshtëm kur është e nevojshme, duke u dhënë mundësinë sistemeve operative të pamodifikuara për t'u ekzekutuar në krye të hypervisor-it. Emulatori software krijon një shtresë që zbut ndryshimet në pjesën hardware të arkitekturave dhe bën të mundur ekzekutimin e së njëjtës makinë virtuale në hoste të ndryshëm me arkitektura të ndryshme pa krijuar probleme. Kjo gjë i jep fleksibilitetin e lëvizjes së gjithë makinave virtuale nga hosti në host në mënyrë shumë të lehtë, por rritet kosto e performancës për shkak të overhead-it nga shtresa emulatore. Virtualizimi i plotë është një model i përdorur nga serveri VMWare ESX. Diagrami logjike jepet në figurën 2.2

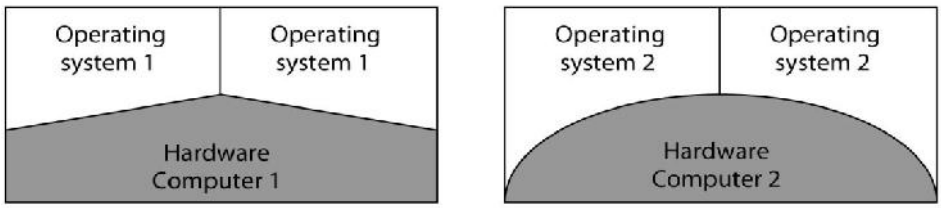


Figura 2.2- Diagrami logjik për virtualizimin e plotë

Virtualizimi i hardware: Është **s**humë i ngjashëm si me paravirtualizimin ashtu dhe me virtualizimin e plotë. Virtualizimi i hardware-it [11], përdor një hypervisor, por është i mundur vetëm në sistemet që sigurojnë një mbështetje hardware për virtualizimin. Sistemet e bazuar në hypervisor si Xen dhe Serveri VMWare ESX, dhe teknologjitë e virtualizimit ne nivel kerneli si KVM, mund të përfitojnë nga suportit i hardware-it për virtualizim që sigurohet në gjeneratën e fundit të Intel (Intel VT, njohur si Vanderpool) dhe procesorët AMD (AMD-V, njohur si *Pacifica*). Makinat virtuale në një mjedis hardware të virtualizimit mund të ekzekutojnë një sistem operativ të pamodifikuar sepse hypervisor-i mund të përdori suportin hardware për virtualizimin për veprimet e privilegjuara dhe të mbrojtura, si dhe kërkesa të aksesit hardware dhe të komunikojë dhe menaxhojë makinat virtuale.

Virtualizimi i bazuar në hypervisor është më popullori në teknikat e virtualizimit që janë në përdorim sot, duke përfshirë teknologjitë më të mira të virtualizimit të serverëve dhe makinave si sistemin operativ VM të IBM, serverin VMWare ESX, Paralels Workstation, produktet virtuale Iron dhe Xen.

2.6 Standartizimi i virtualizimit të serverit të Linux

Edhe pse Xen është një projekt me kod të hapur, mbajtja e Xen dhe njëkohësisht i versionit të përpunuar së fundmi të kernelit Linux është i vështirë. Rritja e popullaritetit të Xen i ka bërë shumë njerëz të shpresojnë për një përfshirje të drejtpërdrejtë të pjesëve të Xen në një linjë kryesore të burimeve të kernelit. Kernel i Linux-it është një përpjekje e hapur qëllimi i të cilës është një API e hapur dhe e përgjithshme, është i aftë të mbështesë më shumë se një zgjedhje virtualizimi të bazuar në hypervisor.

Në vitin 2006, VMWare propozoi një Ndërfaqe të Makinës Virtuale (VMI) [12], që do t'u jepte mundësinë teknologjive të virtualizimit të bazuara ne hypervisor për të përdorur një ndërfaqe të përbashkët në nivel kerneli. Kjo nuk përshtatej me traditën e Xen ndaj pati shumë mosmarrëveshje. Më në fund, në takimin *USENIX* të vitit 2006, VMware dhe Xen ranë dakord të punojnë së bashku (edhe me të tjerë) për të zhvilluar një ndërfaqe më të përgjithshme, të njohur si *paravirt_ops.*

Përfundimi i të gjithë kësaj është që përfshirja e mundshme e pjesëve paravirt_ops në kernelin kryesor do t'i japin mundësinë çdo teknologjie virtualizimi të bazuar në hypervisor që të punojë me kernel Linux, ndërsa projektet e kernel si DVM do t'u mundësojnë përdoruesve të ekzekutojnë makina virtuale, duke përdorur çdo sistem operativ në hardware që i mbështet, pa kërkuar një hypervisor. UML do të vazhdojë t'u japi mundësinë përdoruesve të ekzekutojnë makina virtuale Linux në një sistem të vetëm Linux-i.

2.7 Virtualizimi i kapaciteteve të ruajtjes

Virtualizimi i kapaciteteve të ruajtjes [13] është një abstraksion logjik për ruajtjen fizike. Ai është çelësi i krijimit të sasive fleksibël dhe të zgjerueshme të kapaciteteve ruajtëse për sistemet e sotme kompjuterike.

Virtualizimi i kapaciteteve ruajtëse ka qenë i përhapur prej shumë vitesh, dhe mund të jetë i njohur për çdo njeri që ka punuar me RAID, volume logjike në sisteme të tilla si: Linux apo AIX, ose me sisteme skedarësh të rrjetave si AFS dhe GFS. Te gjitha këto teknologji kombinojnë pajisjet e disqeve fizike të disponueshme, në grupe prej kapacitetesh të ruajtjes që mund të ndahen në seksione logjike të njohur si volume në te cilin mund të krijohet dhe të montohet çdo sistem skedarësh për përdorim në një sistem kompjuterik. Një volum është një ekuivalent logjik i një particioni disku.

Elementët bazë që e bëjnë virtualizimin e kapaciteteve të ruajtjes kaq të pëlqyer në mjediset e kompanive në ditët e sotme është se mundësojnë një kapacitet efektiv të pafundmë qe limitohet vetëm nga numri dhe përmasa e pajisjeve të suportuar fizikisht nga sistemi host ose sistemi host i ruajtjes. Virtualizimi i kapaciteteve ruajtëse mundëson një sasi më të madhe të kapacitetit fizik të disponueshëm për sistemet individuale dhe u jep mundësinë sistemeve ekzistuese të zmadhohen për të mbajtur atë informacion. Teknologji të tilla si *RAID* (Redundand Array of Inexpensive Disks) dhe volumet logjike të siguruara nga Linux LVM, LVM2, dhe paketat EVMS janë përgjithësisht të limituara për t'u përdorur në një sistem në të cilin pajisjet aktuale të ruajtjes janë të lidhura fizikisht. Disa kontrollues RAID janë me porta duale duke u lejuar shumë kompjuterëve akses te të njëjtët volume dhe sisteme skedarësh nëpërmjet kontrollerit RAID.

Për të përdorur menaxhuesit e volumit logjik, duhen përcaktuar particionet e diskut që do përdoren për vëllimet logjike, krijimi i vëllimeve logjike në atë pajisje fizike, dhe më pas krijimi i një sistemi skedarësh në vëllimet logjike. Më pas mund të montohen dhe përdoren këto sisteme skedarësh njësoj sikurse të montoheshin dhe përdoreshin sisteme skedarësh që ishin krijuar në particione të disqeve fizike.

Ashtu si kontrolluesit standardë të diskut, kontrolluesit RAID sigurojnë akses në blloqe të pajisjeve të ruajtjes që i janë bashkangjitur, edhe pse kapaciteti i disponueshëm për çdo bashkësi disqesh varet nga niveli RAID që do të përdoret. Pajisjet e lidhura tek kontrolluesi RAID disponohen për sistemin si të ishin një disk i vetëm i cili mund të ndahet sipas dëshirës, krijohen file në këto pjesë, dhe montimi e përdorimi bëhet njësoj si përdorimi i pjesëzave fizike.

Sistemet operative si Linux mbështesin software-in RAID, ku nuk nevojitet që asnjë kontrollues fizik RAID të jetë prezent. Sistemi software RAID funksionon ekzaktësisht si një kontrollues hardware RAID duke siguruar një akses me blloqe të kapacitetit të disponueshëm por duke përforcuar karakteristikat e niveleve RAID të ndryshme në software se sa në hardware. Software-i RAID është shume efiçent dhe ka një performancë pak më te vogël sesa shume kontrollues hardware. Shumë administrator sistemi aktualisht preferojnë software RAID më shumë se sa hardware RAID sepse kontrolluesit RAID hardware janë shumë të ndryshëm nga një prodhues në një tjetër dhe për më tepër nga një kontrollues në një tjetër. Dështimi i kontrolluesit RAID kërkon një zëvendësim të kontrolluesit të të njëjtit tip nga i njëjti prodhues në mënyrë që të aksesojë të dhëna në pajisjen e kapaciteteve të ruajtjes të lidhur me kontrolluesin e dështuar. Nga ana tjetër, software RAID është plotësisht i lëvizshëm përmes të gjitha sistemeve Linux në të cilët software-i është i instaluar për aq kohë sa ata mbështesin të njëjtat ndërfaqe fizike të diskut.

Teknologjitë e *filesystem-eve* të shpërndara, si AFS dhe GFS kanë mekanizmat e vet të krijimit dhe menaxhimit të brendshme të volumeve logjike, gjithashtu bëjnë të mundur bashkëpërdorimin e fileysteme-ve në këto volume logjike midis shumë sistemeve kompjuterike.

NFS (Network Filesystem) në shumicën e sistemeve operative të ngjashëm me UNIX, bën të mundur bashkëpërdorimin e kapaciteteve të ruajtjes logjike përmes sistemeve të shumta kompjuterike, këtë e bën nëpërmjet eksportimit të një direktorie nga një filesystem në kapacitetin lokal në vend që të montojë direkt një volum specifik ndaj

sistemit apo filesystem në rrjet. Sistemet e filesystem-eve të shpërndarë, si AFS dhe GFS sigurojnë një akses në filesystem me nivele në vëllimet logjike. Në këtë drejtim ata janë të ngjashëm konceptualisht me pajisjet e ruajtjes të lidhura në rrjet të cilat sigurojnë një akses në nivel filesystem-i mbi një rrjet për filesystem-et që ata përmbajnë.

Virtualizimi i kapaciteteve të ruajtjes është bërë shumë i aksesushëm përmes sistemeve kompjuterike të shumëfishta me përdorimin e SAN (Storage Area Networks) të I/O në nivel blloqesh dhe u mundëson sistemeve të shumëfishta për të ndarë aksesim të nivelit të ulët tek tipe të ndryshme të pajisjeve të kapaciteteve të ruajtjes mbi rrjetin. Shumica e SAN-ve përdorin teknologji rrjeti të shtrenjta dhe me fuqi të lartë e të shtrenjtë, teknologjitë e rrjetit të fuqisë së lartë si kanali i Fibrës dhe *Infiniband* për të siguruar nivelet e larta të *throughput-it* dhe performancë të përgjithshme që janë shumë të dëshirueshme kur shumë sisteme ndajnë akses te ruajtjet e të dhënave në nivel protokolli apo nivel blloku.

Teknologjitë e reja si iSCSI (Internet Small Computer Systems Interface), dhe AoE (ATA over Ethernet) sigurojnë mekanizma më pak të kushtueshëm për aksesimin në nivel blloku të pajisjet e ruajtjes të lidhura në rrjet. Siç sugjeron emri, ISCSI mbështet përdorimin e protokollit SCSI mbi rrjetat TCP/IP dhe kërkon një tip të veçantë të kontrolluesit të rrjetit. AoE siguron akses në nivel blloqesh të pajisjeve të përshtatshme ATA duke përdorur vetëm një lidhje standarte Etherneti. Të dyja këto performojnë më mirë në rrjeta me gjerësi brezi të mëdha si rrjetat Gigabit Ethernet, megjithatë ato janë të përdorshme në rrjeta 100-megabit. iSCSI dhe AoE po e kthejnë ruajtjen e të dhënave në rrjet në një mundësi shumë reale për shumicën e qendrave të sotme të të dhënave dhe infrastrukturave IT të çfarëdo përmase.

2.8 Virtualizim në nivel sistemi ose virtualizimi i sistemeve operative

Virtualizimi në nivel sistemi [14], shpesh i referuar si virtualizimi i sistemit operativ, përshkruan implementime të ndryshme të ekzekutimit të mjediseve të shumëfishta në një mjedis të vetëm të kernelit të sistemit operativ. Virtualizimi në nivel

sistemi bazohet te koncepti i ndryshimit të rrënjës (chroot) i cili është i vlefshëm në të gjithë sistemet moderne të ngjashme me UNIX.

Gjatë procesit të butimit të sistemit, kerneli mund të përdorë filesystem-e rrënjë për të ngarkuar driver-at dhe të performojë inicializime të sistemit në faza të hershme. Kerneli mund të udhëtoj në një filesystem tjetër rrënjë duke përdorur komandën chroot në mënyrë që të montojë një filesystem në disk si filesystemin rrënjë përfundimtar. Mekanizmi chroot i përdorur nga virtualizimi në nivel sistemi i mundëson sistemit të nisë serverët virtual me proceset e tyre që ekzekutohen lidhur me direktoritë rrënjë të filesystemit të tyre. Veprimi brenda kufijve të direktorive të tyre rrënjë dhe filesysteme-ve parandalon që serverët virtualë të aksesojnë file në filesystemet e njëri-tjetrit, dhe rrjedhimisht siguron mbrojtje bazë nga shfrytëzimi i proceseve të ndryshme të serverit ose vetë serverit virtual. Edhe nëse një server në chroot komprometohet, ai ka akses vetëm te filet që janë lokalizuar brenda filesystem-it rrënjë të tij.

Ndryshimi bazë midis virtualizimit në nivel sistemi dhe virtualizimit të serverit është se nëse mund të ekzekutohen sisteme operative të ndryshme në sisteme virtuale të ndryshme. Nëse të gjitha serverët virtualë mund të ndajnë një kopje të vetme të kernelit të sistemit operativ siç tregohet ne figurën 2.3, ky është një virtualizim në nivel sistemi. Nëse serverë virtualë të ndryshëm mund të ekzekutojnë sisteme të ndryshme operative, përfshirë versione të ndryshme të një sistemi operativ të vetëm, ky është virtualizim serveri, ndonjëherë i njohur si virtualizim makine. Zgjidhjet e virtualizimit si FreeBSD, FreeVPS, Linux Vserver, OpenVZ, Solaris Zones dhe Containers, dhe Virtuozzo janë të gjithë shembujt e një virtualizimi në nivel sistemi. FreeBSD jails mund të ekzekutojë versione logjikisht të dallueshme të hapësirës përdorues të FreeBSD në krye të një kerneli të vetëm FreeBSD, dhe kështu mund të përdorin instanca të ndryshme të versioneve të librarive, proceseve server dhe aplikimeve. Kontenieret dhe zonat e Solaris ndajnë gjithashtu të njëjtin version të poshtëm te Solaris, dhe mund te përdorin ose sisteme rrënjë plotësisht të ndryshëm të filesysteme-ve, ose të ndajnë pjesëzat e një filesistemi.

Linux-Vserver, FreeVPS dhe OpenVZ mund të ekzekutojë grupe të ndryshme të Linux në serverët e tyre virtualë, por të gjitha ndajnë te njëjtin kernel.

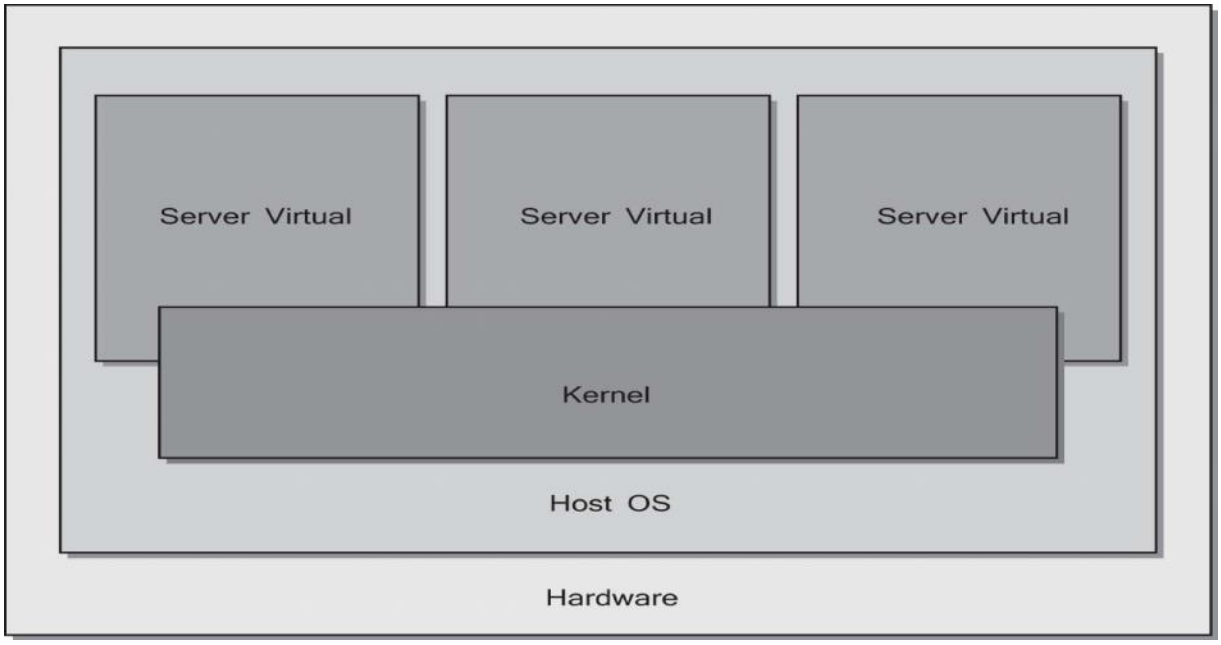


Figura 2.3- Diagrami logjik i virtualizimit në nivel sistemi

Virtualizimi në nivel sistemi siguron disa avantazhe të rëndësishme mbi virtualizimin e serverëve ose makinave. Sekreti është se meqë përdorin së bashku një instancë të vetme të kernelit, zgjidhjet e virtualizimit në nivel sistemi janë më të lehta në dukje se sa makinat e plota (që përfshijnë një kernel) që kërkohen nga teknologjitë e virtualizimit të serverëve.

Kjo aftëson që një host i vetëm fizik të mbështesë më shume "serverë virtual" sesa numri i makinave virtuale të plota që mund të përballojë. Virtualizimet e nivelit te sistemit si FreeBSD chroot , Linux-Vserver dhe FreeVPS janë përdorur për vite me radhë nga biznese si ISP për t'i mundësuar çdo përdoruesi serverët virtual të tyre në të cilët ata kanë kontroll relativisht të plotë pa pasur shanse për të kompromentuar konfigurimin e sigurisë primare, sistemit të konfigurimit të file dhe filesystem-in. Kështu virtualizimi në

nivel sistemi është me i përhapur për konsolidimin e serverëve. Disavantazhi kryesor i virtualizimit në nivel sistemi është që një problem kerneli apo driver-i mund të “rrëzojë” të gjitha serverët virtualë në nivel sistemi.

2.9 Pse është i përhapur virtualizimi në ditët e sotme?

Virtualizimi nuk është një koncept i ri, ai ka qenë në përdorim për dekada në mënyra të ndryshme. Megjithatë, virtualizimi është më popullor tani se kurrë më parë sepse tani është një opsion i përdorshëm nga një grup më i madh përdoruesish dhe administratoresh të sistemeve. Ka disa arsye për rritjen e popullaritetit të virtualizimit:

- Fuqia dhe performanca e pajisjeve hardware x86 vazhdon të rritet. Procesorët janë më të shpejtë se kurrë, përballojnë më shumë memorie se kurrë dhe procesorët multi-core mundësojnë që sisteme të vetme të performojnë njëkohësisht punë të shumëfishta. Këta faktorë kombinohen duke rritur shansin që hardware-i të nënvlerësohet.
- Integrimi në mbështetjen për virtualizimin në nivel hardware në gjeneratat e fundit i procesorëve Intel dhe AMD, në njësitë qendrore, dhe *firmware* e ka bërë virtualizim në hardware-in e produkteve të tregut më të fuqishëm se kurrë më parë.
- Një gamë e gjerë e produkteve si për desktop-et dhe për sistemet e serverëve që ekzekutohen në produkte hardware x86 janë përhapur, po vazhdojnë të përhapen dhe janë bërë shumë popullore.
- Virtualizimi është më i aksesueshëm, më i fuqishëm dhe më fleksibël se kurrë, ai vazhdon të provojë vlerat e tij në mjedise biznesi dhe akademike në të gjithë botën.

2.10 Përafrime bazë të sistemeve virtuale

Përafrimet më të zakonshme për sistemet kompjuterike virtuale që përdoren sot janë si më poshtë:

- **Kernel i sheruar:** Një kernel i vetëm i një sistemi operativ përballon shumë sisteme virtuale. Çdo sistem virtual ka filesystem-in e tij rrënjë. Meqë të gjitha makinat virtuale ndajnë të njëjtin kernel, libraritë që ekzekutohen nga këto makina virtuale duhet të kenë qenë të kompiluara për të njëjtin kernel dhe bashkësi instruksionesh si dhe makina fizike në të cilën punojnë sistemet virtuale. Më shumë detaje rreth kësaj përqasje të virtualizimit dhe disa shembuj te virtualizimit që përdorin këtë përqasje jepen në figurën 2.3.

- **Guest OS:** Çdo server virtual ekzekutohet si një instancë e veçuar e sistemit operativ që ka një aplikacion virtual që ekzekutohet në një instancë të një sistemi operativ të veçantë. Shembujt më të përhapur të kësaj zgjidhjeje virtualizimi janë Parallels Workstation, Vmware Workstation dhe VMWare BSX Server.

Në këtë lloj virtualizimi, hosti copëton burimet e veta në shumë makina të pavarura që mund të suportojnë sisteme operative të ndryshme dhe aplikacione konkurrente. Makina virtuale është mjedis operues i vetë-mbajtur që ekzekutohet në krye të shtresës së virtualizimit dhe sillet si të jetë kompjuter i ndarë.

Makinat virtuale veprojnë brenda një aplikimi që ekzekutohet si një aplikim standard nën sistemin operativ që ekzekutohet në një sistem host fizik. Ky aplikacion menaxhon makinat virtuale, ndërmjetëson aksesin në burimet hardware në sistemin e host-it fizik dhe ndërpret e menaxhon çdo instruksion të privilegjuar të nxjerrë nga makinat virtuale.

Figura 2.4 ilustron këtë përqasje të virtualizimit. Ky tip i virtualizimit në mënyrë tipike ekzekuton makinat virtuale sistemin operativ, utilitetet dhe libraritë e të cilëve janë kompiluar për të njëjtin tip të procesorit dhe bashkësisë së instruksioneve si dhe makina fizike në të cilën sistemet virtuale janë duke u ekzekutuar. Megjithatë mundet që të ekzekutojë makina virtuale, librari dhe utilitete që janë kompiluar për proçesorë të tjerë.

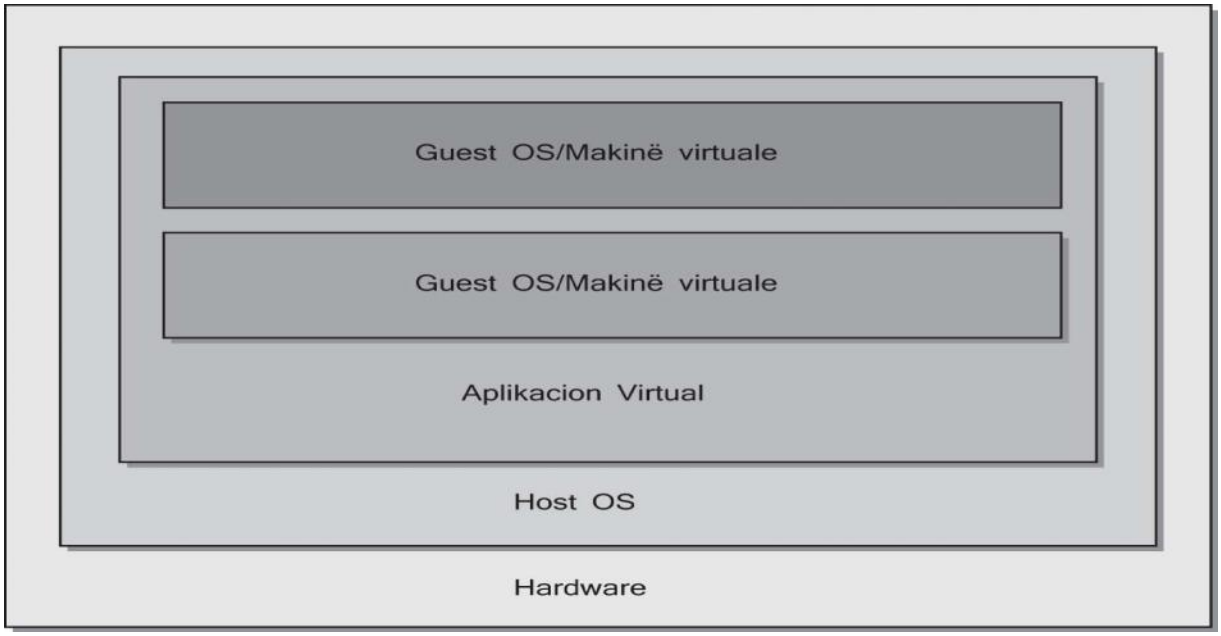


Figura 2.4 -Diagrami logjik për Guest OS

- **Hypervisor-i:** Një hypervisor është një monitor i makinës virtuale që ngarkohet gjatë procesit të butimit, përpara makinave virtuale, dhe ekzekutohet drejtpërdrejtë në hardware-in fizik siç tregohet në figurën 2.5. Një monitor i vogël virtual i makinës (i njohur si hypervisor) ekzekutohet në krye të hardware-it të makinës dhe siguron dy funksione bazë. Së pari, identifikon, kufizon, dhe u përgjigjet veprimeve të privilegjuara apo të mbrojtura të CPU-së të bëra nga çdo makinë virtuale. Së dyti, menaxhon rradhën, shpërndarjen dhe kthimin e rezultateve te kërkesave ndaj

hardware-it nga makina virtuale. Një sistem operativ administrativ vepron në krye të hypervisor-it, siç bëjnë vetë makinat virtuale.

Hypervisor-ët përdoren për te përballuar makinat virtuale në mjediset "paravirtualizim" "virtualizim të plotë" dhe në mjediset e "virtualizimit hardware". Në varësi të tipit të hypervisor-it të përdorur dhe në përqasjen specifike të virtualizimit që ndiqet, kodi i burimit të sistemit operativ që vepron në një makinë virtuale mund te këtë nevojë për t'u modifikuar për të komunikuar me hypervisor-in. Figura 2.6 tregon makina virtuale të bazuara te hypervisor-i që zhvillojnë suportin hardware për virtualizimin por gjithashtu kërkojnë një hypervisor për disa tipa të ndërveprimeve administrativ me makinat virtuale.

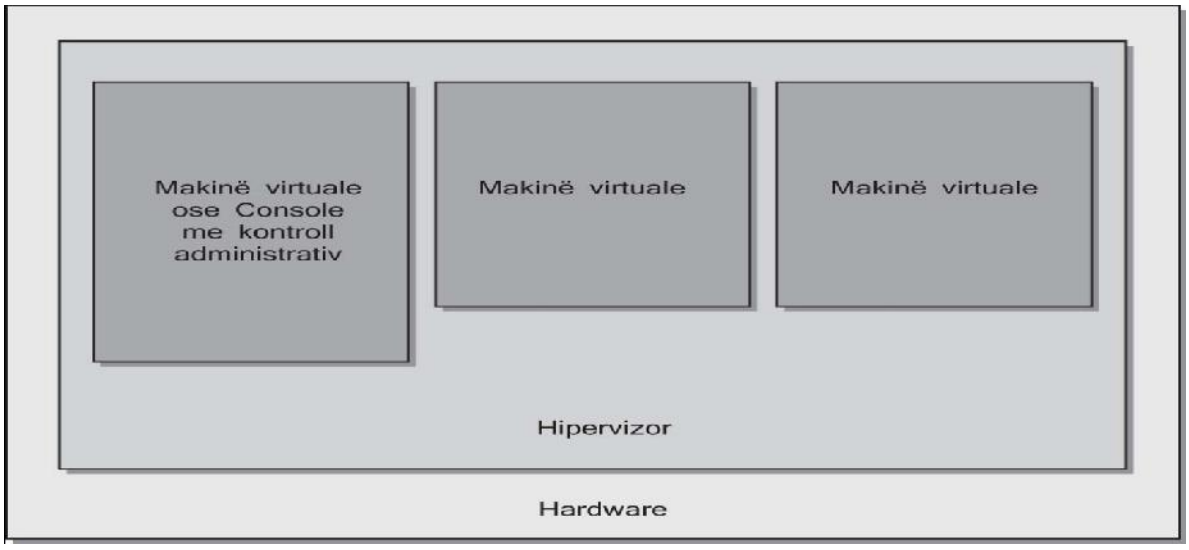
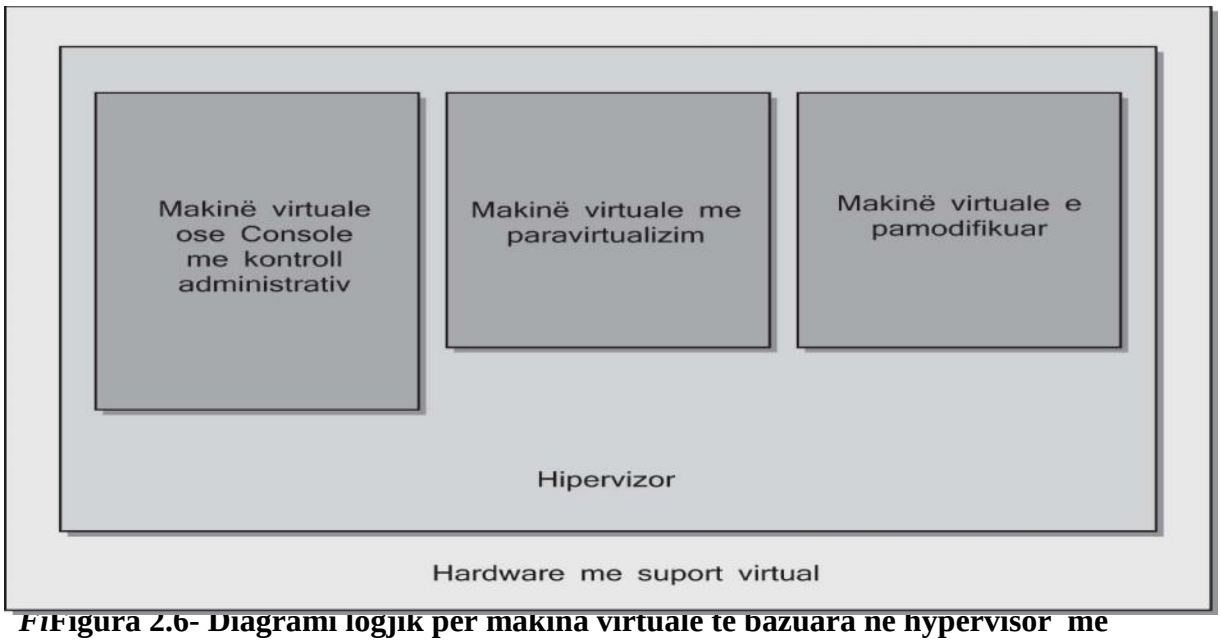


Figura2.5-Diagrami logjik për makina virtuale të bazuara në hypervisor



***FI*Figura 2.6- Diagrami logjik për makina virtuale të bazuara në hypervisor me suport hardware për virtualizim**

- **Niveli Kernel:** Kerneli i Linux i ekzekuton makinat virtuale si çdo proces në hapësirën e përdoruesit siç tregohet në figurën 2.7. Ky tip i virtualizimit ekzekuton makina virtuale, sistemin operativ, libraritë dhe utilitetet e të cilëve janë kompiluar për të njëjtin hardware dhe bashkësi instruksionesh si dhe ato të kernelit Linux në të cilin po ekzekutohen, i cili është kompiluar për makinën fizike në të cilën ekzekutohen sistemet virtuale. Kjo siguron një mënyrë të lehtë për të ekzekutuar makina virtuale të shumëfishta në një host fizik të vetëm. Shembuj të tilla janë User-Mode Linux (UML), dhe makinat virtuale kernel (KVM), të cilat u prezantuan në versionin e kernel-it 2.6.20. UML nuk kërkon ndonjë software administrativ të ndarë në mënyrë që të ekzekutohen ose te menaxhohen makinat e tij virtuale, të cilat mund të ekzekutohen nga rreshti i komandës së Linux-it. KVM përdor një pajisje driver-i në kernelin e host-it për komunikimin midis kernelit të Linux-it dhe makinave virtuale. Kërkon mbështetje procesori për virtualizim. Në shumë mënyra KVM është një

version i specializuar i virtualizimit të plotë, kur kerneli Linux shërben si hypervisor.

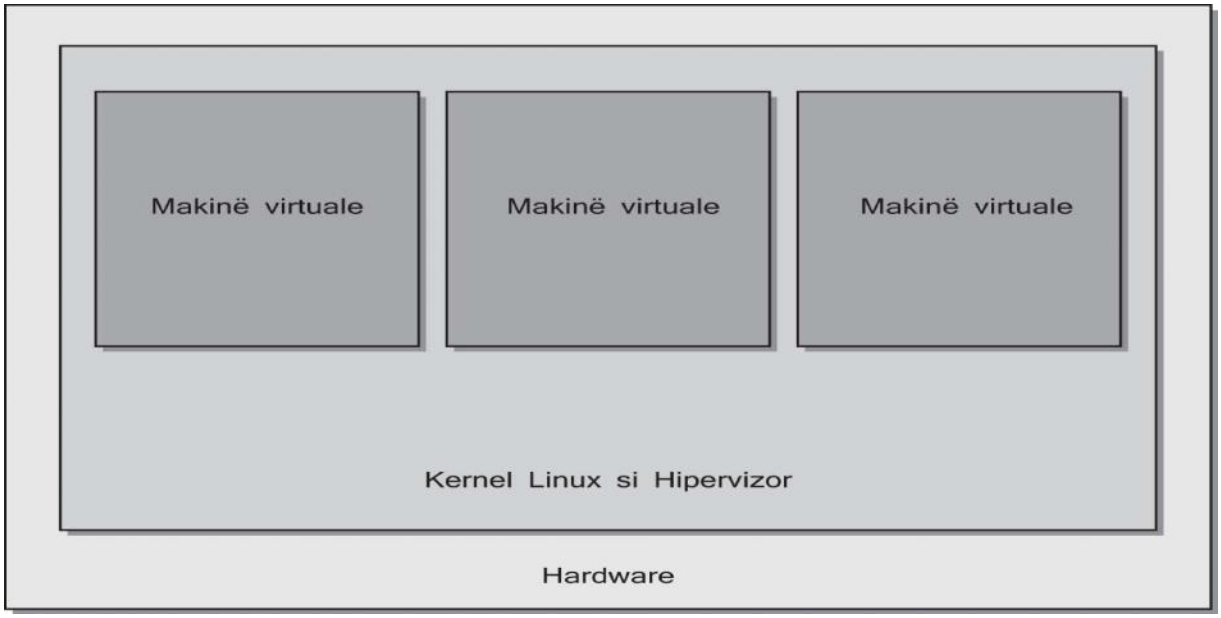


Figura 2.7-Diagrami logjike për virtualizimin në nivel kerneli

- **Emulimi:** Një emulator [15] i ekzekuton makinat virtuale duke simuluar një tip specifik procesori, bashkësinë e instruksioneve shoqëruese të tij si dhe hardware periferikë të detyrueshëm, e për pasojë mund të ekzekutojë sisteme operative dhe software shoqëruese që janë kompiluar për procesorë dhe bashkësi instruksionesh të ndryshme nga ai i përdorur nga hardware-i fizik në të cilin punon. Termat "emulim" dhe "virtualizim server/makine" ngatërrohen lehtë sepse të dyja mundësojnë shumë instanca të sistemeve të ndryshme operative në një sistem të vetëm host. Diferenca kryesore midis të dyve qëndron në faktin se nëse ekzekutojnë makina virtuale që janë kompiluar për bashkësinë lokale të instruksioneve të hardware-it fizik mbi të cilin punojnë makinat virtuale, ose ato që janë

kompiluar për ndonjë procesor e bashkësi tjetër instruksionesh. Teknologjia më e mirë e emulimeve e njohur sot është QEMU, e cila mund të emuloj procesorë x86 32 dhe 64 bit, PowerPC, Motorola 68000, SPARC 32 dhe 64 bit, SH, MIPS dhe ARM dhe ti ekzekutojë sistemet operative shoqëruese në ato mjedise të emuluara. Virtual PC e Microsoft është një mjedis emulimi sepse emulon hardware-in dhe bashkësinë e instruksioneve të PC-së duke i dhënë mundësi të ngarkojë dhe ekzekutojë sisteme operative x86 si Linux dhe Microsoft Windows si në platformë x86 edhe në platformë Macintosh PPC.

2.11 Përmbledhje

Në këtë kapitull u trajtuan teorikisht lloje të ndryshme Virtualizimesh, si edhe u dha një koncept i qartë mbi Virtualizimin si një nga teknikat më të avancuara të kohës. Nëse i referohemi lidhjes nga lart poshtë, do të shohim fillimisht virtualizimin e aplikacioneve. Kjo do të thotë se aplikacione të ndryshme mund të ndajnë instanca mes tyre.

Më pas kalohet në nivel Virtualizimi për Sistemet Operative. Kjo do të thotë që Sisteme të ndryshme Operative mund të ndajnë të njejtën kernel dhe të duken si aplikacione. Këto lloj sistemesh operative gjithsesi përveç disa instancave të përbashkëta janë plotësisht të izoluar nga njëri tjetri, p.sh. në raste virusimi të një Sistemi kjo nuk do të ndikojë te Sistemet operative të tjera.

Virtualizimi në nivel Hardware ose siç quhet ndryshe Virtualizimi i plotë bën që Sisteme Operative me ndërfaqe të ndryshme mund të ndajnë të njejtin hardware. Pjesë e kësaj teknike është Para-Virtualizimi, ku kërkohet të bëhet modifikimi i Kernelit për çdo Domain mbi host. Një rol në këtë lloj virtualizimi luan emulimi i burimeve.

Virtualizimi mund të shtrihet dhe në nivel rrjeti, ku rrjeta me topologji të ndryshme, me burime të ndryshëm mund të abstragohen shumë mirë.

KAPITULLI I TRETË

AVANTAZHET DHE DISAVANTAZHET E VIRTUALIZIMIT

3.1 Avantazhet e Virtualizimit

Virtualizimi mund të sjellë shumë avantazhe operacionale dhe financiare si një teknologji kyçe si për mjediset përpunuese të ndërmarrjeve edhe për mjediset e zhvillimit të software-ve. Më poshtë theksohen këto avantazhe thelbësore dhe shtjellohen se si ato mund të na kursejnë kohën dhe paratë si dhe të ndihmojnë shmangien apo minimizimin e shumë problemeve të infrastrukturës, përdorimit apo disponueshmërisë.

3.1.1 Përdorimi më i mirë i hardware-it ekzistues

Në dekadat e fundit procesorët kanë kaluar nga 8 bit në 16 bit dhe tani në 64 bit. Secili prej këtyre ndryshimeve në madhësinë e procesorit është shoqëruar me një rritje përkatëse në kapacitetin e memories dhe të kapacitetit të ruajtjes që këta procesorë mund të adresojnë dhe aksesojnë. Në mënyrë të ngjashme vazhdojnë të rriten shpejtësitë dhe densitetet e procesorëve, ku procesorët e sotëm e kalojnë lehtësisht frekuencën 2 GHz dhe përmbajnë disa bërthama procesuese në një çip.

Pavarësisht kësaj, shumica e shpejtësisë dhe fuqisë procesuese në shumicën e sistemeve kompjuterike shpërdorohen. Server-at Web me ngarkesë të lartë, sistemet e krijimit, makinat e lojërave dhe mainframe-at të cilat kërkojnë inteligjencë të lartë mund të përdorin të gjithë fuqinë e tyre procesuese.

Ekzekutimi i shumë makinave virtuale në një server mundëson kryerjen dhe përdorimin më të mirë të fuqisë procesuese rezervë. Sistemet multiprocesorë ose multi-core mundet të ekzekutojnë makina të ndryshme virtuale në procesorë apo core CPU të ndryshme, duke marrë më të mirën nga secila pjesë e secilit procesor të vlefshëm në sistem. Gjithashtu merret një përdorim më i mirë i pajisjeve të tilla si ndërfaqet e rrjetit të cilat janë të pranishme në serverët ekzistues duke i ndarë midis makinave virtuale.

Ekzekutimi i disa serverëve virtualë në një hardware të vetëm fizik njihet përgjithësisht si “konsolidim i serverit”. Kjo ka të bëjë me vendosjen e shumë proceseve server dhe proceseve të tyre shoqërues në një sistem fizik të vetëm, duke e rritur rëndësinë e atij sistemi por duke rritur dhe mundësinë që të kthehet në një pikë të vetme dështimi për shumë shërbime. Sot konsolidimi i serverëve do të thotë ekzekutimi i shumë makinave virtuale në një sistem të vetëm fizik. Xen mund të ndihmojnë në eliminimin e pikave të vetme të dështimit *(single points of failure)* në infrastrukturën IT duke mundësuar serverë virtualë të lëvizshëm që mund të lëvizin lehtësisht nga një makinë fizike në një tjetër në raste të problemeve, ose siguron serverë virtualë që mund të rindizen në sisteme të tjera fizike në rastin e ngjarjeve të papritura apo dështimeve katastrofike.

3.1.2 Reduktime në kostot e hardware-it të ri

Ana tjetër e marrjes së më shumë performancë nga server-at ekzistues është se në shumë raste nuk është e nevojshme blerja e një hardware fizik të ri për të vendosur serverë apo shërbime të rinj. Web Serverë të rinj, serverë të rinj skedarësh për grupe të ndryshme ose për të përballuar rritjen e ngarkesës, menaxhim i përmbajtjes së re apo sisteme intraneti dhe sisteme të ngjashme u shtohen shpesh mjediseve të ndërmarrjeve meqë si ngarkesat e sistemeve ekzistuese ashtu dhe numri i përdoruesve përgjithësisht rriten.

Kombinimi i konsolidimit të serverëve me planifikimin e kapaciteteve mund të reduktojë numrin e makinave të reja që duhen blerë për të mbështetur shërbime të reja dhe ekzistuese duke kryer një përdorim më të mirë të sistemeve ekzistuese. Në disa raste, konsolidimi i serverëve mund të mos e eliminojë koston e hardware-it të ri, por mund ta reduktojë këtë kosto. Për shembull, blerja e memories shtesë ose kartave të rrjetit shtesë për sistemet ekzistuese mundëson zgjerimin e kapacitetit të tyre në mënyrë që të mbështesin makina virtuale të reja pa patur nevojë për të blerë gjithë sistemin e ri.

3.1.3 Reduktime në koston e infrastrukturës IT

Kursimi i kostos së blerjes dhe vendosjes së serverëve të rinj nuk është reduktimi i vetëm i kostos IT që shoqëron virtualizimin.

Sallat e makinave kanë një sërë kostosh infrastrukturore për makina të cilat mund të reduktohen duke marrë më shumë performancë nga hardware-i ekzistues në vend të shtimit të sistemeve të reja. Çdo server i ri fizik përdor një sasi të caktuar fuqie dhe vendos një ngarkesë të caktuar në sistemin e ftohjes. Makinat virtuale të shtuara te sistemet ekzistuese kompjuterike nuk i shtojnë këto ngarkesa duke mundësuar shtimin e më shumë serverëve pa rritje në kërkesat për fuqi apo ftohje. Në mënyrë të ngjashme, nëse mund të konsolidohen disa serverë ekzistues në një numër më të vogël sistemesh server, rrjedhimisht çon në reduktimin e kostos së fuqisë dhe ftohjes.

Gjatë konsolidimit të serverëve shpesh mund të kombinohen hardware nga serverët fizikë për të rritur kapacitetin e makinave që ngelen. Për shembull, mund të shtohet memorie nga një sistem i nxjerrë jashtë përdorimit në një server tjetër që mban shumë makina virtuale. Në mënyrë të ngjashme disqet që më parë përbënin ruajtjen lokale të sistemeve të nxjerrë nga përdorimi mund të përdoren si pajisje rezervë, për backup, në sisteme RAID e kështu me radhë.

Në varësi të numrit të shërbimeve dhe të proceseve server që duhet të mbahen dhe të shkallës së suksesit që merret nga konsolidimi i serverëve, virtualizimi mund të reduktojë kërkesat për hapësirë në cilado sisteme që mban një kasë në ISP ose që stivohen pranë njëri-tjetrit. Mund të sjellë edhe kursime të hapësirës.

Veç fuqisë, ftohjes dhe kursimeve në hapësirë, reduktimi i numrit të makinave fizike që menaxhohen mund të reduktojë akseset e largëta dhe kostot e besueshmërisë duke kërkuar më pak sisteme Tastierë-Video-Mouse (KVM), më pak lidhje me burime të pandërprera fuqie e kështu me radhë. Në varësi të mënyrës se si konfigurohet lidhja me rrjetin në hardware-in fizik që mban makinat virtuale dhe numrit të kartave të ndërfaqes

me rrjetin të instaluara në secilin sistem, mund të thjeshtohet kabllimi i rrjetit dhe të reduktohet numri i *hub-eve* dhe switch-eve që kërkohen në sallën e makinave.

3.1.4 Administrimi i thjeshtuar i sistemeve

Përdorimi i virtualizimit për të reduktuar numrin e sistemeve fizike që duhet të menaxhohen dhe mirëmbajtja nuk e redukton numrin e sistemeve për të cilët jemi përgjegjës. Gjithsesi sistemet segmentohen në dy grupe: ato që janë të lidhura me burime specifike fizike dhe ato që janë plotësisht virtuale. Sistemet fizike që mbajnë makinat virtuale janë shembulli parësor i grupit të parë, por ky grup përfshin dhe ato makina virtuale që kryejnë përdorim specifik dhe unik të burimeve fizike si karta rrjeti shtesë, pajisje lokale të ruajtjes e kështu me radhë. Ekzekutimi i shumë makinave virtuale në sisteme të vetme fizike e bën qëndrueshmërinë e atyre sistemeve më kritike për funksionet e biznesit dhe prezanton njëfarë infrastrukture të re software për migrimin apo klonimin e makinave virtuale në rastet e problemeve hardware.

Shumica e grupeve IT të ndërmarrjeve ekzekutojnë një lloj sistemi me gjendje të centralizuar ose *software-i heartbeat* duke mundësuar monitorimin për së largu te statusit të gjithë hardware-it pa kontrolluar secilën konsole. Në varësi të kapaciteteve të paketës monitoruese që ekzekutohet, krijohen seksione të ndara ose nivele vigjilente për sisteme me një varësi fizike në hardware-in lokal, sistemet me një varësi në sistemet e centralizuara të ruajtjes dhe sistemet që janë plotësisht virtuale. Meqë makinat virtuale "mendojnë" se ekzekutohen në hardware-in fizik duhen grupuar mesazhet e lidhura me hardware-in nga makinat virtuale në mënyrë që të identifikohet çdo problem në komunikimin e makinave virtuale/fizike. Në mënyrë të ngjashme duhen grupuar *log-et* software nga makinat virtuale, në mënyrë që të identifikohen problemet në zhvillim apo ato të menjëhershme në shërbimet software si web server-at të cilët mund të mbahen në makina të shumëfishta për balancim ngarkese apo arsye teprice.

Së fundmi, virtualizimi ndihmon në ristrukturimin dhe thjeshtimin e detyrave standarde administruese të sistemeve të cilat harxhojnë shumë kohë si p.sh. *backup-et*.

Shumë makina virtuale përdorin pajisje të ruajtjes së të dhënave të lidhura në rrjet që të jenë sa më të pavarura nga sistemet fizike në të cilat po ekzekutohen, si dhe që të përmirësojnë centralizimin në përgjithësi. Përdorimi në rrjet i ruajtjes së qendërzuar si SAN, iSCSI, ATA-mbi-Ethernet ose filesystem mund ta reduktojë numrin e makinave dhe sistemeve të ruajtjes që kërkojnë backup-e fizike. Në mënyrë të ngjashme nëse do përdoren klientë thin apo "virtualizime desktop-i" në mënyrë që të gjithë përdoruesit të logohen dhe të punojnë në server-a të centralizuar, nuk do të nevojitet backup-i i sistemeve desktop që ekzekutojnë vetëm software të kompjuterëve të largët dhe te të cilët nuk përdoret ruajtja lokale.

3.1.5 Kohë më e lartë pune dhe rikuperim më i shpejtë i dështimeve

Rritja e izolimit të makinave virtuale nga hardware-i fizik e rrit disponueshmërinë e sistemit duke rritur lëvizshmërinë e këtyre makinave virtuale. Lëvizshmëria e makinave virtuale i mundëson atyre të migrohen nga një server fizik te një tjetër nëse në sistemin e parë ndodhin probleme hardware. Makinat virtuale Xen mund të migrohen nga një host fizik në një tjetër pa ndonjë ndërprerje në disponueshmëri. Procesi i migrimit është transparent si për përdoruesit ashtu dhe për çdo proces që ekzekutohet në këto makina virtuale.

Përshtatja e virtualizimit dhe një strategjie për detektim të automatizuar të problemeve si dhe migrimin e makinave virtuale mund të ulë kostot që lidhen tradicionalisht me tepricën dhe dështimet sepse pjesa më e madhe e hardware-it që më parë duhej për të siguruar disponueshmërinë duke pasur sisteme fizike të tepërt tani mund të sigurohen nga mundësia e migrimit të shumë makinave virtuale te platforma hardware të tjerë të përshtatshëm në raste të shfaqjes së problemeve. Mund të migrohen sistemet virtuale pa ndërprerë shërbimin dhe mund të rritet disponueshmëria fizike gjatë ndërprerjeve të energjisë duke e reduktuar ngarkesën në burimet e pandërprera të fuqisë meqë mbahen më pak makina fizike duke patur të njëjtin nivel të disponueshmërisë së sistemit për një kohë më të gjatë.

Kur ndahen dhe caktohen shërbimet dhe programet për disponueshmëri të lartë, një çelës për disponueshmëri të lartë është ndarja efektive e makinave fizike dhe virtuale në varësi të shërbimeve që mundësojnë. Për shembull, në një mjedis plotësisht të virtualizuar, qëllimi parësor i makinave fizike do të ishte mbështetja e makinave virtuale, ato vetë nuk duhet të ofrojnë shërbime shtesë software. Kjo bën të mundur përgjigjen ndaj problemeve të reja hardware në sistemet fizike duke i migruar makinat virtuale në hardware të tjerë fizikë pa u shqetësuar për ndonjë shërbim software që mundësohet nga vetë makinat fizike. Në përgjithësi është mirë që infrastruktura IT të mbahet sa më e pavarur nga sistemet fizike në të cilat ekzekutohet çdo pjesë e saj.

3.1.6 Thjeshtim i zgjerimit të kapaciteteve

Zgjidhjet me virtualizim si makinat virtuale dhe ruajtja virtuale e të dhënave heqin kufizimet që imponohen shpesh herë nga makinat fizike ose ruajtjet lokale të të dhënave. Makinat virtuale mund të lëvizen nga një pjesë fizike e hardware-it në një tjetër që të kenë mundësi të përfitojnë nga zhvillimet hardware, si CPU më të fuqishme, më shumë core CPU, memorie shtesë, karta shtesë ose më të shpejta rrjeti e kështu me radhë. Në mënyrë të ngjashme virtualizimi i ruajtjes bën të mundur rritjen në mënyrë transparente të sasisë të vlefshme të kapaciteteve ruajtëse dhe madhësisë së particion-eve dhe filesystem-eve ekzistuese.

3.1.7 Mbështetje më e lehtë për sistemet dhe aplikacionet e vjetra

Virtualizimi është një zgjidhje shumë e mirë për nevojën për të ekzekutuar software të vjetër. Shumë biznese kanë aplikacione nga të cilat varen por të cilat nuk mundësohen më nga ndonjë shitës specifik ose që nuk janë përmirësuar për t'u ekzekutuar në sisteme operative apo hardware të rinj. Edhe pse varësia nga software të vjetër të cilët varen nga versione specifike të sistemeve operative apo të hardware-it është problem nga pikëpamja e biznesit, ky është ende një realitet për bizneset.

Mbështetja për software të vjetër dhe mjedise operative ishte një nga motivet kryesore të virtualizimit kur ky koncept u paraqit për herë të parë në një sistem operativ nga IBM në vitet 1960. Duke i ekzekutuar sistemet operative në particione logjike (të njohura në mainframe si LPAR) klientët mund të kalonin në sisteme operative të reja dhe hardware të rinj e më të fuqishëm pa humbur aftësinë për të ekzekutuar software-in ekzistues dhe sistemet operative shoqëruese nga të cilat varej biznesi i tyre.

Përdorimi i virtualizimit për të zgjidhur problemet me software-in e vjetër është një proces i thjeshtë. Konsiston në instalimin e sistemit të vjetër operativ në një makinë virtuale, instalimin e software-it të vjetër dhe sigurimin që ai funksionon siç duhet në mjedisin e ri. Instalimi dhe përdorimi i software-it të vjetër të varur nga identifikues platformash hardware unike dhe tradicionale si adresa MAC e një karte Etherneti thjeshtohet nga software virtualizues si Xen i cili bën të mundur vendosjen e adresës MAC që i shoqërohet çdo makine virtuale. Për shembull, nevoja për aksese rastësore në software që punojnë vetëm në sisteme operative të vjetra Microsoft Windows mund të arrihet lehtësisht duke krijuar një makinë virtuale në të cilën instalohet sistemi i vjetër Windows dhe software i kërkuar.

Zgjidhja e çështjeve të software-it të vjetër nëpërmjet virtualizimit është e mundur vetëm për software të vjetër që ekzekutohen në të njëjtën arkitekturë procesori si dhe software i virtualizimit. Për shembull, nuk mund të mbahet software të vjetër për platforma SPARC në software virtualizimi për platforma x86. Duhet pasur kujdes që zgjedhja e virtualizimit të mbështesë SO e vjetër, Xen është shumë fleksibël në këtë aspekt, por shumë zgjedhje virtualizimi të tjera nuk janë.

3.1.8 Zhvillimi i sistemit me nivele të thjeshtuar

Një zgjidhje tradicionale për zhvillimin dhe testimin e kernel-it dhe drivera-ve është që të kryhet zhvillimi në kontekstin e zgjidhjeve tradicionale të makinave virtuale Linux si User-Mode Linux (UML). Aftësia për të rinisur një makinë virtuale, për të testuar kernel dhe driver të rinj është shumë më e shpejtë dhe më pak ndërprerëse për

procesin e zhvillimit se sa rinisja e makinës fizike. Ky përafrim mund të japë dhe avantazhe të rëndësishme për korrigjimin e gabimeve në nivel të ulët nëse po punohet në një sistem desktop që mbështet makinat virtuale, sepse mjedisi i zhvillimit, sistemi i zhvillimit dhe makina virtuale mund të bashkekzistojnë në një platformë desktop. Zgjidhjet e virtualizimit si Xen mundësojnë një mjedis të thjeshtë zhvillimi për përdorim.

Zgjidhjet e virtualizimit të bazuara në Hypervisor janë mjedisi i duhur për testime finale të driver-ave hardware sepse paraqesin një mënyrë të tërthortë që ndikon në njëfarë mase te performanca, gjithashtu fsheh njëfarë aksesi të hardware-it që mund të kërkojnë driver-at. Gjithsesi virtualizimi është një mjedis shumë i mirë zhvillimi për drivera të nivelit të lartë dhe software të sistemit si filesystem-et të lidhur në rrjet. Në mënyrë të ngjashme driver-at hardware duhet të testohen kundrejt zgjidhjeve të virtualizimit të bazuara te hypervisor-i sa herë që është e mundur të verifikohet përputhshmëria.

Zhvillimi dhe testimi në makina virtuale është një përdorim i zakonshëm i LPAR në *mainframet* e sotme të IBM ku zhvilluesit mund të punojnë dhe të zhvillojnë distribucione Linux që ekzekutohen në particione Logjike që fizikisht qëndrojnë në mainframe.

3.1.9 Instalimi dhe shpërndarje e thjeshtuar e sistemit

Virtualizimi mundëson një zgjidhje të shpejtë, fleksibël dhe me kosto efektive për shpërndarjen e sistemeve të reja, kjo në varësi të shpejtësisë dhe memories së vlefshme në sistemin e server-it. Përdorimi i makinave virtuale mund të thjeshtojë shpërndarjen e sistemeve të reja duke përdorur një imazh të vetëm të filesystem-eve si bazë për të gjithë instalimet e reja. Për të instaluar një sistem të ri thjesht krijohet një makinë e re virtuale duke klonuar atë filesystem dhe duke nisur një instancë të re të makinës virtuale që përdor filesystem-in e ri.

Mundësia për të mbajtur përdoruesit dhe klientët në makina virtuale private mundet gjithashtu të përdoret për të thjeshtuar infrastrukturën për bizneset që kërkojnë numër të madh sistemesh që personalizohen shpesh nga përdoruesit. Ky tip virtualizimi i

mundëson secilit përdorues të ketë akses në makinën e tij dhe kontroll të plotë në ekzekutimin dhe mjedisin software. Përdorimi i makinave të plota virtuale mundëson gjithashtu ofrimin e çdo sistemi operativ të virtualizueshëm për klientët në vend që të kenë të gjithë një kernel çka do të kufizonte klientët me preferenca të ndryshme të Linux, BSD e kështu me radhë.

Kur përdoren makina virtuale të plota për shpërndarjen e sistemeve të rinj, aftësia për të migruar makinat virtuale nga një host në një tjetër mund të rezultojë si një aset kur makinat virtuale përdoren si një mekanizëm për shpërndarjen e sistemeve. Të pasurit e një sistemi zhvillimi të pavarur nga një platformë hardware fizike mund ta bëjë jetën e zhvilluesve më të lehtë dhe më produktive nga thjeshtimi i migrimit të atyre sistemeve që vendosen në makina më të shpejta e të fuqishme, sisteme me pajisje më të mira periferike e kështu me radhë. Kuptohet që mundësia apo jo e migrimit varet nga konfigurimi dhe kërkesat specifike hardware të secilës makinë virtuale, por mund të garantohen lehtësisht nëpërmjet planifikimit të zgjuar dhe projektimit të mirë të sistemit virtual.

Së fundmi, virtualizimi i desktop-it thjeshton shpërndarjen e sistemeve të reja duke reduktuar sasinë e software-ve që ka nevojë të instalohet lokalisht. Mundësia që përdoruesit të bashkëpërdorin një bashkësi software-sh që është e instaluar në një sistem serveri qendror kërkon kujdes të veçantë në lidhje me çështjet e licencimit për të siguruar që nuk shkelen termat e licencës të secilit software. Këto çështje shpesh mund të zgjidhen me përdorimin e software-it “open source” duke eliminuar çështjet e licencimit, ose me përdorimin e licencave “notuese” të cilat nuk kërkojnë licencë dhe që ruhen në një *pool* dhe i caktohen në mënyrë të përkohshme përdoruesve kur ata përdorin software-in.

Rritja e centralizimit të burimeve të përbashkëta dhe standardizimi i sistemeve të shpërndara mund të japë avantazhe të dukshme për administratorët e sistemit. Shpërndarja e sistemeve të lehtë desktop dhe përdorimi i desktop, Serveri Terminal i Microsoft, apo paketave të ngjashme për t’u lidhur në një server qendror thjeshton instalimin e software-it për secilin sistem, redukton kohën jo produktive sepse të gjithë sistemet desktop janë plotësisht të këmbyeshëm si dhe thjeshton detyrat e administrimit të

sistemeve si backup-et duke siguruar që asnjë skedar i rëndësishëm, i kërkuar apo personal të mos ruhet në disqet lokale.

3.1.10 Testim i sistemeve dhe aplikacioneve

Veç konsolidimit të serverëve dhe kursimeve në kosto të hardware-it e të infrastrukturës, testimi i sistemeve software dhe mjediset e sigurimit të cilësisë janë përfitimet më të mëdha të virtualizimit. Grupet e testimit të software-ve dhe të sigurimit të cilësisë në mënyrë tipike kanë nevojë të jenë në gjendje të testojnë një produkt specifik software në shumë sisteme operative të ndryshme apo versione të një sistemi operativ. Virtualizimi i serverëve në këto raste është shumë efektiv në kosto dhe kohë duke reduktuar sasinë e hardware-it të nevojshëm dhe duke reduktuar apo eliminuar shumë nga koha e kërkuar për instalimin e sistemit, riinstalimin dhe konfigurime të mëvonshme.

Virtualizimi i serverit e bën të lehtë instalimin e produkteve software dhe testimin e tyre në sisteme të ndryshme operative ose në versione të ndryshme të të njëjtit sistem pa kërkuar hardware të dedikuar për secilin. Përdorimi i një “fotografie” të ruajtur të një sistemi operativ është jo vetëm më e shpejtë se sa riinstalimi i të gjithë sistemit virtual apo fizik, por mund ta bëjë riinstalimin të panevojshëm nëse mund ta ktheni një makinë virtuale mbrapsht në gjendjen origjinale nëpërmjet “fotografive” apo përdorimit të disqeve jo të përhershëm që mbështeten nga disa zgjidhje virtualizimi.

Një përdorim interesant i virtualizimit në testimin e sistemeve jashtë sigurimit të cilësisë apo grupeve të testimit të sistemeve është përdorimi i virtualizimit për testimin e lëshimeve të reja të një sistemi operativ dhe software-it shoqërues të tij. Si shembull specifik, versionet më të reja të Microsoft Windows shpesh paraqesin mospërputhje me faqosjet ekzistuese të disqeve, ngarkuesit e kështu me radhë. Testimi i një sistemi operativ të ri brenda një makine virtuale, jep një model të mirë në të cilin mund të eksperimentohet me sistemin e ri operativ, të testohet e të verifikohet përputhshmëria e software-it etj, të gjitha këto pa “shqetësuar” në mënyrë të përhershme disqet, particionet dhe aplikacionet në sistemet ekzistuese.

3.2 Disavantazhet e virtualizimit

Siç u pa ka shumë avantazhe për integrimin e virtualizimit në një mjedis përpunues, në të njëjtën kohë virtualizimi nuk është zgjidhje për të gjitha problemet IT, ai nuk është i përshtatshëm për të gjithë skenarët dhe paraqet kosto e probleme.

3.2.1 Probleme të pikës së vetme të dështimit (single point of failure)

Konsolidimi i serverëve çon në një përdorim më të mirë të hardware-it ekzistues duke mundësuar përdorimin e fuqisë procesuese rezervë për të ekzekutuar disa makina virtuale në një host të vetëm. Në një organizatë IT tipike, secila prej makinave virtuale ekzekuton një server të vetëm ose një bashkësie shërbimesh të lidhura me njëra-tjetrën si *servera mail* dhe software-in përkatës anti-spam, server DNS, server print, server skedarësh etj. Mangësia e konsolidimit të serverëve është se rrit mundësinë për dështim të një makine të vetme fizike e cila mban shumë serverë virtualë çka ndikon shumë në punën e kompanisë. Nëse shumë serverë dhe shërbime shoqëruese ekzekutohen në makina individuale, dështimi i një makine të vetme ka një ndikim në vetëm një server. Kur shumë serverë ekzekutohen si makina virtuale në një hardware të vetëm, dështimi i atij hardware-i mund të nxjerrë jashtë shërbimi të gjithë server-at.

Zgjidhja për këtë lloj problemi është planifikimi i detajuar. Kur projektohet një infrastrukturë IT e bazuar te makinat virtuale, është shumë e rëndësishme të sigurojmë që planifikohet për disponueshmëri dhe dështime sa herë që është e mundur. Disa përafrime të zakonshme janë të mëposhtmet:

- Vendosja e një hardware-i të tepërt si karta të rrjetit në sistemet host dhe lidhja bashkë në mënyrë që dështimi i një karte të vetme të jetë transparent për makinat virtuale.

- Blerja dhe mbajtja e hardware-ve të dyfishtë për sistemet fizike që mbajnë makina virtuale të rëndësishme. Kostoja e të pasurit të një burimi ushqimi rezervë, bordeve, kartave të rrjetit etj.. është e parëndësishme krahasuar me koston e pritjes deri sa një shitës të dërgojë pjesët për zëvendësim.

- Replikimi i makinave virtuale që mbajnë shërbime kritike në disa sisteme fizike në mënyrë që ti mbijetohet dështimit të një makine të vetme fizike dhe kohës shoqëruese të humbur duke kaluar te serverët alternativë.

- Ekzekutimi i software-ve të centralizuara të monitorimit të sistemit që të lajmërojë për probleme hardware dhe software përpara se këto probleme të bëhen kritike.

Dështimet e centralizuara si ndërprerjet e ushqimit apo të rrjetit në sallën e makinave do të jenë gjithmonë probleme të mundshme dhe nuk ndryshojnë shumë në mjediset me virtualizim.

3.2.2 Bashkëpërdorimi i serverëve dhe çështje të performancës

Planifikimi për rritje të kërkesave është i lehtë kur krijohet çdo infrastrukturë IT. Është e rëndësishme të kryhet planifikimi i kapaciteteve kur specifikohen kërkesat hardware dhe projektohet infrastruktura IT. Planifikimi i kapaciteteve mund të jetë edhe më i rëndësishëm kur projektohet një infrastrukturë IT e bazuar te virtualizimi sepse masa në të cilën serverët virtual mund të mbajnë ngarkesën e lartë dhe të jenë të lëvizshme nëpër sisteme të shumëfishtë fizik varet shumë nga konfigurimi i tyre.

Ndërsa zgjidhje për virtualizimin e serverëve, si paravirtualizimi japin abstraksion domethënës të hardware-it të poshtëm, duhet patur kujdes në projektimin e serverëve virtualë që të jenë sa më të pavarur nga kufizimet fizike specifike. Për shembull, përdorimi i kapaciteteve lokale të ruajtjes do të jetë gjithmonë jofleksibël. Kërkesat për

kapacitete ruajtëse të server-ëve virtualë përbëjnë një problem të mundshëm për aq kohë sa edhe ato nuk abstragohen nga sistemet fizike, ose nëpërmjet përdorimit të sistemeve të filesystem-eve të lidhur në rrjet ose nëpërmjet ndonjë teknike të virtualizimit të kapaciteteve ruajtëse. Në mënyrë të ngjashme aplikacionet që varen nga avancimet më të fundit dhe më të mira të procesorëve mund të ekzekutohen ngadalë në mjedise të virtualizuara sepse nuk mund të marrin akses të drejtpërdrejtë në hardware-in që kërkojnë.

Shtimi i përdoruesve në makinat virtuale ekzistuese ose kryerja e më shumë detyrave me procesime intensive e rrit ndjeshëm sasinë e memories që kërkon një makinë virtuale si dhe sasinë e memories që mund t'i alokojë sistemi fizik. Gjithashtu kryerja e më shumë punëve intensive ndaj të dhënave në server-at virtualë mund të ndryshojë kërkesat për kapacitete ruajtëse të atyre serverëve si dhe mënyrën sipas të cilës ato e përdorin kapacitetin ruajtës, duke kërkuar hapësirë shtesë, ndryshime në mënyrën dhe vendin e alokimit të hapësirës *swap* dhe të faqosjes e kështu me radhë.

Nëse nuk mund të përballohen licencat për software-in nga i cili varet kompania, përdorimi i skemave fleksibël të licencimit si licencat "notuese" është kritike për server-at që mbajnë numër të madh përdoruesish. Një tjetër problem i mundshëm i lidhur me licencimin mund të shkaktohet nga shitësit software të cilët nuk e shërbejnë software-in e tyre në mjedise virtualizimi.

Edhe fusha të virtualizimit si virtualizimi i desktop-ve nuk janë të përshtatshme për të gjithë përdoruesit. Për shembull, përdoruesit e largët ose përdoruesit e laptop-ve do të kenë nevojë për më shumë software lokal të instaluar në sistemet e tyre për të garantuar mundësinë e tyre për të kryer punët pa varësi të vazhdueshme në akseset e largëta te sistemeve të centralizuara.

3.2.3 Dyndjet e rrjetit nga secili server

Shumica e makinave të plota virtuale përdorin ndërfaqe virtuale rrjeti, nënrrjete, dhe *paketa bridging* për ti lidhur ato ndërfaqe me hardware-in fizik. Nëse sistemi fizik

disponon vetëm një ndërfaqe rrjeti, ekzekutimi i shumë makinave virtuale që kryejnë detyra intensive rrjeti mund të shkaktojë shumë kërkesa në hardware-in fizik të rrjetit. Kjo mund të rezultojë në probleme të performancës së rrjetit për një host të vetëm ose për të gjithë hostet që bashkëpërdorin ndërfaqen e rrjetit. Një zgjidhje e qartë është të instalohen disa ndërfaqe rrjeti në sistemin fizik dhe tu caktohen makinave virtuale specifike. Fatkeqësisht ky tip konfigurimi mund të komplikojë migrimin e këtyre makinave virtuale nga një host në një tjetër për të zgjidhur problemet hardware që mund të ndodhin apo problemet me performancën.

3.2.4 Rritja e kompleksitetit të rrjetit dhe kohës së korrigjimit të gabimeve

Lidhja në rrjet e makinave virtuale të plota përdor ndërfaqe virtuale rrjeti, nënrrjete dhe *software bridging* për ta bërë secilin host të duket sikur ka një ndërfaqe rrjeti unike. Përdorimi i makinave virtuale të plota ku secila ka ndërfaqen e saj të rrjetit (virtuale apo fizike), adresën IP, e kështu me radhë mund të jetë më komplekse sesa menaxhimi i shumë hosteve fizik me të njëjtat karakteristika. Kjo jo vetëm për shkak të niveleve shtesë të software-it që kërkojnë zgjidhjet e ndryshme të lidhjes virtuale në rrjet por edhe sepse *firewall-et* dhe mekanizmat e tjerë të kontrollit kanë nevojë të konfigurohen që të lejojnë trafik specifik në drejtime specifike dhe midis hosteve virtual specifik.

Shumë site ngrenë makina virtuale në subnet-et e tyre çka kërkon serverë të ndarë DHCP që të menaxhohen brezat shoqërues të adresave IP. Këto subnete gjithashtu thjeshtojnë firewall-et dhe çështjet e routimit, sepse firewalli apo routeri mund të konfigurohet lehtësisht për të trajtuar bllokun e adresave IP që shoqërojnë atë subnet në mënyrë të ndryshme.

Në botën reale, secila pajisje fizike *Ethernet* ka një adresë unike MAC por shumica e makinave virtuale të plota mund të marrin adresa specifike MAC. Duhet pasur kujdes kur klonohen apo personalizohen të dhënat e makinave virtuale përndryshe do të jetë e lehtë të vendosen aksidentalisht disa hoste me ndërfaqe rrjeti që kanë adresa të

njëjta MAC. Kjo mund të shkaktojë probleme të routimit dhe tejçimit të paketave dhe është e vështirë të identifikohet.

Në përgjithësi, përdorimi i ndërfaqeve virtuale të rrjetit dhe makinave virtuale mund të komplikojë shumë detyra të menaxhimit të rrjetit nëse nuk shpenzohet një sasi e konsiderueshme kohe për zgjidhjen e problemeve të mundshme gjatë fazës së planifikimit të infrastrukturës së makinave virtuale.

3.2.5 Rritja e kompleksitetit aministrativ

Administrimi i thjeshtuar i sistemeve merret si një përfitim i mundshëm i virtualizimit, por kjo nuk është e thënë të jetë gjithmonë e vërtetë nëse përdoren utilitete të menaxhimit të sistemeve të shpërndara të cilat nuk i njohin makinat virtuale. Nëse është kështu ose nëse përdoren disa zgjidhje virtualizimi në të njëjtën kohë, duhet që çdo utilitet menaxhimi t'i njohë makinat virtuale ose do të duhet të kufizohet përdorimi i tyre vetëm në ato sisteme me të cilat mund të komunikojnë. Ky mund të mos jetë problem por duhet marrë në konsideratë.

3.2.6 Identifikimi i kandidatëve për virtualizim

Qëllimi i shumicës së përpjekjeve të virtualizimit është konsolidimi i shumë server-ave në platforma specifike hardware. Hapi i parë është është të konsiderohet software që duhet mbështetur. Identifikimi i makinave hardware me pjesë periferike të vjetra ose të veçanta është një hap i dytë shumë i rëndësishëm në planin e progresit duke dhënë një mekanizëm të shkëlqyer për kontrollin e hoste-ve fizike software-i i të cilëve planifikohet të kalohet në makina virtuale, por përgjithësisht është më efektive ta nisësh planifikimin e virtualizimit nga identifikimi i software-it të nevojshëm dhe kërkesat për mbajtjen e tij.

Identifikimi i software-it ekzistues që mund të lëvizë nga një host fizik në makina virtuale përfshin një numër faktorësh të ndryshëm, duke përfshirë kërkesat hardware,

sistemin operativ dhe kërkesat software të mjedisit të ekzekutimit, modelin e përdorimit të software-it dhe ngarkesën që do të vendosë në host-et e tij.

Mbledhja e hardware-it, sistemeve operative dhe të dhënave të mjedisit të ekzekutimit kryhet më lehtë me një listë të ndonjë forme për t'u siguruar që po mblidhen të njëjtat lloje të dhënash për të gjithë aplikacionet dhe sistemet që do të virtualizohen. Një skedar me kolonat e duhura është zakonisht modeli më i mirë për mbledhjen e këtij informacioni sepse mundëson renditjen sipas disa kolonave dhe identifikimi i software-it me sisteme fizike host që të kombinohen në një makinë virtuale të vetme.

Lista e mëposhtme tregon informacionin që duhet mbledhur për secilin nga aplikacionet që do të kalohet në makinë virtuale. Kjo jo vetëm që do të japë një referencë të mirë për informacion suporti dhe licencimi por mund të ndihmojë edhe të identifikohen aplikacionet që janë kandidatë për tu kaluar në makina virtuale.

- ***Aplikacioni dhe versioni*** – Emri dhe numri duhet të jetë specifik i versionit të aplikacionit.
- ***Sistemi operativ aktual dhe versioni*** – Sistemi operativ në të cilin aplikacioni ekzekutohet.
- ***Korrigjimet apo paketat e shërbimit të sistemit operativ*** – Çdo korrigjim specifik që është aplikuar te sistemi operativ përfshirë dhe paketat e shërbimit për aplikacionet e bazuar në Windows.
- ***Sisteme operative të tjera të pranueshme –*** Sisteme të tjera operative dhe numrat shoqërues të versioneve që ky version i këtij aplikacioni mund të përdorë.
- ***Mjedisi i ekzekutimit të software-it –*** Librari ndihmëse apo paketa software për të cilat aplikacioni mund të ketë nevojë duke përfshirë dhe numrat e versioneve. Kjo përfshin software si makinën virtuale të Java apo

GNU Java runtime mjedise *skripting* si Perl, awk, sed etj. interpretuese dhe programe ndihmëse me të cilat aplikacioni mund të nisë komunikimin.

- ***Privilegjet dhe/ose përdoruesit e kërkuar*** – Privilegjet e kërkuara për instalimin dhe ekzekutimin, duke përfshirë dhe çdo përdorues apo grup specifik që mund të përdorë software-in.
- ***Hardware dhe driver-at shoqërues*** – Çdo driver që është i shoqëruar me këtë version të aplikacionit si një kartë specifike video, kartë zëri, ndërfaqe rrjeti apo pajisjet e ruajtjes. Nëse ky hardware do të kërkonte drivera special nga prodhuesi hardware duhet të vënë re burimin dhe numrin e versionit të tyre. Nëse hardware përdor drivera standardë sistemi është mirë që të vihen re për tu siguruar që janë të vlefshëm dhe të mbështetur për çdo sistem operativ tjetër te i cili mund të kalohet aplikacioni. Gjithashtu duhen parë dhe kërkesat për rezolucionin e videos.
- ***Rezolucioni akutal i videos*** – Rezolucioni i videos me të cilin po punon aplikacioni aktualisht. Ky rresht i listës mund të shënohet N/A për serverët apo software-in command-line.
- ***Kërkesat për memorie*** – Të gjithë kërkesat specifike për memorie që lidhen me aplikacionin. Këtu duhet të përfshihen limitet në sasinë maksimale të informacionit që mund të përdorin server-at apo aplikacioni.
- ***Memoria në sistemin aktual host*** – Sasia e memories që është e vlefshme në sistemin në të cilin po ekzekutohet ky aplikacioni. Ky informacion mund ta përdoret për të përcaktuar madhësinë e memories së makinës virtuale nëse aplikacioni nuk ka kërkesa eksplicite për sasi të memories.
- ***Performanca aktuale e aplikacionit*** – Perceptimi se sa mirë ekzekutohet dhe performon aplikacioni në hostin aktual. Edhe pse nuk është një matje empirike, ky informacion mund të përdoret që të ndihmojë në vlerësimin e kërkesave për memorie dhe procesor të secilës makinë virtuale te e cila

kalohet aplikacioni. Kjo mund të ndihmojë përcaktimin për shembull nëse do të lidhet një makinë virtuale me një procesor specifik për të minimizuar mbingarkesën administrative .

- ***Liçensimi aktual*** – Nëse aplikacioni kërkon licencë apo jo, e nëse është kështu tipin e licencës: për kopje, notuese, e kyçur në nyje (e kyçur për një sistem specifik bazuar në një identifikues bordi, IP-je rrjeti apo adrese MAC), e kështu me radhë.
- ***Liçencimi i virtualizimit*** – Nëse aplikacioni mund të përdoret në një mjedis të një makine virtual. Kjo duhet të përfshijë gjithashtu informacion në lidhje me kërkesën për ndryshime apo jo në licencë, blerjen e një licence të re etj.

Siç mund të shihet mbledhja dhe organizimi i gjithë këtij informacioni kërkon njëfarë kohe, por kjo do të thjeshtojë identifikimin e aplikacioneve që mund (ose nuk mund) të kalohen në një makinë virtuale dhe karakteristikat e veçanta që makina virtuale duhet të ketë që të japë një mjedis të suksesshëm dhe të fuqishëm ekzekutimi për aplikacionet dhe server-at .

3.3 Përmbledhje

Në këtë kapitull u dhanë disa avantazhe dhe disavantazhe të teknikës së Virtualizimit. Avantazhet kryesore të kësaj teknike lidhen kryesisht me sigurinë e të dhënave. Siguria e tyre konsiston në mundësinë e lëvizshmërisë së tyre nga një Data Center në një tjetër, ku në shumicën e rasteve, japin shumë pak ose aspak shqetësim për përdoruesin. Një tjetër avantazh është mundësia për të balancuar ngarkesën midis burimeve. Kështu gjatë një aktiviteti, nëse një burim është shumë i ngarkuar, p.sh. CPU-ja e një Serveri që po ekzekuton një aplikacion në të, mund të realizohet zhvendosja e ngarkesës drejt CPU-ve të Serverave të tjerë në atë Cluster duke formuar fenomenin “High Availability”.

Një tjetër avantazh i kësaj teknike është toleranca ndaj gabimeve. Kjo do të thotë që nëse një Sistem Operativ bie ose infektohet nga një virus, kjo nuk do të shkaktojë shqetësim për kopjet e tjera të ngritura mbi atë makinë fizike ose në makina të tjera.

Disavantazhi i vetëm i kësaj teknike është "*Single Point of Failure*", që do të thotë se nëse makina fizikisht "bie", do të rrezohen dhe të gjitha kopjet e ngritura mbi atë makinë. Perveç saj mund të thuhet se efiçensa në kohën e transmetimit, shfrytezimit të burimeve, kohës së shkrimit apo leximit në disk është shumë herë më e ulët se në kushte reale.

KAPITULLI I KATËRT

HYPERVISORËT E NDRYSHËM. KARAKTERISTIKAT E TYRE.

4.1 Këndvështrim i përgjithshëm i Xen dhe Virtualizimit x86

Të mirat e virtualizimit në komoditetin e hardware-it janë:

- Të jetë i aftë për të ekzekutuar makina virtuale të shumta (VMs) në një pjesë të vetme të hardware-it, në izolim nga një makinë tjetër kështu që secili mund të përballoje proçeset e veta, adresën unike, aksesin, dhe mbylljen (kyçjen) e sistemit të burimeve , të menaxhohet dhe administrohet në mënyrë të pavarur.
- Tu mundësojë makinave të shumta virtuale të ngarkohen në një pjesë të vetme të hardware-it fizik, duke ndarë aksesin dhe menaxhimin e burimeve në atë hardware fizik pa humbje në performancë nëpër një makinë të vetme e cila ngarkon një instancë të vetme të dedikuar të një sistemi operimi.
- Për të lejuar sistemet e operimit të pamodifikueshëm të ngarkohen si makina virtuale në një sistem hosti.
- Për të qenë të aftë të administrojë në mënyre të qëndërzuar makinat virtuale të izoluara brenda kontekstit të një makine fizike ose një infrastrukture IT, duke i bërë të mundur administratorit të rindajë dinamikisht burimet ose madje të transferoj makinat virtuale nga një host në një tjetër pa ndërprerje në shërbim.

Të gjitha proçeset e hapsirës së përdoruesit në një sistem të ngjashëm me UNIX ekzekutohen në lidhje me një direktori rrënjë, e cila paraqet nivelin bazë të sistemit të skedarëve. Përafrimi më i thjeshtë për të izoluar bashkësi te ndryshme të proçeseve nga njëri -tjetri është të ekzekutosh bashkësi të ndryshme të proçeseve në lidhje me

direktorine root të tyre. Kjo bëhet duke perdorur një thirrje sistem dhe nje program“perfshires” të lidhur me të, të njohur si “chroot” i cili mund të ekzekutohet vetem nga përdoruesi root. Nje thirrje “chroot” ndikon në mjedisin e ekzekutimit të proçeseve koherente dhe të gjithë miniproçeseve që i pasojnë. Sapo ekzekutojmë një thirrje chroot, të gjitha libraritë, direktoritë e perkohshme, nyjet e pajisjeve, privilegjet e sistemit dhe skedaret e konfigurimit dhe kështu me rradhë duhet te jenë të vlefshme në lidhje me direktorinë që sapo i bëmë chroot. Ky është përafrim i serverit virtual i paraqitur nga FreeBSD jails dhe është përafrimi bazë mbrapa produkteve të virtualizimit të nivelit të sistemit të operimit.

Nqs bëhet në mënyre të saktë, përafrimi i *chroot* siguron një mënyre të lehtë të izolimit dhe menaxhimit të seteve të ndara të proçeseve hierarkike, por të gjitha sistemet ende ndajnë një bërthame të vetme dhe të shoqëruar me setin e burimeve fizike. Megjithëse mund të rregullojmë prioritetet individuale të këtyre proçeseve, s’ka ndonjë mënyre për të lidhur burimet specifike me ato.

Sapo avantazhet e përafrimit me *chroot* ezaurohen, një teknik e zakonshme është modifikimi (futja e teknikave te reja) në izolimin e performances në sistemin operativ që host-on serverat virtual. Ky është një përafrim i marrë nga shumë oferta të serverave virtual të tilla si: FreeVPS,Linux-|Vserver, OpenVZ, Virtuozzo, Solaris Containers dhe Zones, dhe kështu me radhë.

Të gjitha zgjidhjet e diskutuara deri tani janë vetëm për UNIX dhe zgjidhje me një bërthame. Duke qënë zgjidhje vetëm për UNIX nënkupton që të gjitha këto kërkojnë sisteme thirrjeje, sisteme filesh dhe strukturë të dhënash të proçeseve, sipas UNIX/Linux/BSD. Duke qënë zgjidhje me kernel të vetëm nënkupton që një kernel i vetëm i sistemit operativ është duke menaxhuar të gjitha kërkesat tek ose nga makinat virtuale dhe arbitrimin midis tyre. Të dyja këto fakte na paraprijnë nga ngarkimi i sistemeve të tjera të operimit në makinat virtuale për dy arsye kryesore: vetëm proçeset natyrale të UNIX/Linux/BSD mund të ekzekutojnë natyrshëm në një kernel UNIX/Linux/BSD, dhe të ngarkojë sisteme të shumta dhe të plota operimi në të njëjtën

kohë në komoditetin e hardware-it x86 i cili mundet të dështojë shumë shpejtë për shkak të konflikteve në mënyrat e privilegjimit në të cilat instruksione të ndyshme të sistemeve të operimit duhet të ekzekutohen.

4.1.1 Niveli i mbrojtjes së X86: Një unazë për ti sunduar të gjitha

Proçesorët zakonisht punojnë në mënyra mbrojtjeje të ndryshme për të parandaluar një akses të paautorizuar tek proçesori fizik dhe burimet e pajisjeve. Proçesoret origjinal x86 paraqesin një menyrë të vetme të operimit, e njohur si real mode. Sa më shumë aftësi te sofistikuara, të tilla si adresime shtesë të memorjes dhe aftësi menaxhimi iu shtuan bashkësisë së instruksioneve x86 32-bit, mënyra të ndryshme operimi u përcaktuan për proçesorin. Këto bëjnë diferencën midis tipeve të aksesit të burimit që nevojiten kundrejt atyre që janë të lejuara por që nuk nevojiten nga lloje të ndryshme proçesesh. Më të njohurat janë:

- **Real mode(Mënyra reale):** Një proçes ka akses direkt në të gjitha aspektet e hardware-it, por është i limituar në kapacitetin e adresimit të memorjes në një proçesor 8088.
- **System management mode(Mënyra e menaxhimit të sistemit) (SMM):** Një mënyre 16-bitëshe që siguron akses direkt dhe afatshkurtër të burimet hardware për qëllime të brendshme të mirëmbajtjes së sistemit.
- **Protected mode(Mënyra e mbrojtur):** Nivele të shumta mbrojtjeje ,të njohura si unaza, janë të mbështetur në mënyrë që të bëhet dallimi midis tipeve te aksesit të memorjes dhe aksesit të hardware-it të përgjithshëm, të cilat kërkohen nga një sistem operativ dhe nga aplikacionet e nivelit të lartë. Në “protected mode”, proçesorët modern Intel x86 dhe proçesorët kompatibël me to favorizojnë katër nivele mbrojtjeje të njohura si unaza, të cilat janë numëruar nga 0 deri ne 3 në traditën më të mirë te gjuhës së programimit C.

Ekzistojnë edhe të tjera mënyra te tilla si “unreal mode”, “virtual real mode” (virtual 8086 mode) dhe “long mode”. Ky seksion ofron një prezantim të nivelit të lartë ndaj implikimeve të unazave të ndryshme që mundësohen nga “protected mode” dhe diskuton tipin e kodit që ekzekutohet në çdo unazë dhe aftësitë e tij hardwer-ike dhe të memorjes.

Në mjediset standarte të sistemeve të operimit, bërthama e sistemit të operimit ngarkohet në unazën 0 dhe mund të ekzekutojë kështu të gjitha instruksionet e privilegjuara, të menaxhojë memorjen, dhe të specifikojë në mënyrë direkte rangun e adresave të memorjes te disponueshme në një sistem operimi. Aftësia për të menaxhuar memorjen përtej kapacitetit të adresimit fizik të një proçesori 32-bit x86 nëpërmjet përdorimit të segmenteve të memorjes ishte një nga motivet primare për mënyrën “protected mode”. Unazës 0 zakonisht i referohemi si “supervisor mode” ose “kernel mode” sepse ajo kontrollon të gjithë aksesin direkt në hardware dhe memorje. Aplikacionet dhe proçeset e serverit zakonisht ekzekutohen ne unazën 3, të njohur si “user mode”. Unazat 1 dhe 2 janë zakonisht të papërdorura në mjediset e sistemit të operimit vanilla, përveç atyre që përdorin akoma sistemin OS/2 të IBM.

Në brendësi mbrojtja në nivelin e kontrollit aktivizohet sa herë që ngarkohet një segmenti i ri kodi. Kodi që është duke u ekzekutuar kur një proçesor hyn ne “protected mode” ka një nivel aktual privilegji 0 (CPL), dhe prandaj ngarkohet në unazën 0. Secili segment kodi që ngarkohet më pas ka një përshkrues të nivelit të privilegjeve prej dy bitesh(DPL) që identifikon nivelin e privilegjit të atij segmenti. Një segment kodi asnjëherë nuk ngarkon një tjetër që ka nje DPL me më shumë privilegje se sa CPL. Kjo do të thotë që niveli aktual i mbrojtjes duhet të jetë gjithmonë më i vogël ose i barabartë me nivelin e mbrojtjes të çdo segmenti të ri kodi që ai ngarkon. Përpjekja për të shkelur këtë rregull shkakton një përjashtim që nuk e përfill segmentin në fjalë. Kodi që ekzekutohet ne unazën 0 është përgjegjës për caktimin e niveleve të mbrojtjes të secilit segment kodi.

Për të ndërmjetësuar aksesin hardware midis makinave virtuale, Xen ekzekuton hypervisorin e tij në unazën 0, ndërsa kernelet e modifikuar ekzekutohen në unaza më të sipërme dhe më pak të privilegjuara (Aplikacionet ende ngarkohen ne unazën 3). Në hardware 32 bit, kernelet Xen të modifikuar ngarkohen ne unazën 1, ndërsa në sistemet 64 bitëshe, kernelet e modifikuar ngarkohen në unazën 3 bashkë me aplikacionet sepse nuk është e mundur të mbrosh unazën 0 nga proçeset që ekzekutohen në unazën 1. Fakti që Xen ngarkohet në një unazë të një niveli më të lartë se "supervisor mode" është edhe origjina e termit "hypervisor".

4.1.2 Virtualizimi dhe nivelet e mbrojtjes së X86

Problemi klasik me disa instruksione të privilegjuara x86 është që ato nuk ngecin kur ekzekutohen me privilegje të pamjaftueshme, por dështojne duke destinuar kështu instruksionet që vijojnë të ekzekutohen gjithashtu në një menyre të gabuar ose në gjendje të panjohur. Zgjidhjet e Sistemeve të operimit Guest (Guest-OS solutions), të tilla si VMware Workstation, Parallels Workstation, dhe VirtualBox, na lejojnë të ngarkojmë istanca të pamodifikuara të sistemit të operimit njëkohesisht, sepse mjedisi i aplikimit në të cilin makinat virtuale janë ngarkuar monitoron në mënyrë të vazhdueshme instruksionet që makinat virtuale po përpiqen të ekzekutojnë, dhe rishkruan instruksione të privilegjuara që do të dështonin, duke i zëvendësuar me implementime lokale që nuk ngecin por bëjnë gjënë e duhur.

Teksa paketat e virtualizimit Guest-OS janë zgjidhje të shkëlqyera për desktop kur kemi nevojë të ngarkojme sisteme operimi të shumta, të pamodifikuara X86 në një sistem të vetëm fizik, e meta e kësaj zgjidhjeje është e qartë. Zgjidhjet e virtualizimit GuestOS kërkojnë që sistemi të ekzekutojë një sistem të plote operimi. Si aplikacioni i virtualizimit dhe çdo sistem operativ i pamodifikuar i ngarkuar brenda tij duhet të konkurrojnë me sistemin primar te operimit dhe me njëri-tjetrin për burimet. Përfundimisht, duke ekzaminuar grupin e instruksioneve nga çdo sistem operimi virtual dhe rishkrimi i tij kur është e nevojshme kërkon fuqi proçesuese të konsiderueshme.

Një alternative tjetër virtualizimi me performancë te lartë që suporton shumë sisteme operimi të ndryshme në një makine të vetme është të boot-osh një monitorues të vogël makine virtuale (virtual machine monitor) në hardware në vend të një sistemi operimi të një niveli më të lartë dhe që kërkon më shumë burime. Monitori ekzekuton direkt ne hardware, menaxhon te gjithë nivelin e ulët të shpërndarjes së burimeve për makinat virtuale, dhe siguron ndërfaqe që aplikacionet administrative të niveleve të larta mund të ngecin (hook into) per të kryer detyra të ndryshme administrative. Monitoruesit e makinave virtuale quhen shpesh hypervisor sepse ata ekzekutojne në mënyre efiçente e me më shume privilegje se sa kodi i supervisorit x86 në një sistem operimi të nivelit të lartë. Ky është përafrim i marrë nga zgjidhje te virtualizimit si VMware ESX Server dhe Xen. Hypervisori i përdorur nga VMware ESX Server kryen të njëjtin lloj të monitorimit dhe menaxhimit të grupit të instruksioneve të ndjekur nga produkti i Workstation-it VMware, por siguron kryesisht performancë më të lartë sepse ekzekutohet në nivel më të ulët qe është me afër hardware-it dhe nuk konkurron me proçese të nivelit të lartë.

Nga ana tjetër, hypervisori i përdorur nga Xen është shumë i ndryshem. Kur ekzekutojmë në hardware që nuk siguron mbështetje per virtualizim, hypervizori Xen kërkon qe makinat virtuale të ngarkuara në krye të hypervizorit të tij të përdorin një version të personalizuar të sistemit të operimit. Këto versione të personalizuara janë modifikuar për të shmangur instruksionet e privilegjuara duke përdorur një model të hardware-it abstrakt që ndryshon nga hardware-i specifik që është i vlefshëm në makinën fizike dhe që rrjedhimisht ekzekutohet në një nivel më të ulët privilegjesh se hipervisori. Hypervizori trajton menaxhimin e memories, mbarëvajtjen e CPU-së, thirrjet sistem, ndërprerjet e hardware-it, timer-at, dhe të gjitha pajisjet direkte I/O. Ky përafrim në virtualizim eshte i njohur si ‘’paravirtualizim’’ sepse nevojiten modifikime tek një sistem operimi në mënyre që te trajtohen operacionet e privilegjuara duke komunikuar me hypervizorin ose një kernel administrativ dhe të modifikuar Xen me më pak privilegje.

Ndryshimet e sistemit të operimit që kerkohen për nderveprim me hypervizorin nuk e ndryshojnë Ndërfaqen Binare të Aplikacioneve të nivelit të lartë te

kernel-it, prandaj kerneli dhe kodi kernel si driverat e pajisjeve janë të vetmet që kërkojnë modifikim per paravirtualizim. Kjo gjithashtu do të thotë se nuk kërkohen ndryshime te aplikacionet përdorues që ato të ekzekutohen në një mjedis të virtualizuar, çka përbën një aspekt kyç të një mjedisi të suksesshëm virtualizimi.

4.2 Domain-et Xen dhe Hypervisori

"Domain-i Xen" është një instancë specifike e një makine virtuale Xen që ekzekutohet në një pjesë fizike specifike të hardware-it. Xen suporton dy lloje bazë të domaineve, me përdorime dhe kapacitete të ndryshme.

Xen-i menaxhon aksesin në memorje dhe burimet hardware nëpërmjet një kombinimi te hypervisorit te tij dhe një kerneli Xen të modifikuar me privilegje specifike që përdoret per te menaxhuar, monitoruar, dhe administruar të gjitha makinat e tjera virtuale Xen që ekzekutohen ne nje pjese specike te hardware-it. Ky kernel i privilegjuar Xen është i njohur si domain0 (ose domain-O). Kerneli domain0 dhe sistemi Linux qe ngarkohet ne te hoston një Xen "daemon"(/usr/sbin/xen, e shkruajtur ne gjuhen e programimit Python) dhe software administrativ që përdor "daemon-in" Xen për të komunikuar me hypervisorin. Xen siguron një aplikacion administrativ command-line, të njohur si xm (Menaxhimi i Xen), në te cilën fillojmë komanda për të krijuar, lidhur, dhe fikur domain-e te tjera, dhe për të kontrolluar sasine e memorjes qe i është caktuar një domain-i, parametrat shoqërues të skedulimit e kështu me radhë.

Domain-et e inicuar nga një sistem domain0 janë të njohur si domain-e guest, domain-e të pa privilegjuara, ose thjesht domain-e U. Kur një sistem domain0 krijon, lidhet tek, modifikon, ose ndërpret ndonjë domainU, ai perdor një skedar konfigurimi që siguron informacion të detajuar rreth domain-it guest, të tilla si memorja fizike qe është caktuar fillimisht tek ai domain, aksesi që domain-i guest ka tek disqet fizike dhe pajisjet e rrjetit ne sistemin e domain0, si rrjeti është i konfiguruar ne domain-in guest, ndonjë suport grafik mund të përdorim per tu lidhur me panelin e komandimit te domain-it guest e kështu me radhë.

Kur Xen doli për herë të parë, kernelët e përdorur nga sistemet domain0 dhe domainU duhet të kompiloheshin me opsione konfigurimi të ndryshme te kernelit në mënyre që të bëheshin dallime midis tyre. Sot një kernel Xen standart mund të detektojë nëse ai është ngarkuar nga hypervizori ose nëpërmjet një sistemi domain0, dhe sillet në përputhje me rrethanat.

4.2.1 Ndërveprimi me hypervisori-in

Një nga qëllimet e Xen ka qënë gjithmone të ndajë kërkesat e implementimit nga vendimet per politikat, duke ia lënë opsionet e administrimit dhe konfigurimit sistemit domain0 sesa lidhja e tyre ne hypervisor. Ky është nje faktor i rëndësishem lidhur me fuqine dhe popullaritetin e Xen. Alokimi dhe menaxhimi i CPU-së dhe i memorjes bëhet nga hypervisori sepse kjo është një kërkese fizike për të ngarkuar makina virtuale të shumta në një sistem të vetëm fizik. Në mënyrë të ngjashme, hypervizori është përgjegjës për krijimin, menaxhimin, dhe fshirjen e ndërfaqeve virtuale të rrjetit dhe të pajisjeve virtuale bllok të shoqëruara me çdo domain guest, sepse secila nga këto përmban informacion kontrolli të aksesit qe është i nevojshëm për të ekzekutuar makina virtuale në një platform fizike hardware të vetëm duke mos dëmtuar kontrollin dhe politikat e tyre, të cilat duhet te jenë të ndara.

Domain-et përdorin një thirrje të veçantë sinkrone të njohur si hypercall e cila funksionon si një software “trap” për të kërkuar operacione të privilegjuara në hypervisor, pak a shumë si një sistem thirrjeje tradicional. Përgjigjet nga hypervisori kthehen në menyrë asinkrone nëpërmjet një mekanizmi sepse hypervizori mund të jetë duke shërbyer hypercall-e nga shumë domain-e në një kohë të caktuar. Këto evente punojnë në mënyrë shumë të ngjashme me sinjalet tradicionale të UNIX duke mbështetur si njoftimin e ngjarjeve të lidhura me të dhënat si përfundimi i një I/O-je apo ardhja e të dhënave nga rrjeti, dhe njoftime administrative të nivelit të lartë për gjëra si mbyllja e sistemit apo kërkesa për migrim.

4.2.2 Kontrollimi i Skedulimit të Hypervisor-it

Një skeduler është nje pjesë e një sistemi operimi që përcakton cilat proçese janë te disponueshme për ekzekutim dhe cilat duhet të ekzekutohen si dhe në cilën rradhë. Në rastin e nje skeduleri hypervisor, skeduleri përcakton cilat makina virtuale duhet të marrin CPU-në fizike dhe resurset e sistemit, me cilat CPU fizike ato duhet të lidhen dhe në cilën renditje. Skedarët e konfigurimit për domain-et Xen sigurojnë një mekanizëm për të lidhur një ose më shumë CPU në domain-et specifike të Xen-it në menyrë që të garantojnë fuqi proçesimi për domainet Xen që po ekzekutojnë shërbime me ngarkesë të lartë. Për më tepër, hypervisori Xen suporton një API standarte për skedulerat që ai përdor në brendësi për balancimin e ngarkesës dhe kërkesat për shërbime nga domain-e të shumëfishta. Hypervisori Xen vjen me skedulera të shumtë nga të cilët mund të zgjedhësh duke specifikuar emrin e një skeduleri specifik si një opsion të rreshtit të komandave ndaj opsionit të boot-imit të hypervisor-it në skedarin e konfigurimit për bootloader-in tuaj. Nqs përdorim bootloader-in GRUB, skedari i konfigurimit është skedari */boot/grub/menu.lst.*

Një skeduler mund të specifikohet duke përdorur opsionin sched=emri në rreshtin e komandave të boot-imit për hypervisor-in Xen, ku emri identifikon skedulerin që dëshirojmë të përdorim. Xen 3.x suporton skedulerat e mëposhtëm:

- **sedf:** Simple Earliest Deadline First , i cili suporton bashkëpërdorim të peshuar të CPU-ve dhe përdor një rradhë prioriteti për të analizuar kërkesa nga domaine të ndryshme për tu përpjekur që të garantojë reagueshmëri. Mund t'i vendosim parametrat për këtë skeduler nga komanda xm duke përdorur komanden xm sched-sedf duke identifikuar domain-in për te cilin dëshirojmë t'i vendosim politikat, të ndjekur nga 5 parametra:
  - *period*: Perioda maksimale e skedulimit në nanosekonda.
  - *slice*: pjesëza e kohës që duhet shoqëruar me çdo kërkesë në nanosekonda.

- *hint*: Një vlerë opsionale e shkallëzimit për periodën kur ka shumë I/O në progres.
- *extra*: Një flamur (flag) që tregon nese një domain mund të zgjerojë intervalin e kohës.
- *weight*: Pesha relative e një CPU virtuale per domain-in specifik.

- **credit:** Një skeduler me ndarje të drejtë proporcionale që përshtat në mënyrë dinamike prioritetin e CPU-së dhe domain-it duke matur përdorimin. Mund të vendosim parametrat për këtë skeduler nga rreshti i komandave xm duke përdorur komanden xm sched-credit (ose nëpërmjet mjeteve të tjera menaxhimi të Xen-it) duke identifikuar domain-in për të cilin dëshirojmë t'i vendosim një politikë të ndjekur nga dy parametra:
  - *weight*: Prioriteti relativ i një domain-i specifik. Vlerat variojnë nga 1 deri 65535, default-i eshte 256
  - *cap*: Maksimumi që një domain mund të konsumojë nga një CPU edhe nëse sistemi host ka cikle bosh të CPU-së. CAP shprehet si një përqindje e një CPU-je fizike. Duke përdorur vlera më të mëdha se 100 mund të specifikohet më shumë se një CPU. Një vlerë 0 nënkupton mungesë kufizimi.

Versione më të hershme të Xen suportonin skedulerë të tjerë të hypervisor-it si bvt (Borrowed Virtual Time) dhe rrobin (Round Robin), por këto janë në proçes heqjeje nga kodi bazë i Xen-it.

4.3 Llojet e makinave virtuale të suportuara nga Xen

Për shkak të popullaritetit të tij, fleksibilitetit, fuqisë, dhe modelit të liçensës open source, Xen është një paketë software-ike e cila zhvillohet në mënyre konstante dhe të shpejtë. Kombinuar me zhvillime të ngjashme në kapacitetin e proçesorëve x86 të tregut

dhe hardware-it të lidhur me to, statusi i Xen (dhe statusi i virtualizimit x86 në pergjithësi) është vështirë per tu percaktuar .

Një shembull i mire i kësaj është zgjerimi, nga i cili Xen mund të suportoje të dyja makinat virtuale 32bit dhe 64bit të llojeve të ndryshme ne një sistem të vetëm. Përpara se suporti i hardware-it për virtualizimin ishte shtuar në proçesorët me kapacitete Intel VT (Virtualization Technology) dhe AMD SVM (AMD Secure Virtual Machine), mund të ekzekutohen vetëm domain-e Xen 32 bitësh domainU në një host fizik, në të cilin po ekzekutohet një hypervisor Xen 32 bitesh dhe domain0. Kjo po ndyshon, por ka ende një numër problemesh të nivelit kernel që duhet të merren në konsideratë kur na duhet të ndërtojmë një infrastrukturë Xen dhe në mënyrë të veçantë, konsiderata ndër-makinë si migrimi i domain-it dhe failover-i.

4.3.1 Sistemet e paravirtualizuar

Xen përdor paravirtualizimin në makinat e tij virtuale për shkak të performancës dhe avantazheve administrative që ai siguron. Paravirtualizimi nënkupton që sistemi i operimit guest është modifikuar në mënyrë të tillë që ai mund të kryejë ndërfaqje me hypervisor-in dhe që operacionet e privilegjuara që sistemi i operimit do të kryej në unazën 0 të mbrojtjes të jenë të përkthyera në hyper-thirrje (hypercalls) te hypervisor-i. Hypervisori është përgjegjës për akses direkt tek hardware-i fizik i hostit, për menaxhimin e memorjes së ndarë dhe të përdorur nga secili domain, dhe për menaxhimin e disqeve virtuale dhe aksesin në rrjet nga të gjithë domain-et guest duke përfshirë domain0, domain-in me privilegje administrative të Xen.

Sistemet x86 dhe ata të ngjashëm me to kanë pasur pengesa për shumë vite nga problemet e lidhura me aksesin e memorjes. Edhe në sisteme Linux që përdorin kernel Linux 2.6, kerneli në përgjithësi mund të aksesoje vetëm 4GB të memorjes kryesore, dhe aplikacionet janë përgjithsisht të limituara në një sasi të vogël. Për të zgjidhur këtë problem, Intel-i paraqiti një bashkësi të Zgjerimit të Adresave Fizike (e njohur si PAE) që lejon sistemet moderne të aksesojnë deri në 64GB të memorjes kryesore duke e parë

memorjen fizike deri në 16 rangje 4 GB. Si kerneli dhe aplikacionet duhet të jenë kompiluar në një formë të veçantë për të qenë të aftë të përdorin këtë model memorjeje, por duke bërë këtë disponohet një sasi memorje relativisht e madhe në dispozicion të sistemit dhe aplikacioneve duke rritur performancën, kapacitetin dhe shkallëzimin në përgjithsi.

Kur ekzekutojmë domain-e Xen në hardware fizik që nuk sigurojnë suportin e hardware-it për virtualizim, te gjithë kernelet për të gjithë domain-et duhet të perdorin të njëjtin model memorjeje dhe të njejtën madhësi instruksionesh. Nqs hypervisori dhe kerneli i domain0 i paravirtualizuar që boot-ohet jane 64 bit, të gjithë kernelet e tjerë të ngarkuar në atë makine duhet të jenë gjithashtu 64 bit. Nqs hypervisori dhe kerneli domain0 i paravirtualizuar që boot-ohet janë 32 bit dhe përballon memorjen e zgjeruar PAE, të gjithë kernelat e tjerë të ngarkuar në atë makinë duhet të jenë gjithashtu kernel-a PAE 32 bitëshe.

Kjo ndryshon në mënyrë drastike nqs sistemi është duke përdorur një proçesor që ka suportin e hardware-it për virtualizim, të tillë si proçesorët me suport te Intel VT dhe AMD SVM. Këto proçesor lejojnë ngarkimin e sistemeve të operimit guest të pamodifikuar, dhe gjithashtu zgjerojnë llojet e domain-eve të paravirtualizuar që mund të ngarkohen në hardware.

4.3.2 Sistemet Guest të pamodifikuar

Nqs sistemi është duke përdorur një proçesor që ka suport hardware për virtualizim, të tillë si proçesoret Intel VT dhe AMD SVM, mund të ekzekutojmë sisteme të pamodifikuar operimi si makina virtuale nën domain0. Suporti hardware për virtualizim na lejon të ekzekutojmë Windows-in si një domain në një host Xen, dhe gjithashtu mund të perdorim një disk instalimi “vanilla” Linux dhe instalojmë një distribucion standart Linux si një makinë virtuale pa modifikim. Sot proçesorët qe sigurojnë suportin e hardware-it për virtualizim janë këto:

- **Intel VT (Virtualization Technology, njohur si Vanderpool):** Zgjodhi proçesorët Pentium 4 dhe Pentium D, Xeon 5000 dhe të mëvonshmit, Xeon LV, Core Duo, Core 2 Duo, dhe Core 2 Quad.
- AMD-V/SVM (Virtualization/Secure Virtual Machine njohur si Pacifica): Zgjodhi proçesorët Athlon, Opteron, dhe Turion Socket F dhe AM2.

Për të përcaktuar nëse proçesori në një sistem të dhënë përballon virtualizimin, mund të kontrollojmë hyrjet e flamujve në skedarin /proc/cpuinfo i cili ka pamjen e mëposhtme:

 # cat /proc/cpuinfo | grep flags

*flags : fpu tsc msr pae mce cx8 apic mtrr mca cmov *

*pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm *

syscall lm constant_tsc pni monitor ds_cp1 est cid cx16 xtpr

Nqs përdorim një CPU të Intelit do të shikojmë për fushen vms në fushën e flamujve. Nqs përdorim një CPU AMD do të shikojmë për fushën vmx ne fushën e flamujve.

Pavarsisht nëse CPU-ja e mbështet virtualizimin hardware apo jo, përcaktuesi përfundimtar i sistemeve të përgjithshme operative që mund të ekzekutohen në një host specifik domain0 të Xen është skedari /sys/hypervisor/properties/capabilities. Hyrjet në sistemin e skedarëve /sys japin informacion hardware dhe të nivelit të sistemit për gjendjen e sistemit Linux. Mund të përdorim komandën standarte *cat* për të parë përmbajtjen e këtij skedari, si në shembullin e mëposhtem nga sistemi intel VT 64 bitësh:

 # cat /sys/hypervisor/properties/capabilities

Xen-3.0-x86_64 hvm-3.0-x86_32 hvm-3.0-x86_32p hvm-3.0-x86_64

Përmbajtja e skedarit thotë që ky sistem mund të ekzekutojë domain-e të paravirtualizuar 64 bitësh (xen-3.0-x86_64), sisteme operimi të pamodifikuar 32 bitësh si

makinë virtuale hardware (hvm-3.0-x86_32), domaine të paravirtualizuar 32 bitësh si makina hardware virtuale (hvm-3.0-x86_32p), dhe sisteme operimi të pamodifikuar 64 bitësh si makina virtuale hardware (hvm-3.0-x86_64). Një rregull i përgjithshem dhe mjaft i mirë është që sistemet me suportin e hardware-it për virtualizim që ekzekutojnë Xen-in janë përgjithsisht kompatibël me sistemet e vjetra. Me fjalë të tjera një sistem Xen me më shume kapacitete mundet përgjithsisht të ngarkojë një kernel Linux me kapacitet të njëjtë ose më të vogel. Një Xen kernel 32 bit me suport PAE mund të ekzekutojë një kernel Linux pa PAE në një makinë virtuale hardware. Një sistem Xen 64 bit mund të ekzekutojë çdo kernel Linux 32 bit në një makinë virtuale hardware.

4.3.3 Kombinimi i kernel-ave 32 bit dhe 64 bit, të sistemeve të skedarëve dhe aplikacioneve

Sikurse edhe hypervisori edhe kerneli domain0 janë përgjegjës për menaxhimin e ndarjes së memorjes dhe përdorimin nga domein-e te tjerë, mund të hasim konflikte kur përpiqemi te kombinojmë domain-e të paravirtualizuar 32 bit dhe 64 bit të ngarkuar në një sistem të vetëm host fizik. Në fakt problemi nuk është i lidhur me faktin nëse sistemi i skedarëve dhe aplikacionet janë 32bit ose 64 bit, por është specifike ndaj kernelit. Mund të përdorim një sistem skedarësh 32 bit me një kernel Xen 64 bit të paravirtualizuar pa ndonjë problem, për aq kohë sa çdo modul kerneli 64 bitësh që ai kërkon mund të ndodhet në sistemin e skedarëve root. (Sigurisht, kerneli do të ekzekutohet edhe pa ato, por periferikë specifik, hardware të brendshëm, dhe nënsisteme të lidhur me to ndoshta nuk do të jenë të përdorshëm.) Gjithashtu mund të ndërtojmë manualisht kernel 32 bitësh për ta përdorur me makina virtuale të paravirtualizuara në një sistem 64 bitësh, por kjo kërkon perpjekje shtesë dhe kohë mirëmbajtjeje.

Në mënyre të ngjashme, kernel-e Linux 64 bitësh përgjithsisht mund të ngarkojnë aplikacione specifike 32 bit nga një makinë 64 bit, duke supozuar që aplikacioni është i lidhur (linked) në mënyrë statike ose që ai përdor librari të shpërndara dhe që libraritë 32 bitëshe që përputhen janë instaluar në sistemin 64 bit. Shumica e

sistemeve Linux X86_64 sot ndjekin hierarkine standarte të sistemit të skedarëve (FHS) me njëfarë zgjerimi, e cila dikton qe libraritë 32 bit janë instaluar në /usr/liband dhe që librarite 64 biteshe janë instaluar në /usr/lib64.

4.4 Software të tjerë të përhapur virtualizimi

Xen është teknologjia e virtualizimit më e dobishme për Linux tani, por janë përdorur zgjidhje të tjera virtualizimi kohë përpara se Xen të ishte i vlefshëm.

Secila teknologji virtualizimi për Linux ka zgjidhjet dhe meritat e saj teknologjike, por njohja më shumë rreth alternativave na bind që Xen është zgjidhja teknike e duhur.

4.4.1 Free VPS

Free VPS është një nga zgjidhjet origjinale të virtualizimit në nivel sistemi operativ për Linux së bashku me projektin Linux – Vserver. Free VPS mundëson një sërë përmirësimesh në llogari sistemi, rrjeti, dhe një sërë zgjerimesh të tjera administrative. Free VPS mundëson izolim të plotë të sistemit të skedarëve dhe proçeson ekzekutimin në çdo server virtual, si dhe kufizimin e secilit server për burime të tilla si ngarkesa e rrjetit, kapaciteti i diskut dhe konsumi i memorjes. Free VPS gjithashtu mundëson kapacitete të ndara administrative për secilin server virtual dhe më shumë detyra administrative si backup dhe monitorim që mund të kryhen nëpër servera virtual të shumfishte ose të gjithë serverat virtual.

Njësoj si zgjidhjet e tjera të virtualizimit në nivel sistemi operativ, mangësia kryesore e Free VPS është se kufizon ekzekutimin e një instance të vetme të një kerneli të vetëm Linux, duke shkaktuar një pikë të vetme dështimi.

4.4.2 Makina Virtuale Kernel

Linux Torvalds pranoi korrigjime që përfshinin një teknologji fare të panjohur virtualizimi të quajtur Makina Virtuale bazuar në Kernel (KVM) nga një kompani e

panjohur [47]. Këto korrigjime u përfshinë në lëshimin 2.6.20 të linjës kryesore të kernelit Linux. Më kryesorja e një teknologjie virtualizimi në kernel si KVM, është User-Mode Linux (UML). Njësoj si UML, KVM mbështet krijimin dhe ekzekutimin e sistemeve virtuale Linux që ekzekutohen si proçes i ndarë në një sistem Linux. Ndryshe nga UML, KVM kërkon që sistemi i hostit fizik të përdorë një proçesor që mbështet Teknologjine e Virtualizimit të Intel apo Makinën e Sigurtë Virtuale të AMD (SVM/AMD – V).Këta janë të njëjtit proçesorë që kërkohen për ekzekutimin e sistemeve guest të pamodifikuar si Xen.

Një nga arsyet kryesore për përfshirjen e KVM në kernel-in e linjës kryesore (përveç implementimit të pastër) ishte fakti që KVM nuk kërkon hypervisor ndaj është një zgjidhje e pastër Linux, ku Linux është ende sistemi operativ që ekzekutohet. Në brendësi KVM punon përmes një pajisjeje nyje të quajtur /dev/kvm e cila krijohet dhe menaxhohet nga module kernel të ngarkueshme nga CPU specifike si intel_kvm.ko dhe amd_kvm.ko njëra nga të cilat ngarkohet si kvm.ko. Duke hapur këtë pajisje krijohet një makinë e re virtuale, e cila më pas mund të menaxhohet nëpërmjet thirrjeve ioctl(), e cila kryhen operacione si krijimi i një CPU-je virtuale, alokimi i memorjes, detektimi i thirrjeve të privilegjuara e kështu me rradhë. KVM kërkon gjithashtu një aplikacion të hapësirës së përdoruesit për të parë dhe ndërvepruar me secilën makinë virtuale.

KVM aktualisht mbështet makina virtuale Linux (x86 dhe x86_64) dhe makina virtuale Windows. Për momentin mbështeten vetëm makina virtuale Windows x86 32 bit, edhe pse puna në makinat virtuale 64 bit Windows është në progres.

Përfitimi kryesor i KVM është fakti që ndryshe nga Xen, nuk kërkon modifikim të kernelit Linux. Gjithsesi nuk mundëson përfitimet në performancë të sistemeve Linux të paravirtualizuar, por është i aftë të marrë avantazhe të elementeve standarte Linux si skeduleri Linux në vend të skedulerëve të ndryshëm që janë të ndërtuar në Xen.

4.4.3 VServer Linux

Linux – VServer është zgjidhja origjinale e virtualizimit në nivel sistemi operativ për Linux, duke i paraprirë projektit FreeVPS. Linux – VServer jep izolim të plotë të sistemit të skedarëve dhe proçeseve që janë duke u ekzekutuar në secilin server virtual, dhe kufizime për çdo server për burime si ngarkesa e rrjetit, hapësira e diskut dhe konsumi i memorjes. Linux – VServer jep gjithashtu aftesi të ndara administrative për secilin server virtual. Burimet e skedarëve apo të direktorive mund të shpërndahen midis serverave virtual duke përdorur linke të implementuar në mënyrë të veçantë që mund të krijohen me mjete speciale të quajtura vunify dhe vhashify të cilat e thjeshtojnë identifikimin e skedarëve dhe direktorive të përdorura gjerësisht.

Njësoj si zgjidhjet e tjera të virtualizimit në nivel sistemi operativ, disavantazhi kryesor i Linux – VServer është se kufizon ekzekutimin e një instance të vetme ne një kernel të vetëm Linux, duke dhënë një kufizim edhe një pikë të vetme dështimi.

4.4.4 Microsoft Virtual Server

Microsoft Virtual Server është një zgjidhje virtualizimi që na mundëson të krijojmë dhe të ekzekutojmë makina virtuale në versione 32 bit të sistemeve operative Windows XP dhe Windows Server 2003. Microsoft Virtual Server është zhvilluar nga Connectix dhe më pas u zhvillua nga Microsoft.

Puna për gjeneratën e ardhshme të teknologjisë së virtualizimit të Microsoft, me emër te koduar Viridian, po vazhdon në mënyrë aktive. Microsoft po bashkëpunon me Xen (nëpërmjet XenSource) në këtë përpjekje. Po ashtu Microsoft po bashkëpunon me distribucione Linux të fokusuara te Xen si Novell për të zhvilluar zgjidhje të bazuara në Xen për teknologjitë e tij të virtualizimit.

4.4.5 Open VZ / Virtuozzo

OpenVZ dhe Virtuozzo janë përkatesisht versionet open source dhe komerçial të një pakete software virtualizimi në nivel sistemi operimi. Virtualizimi në nivel sistemi

operativ mund të sigurojë përmirësime të konsiderueshme të performances ndaj zgjidhjeve të tjera të virtualizimit, kryesisht sepse një kernel i vetëm ekzekutohet në makinë dhe menaxhon të gjithë serverat virtuale në një host të vetëm fizik. Kjo redukton konsumimin e memories nga secili server virtual, dhe optimizon dhe racionalizon I/O brenda serverave virtual, sepse i njëjti kernel po kontrollon aksesin e pajisjeve dhe burimet. Për të njëjtat arsye zgjidhja me kernel të vetëm gjithashtu thjeshtëzon ndarjen e burimeve për servera të ndryshëm virtual, duke e bërë më të lehtë lidhjen e CPU-ve specifike, memories, dhe hapsirës së diskut me serverat virtual specifik. Një zgjidhje me kernel të vetëm gjithashtu thjeshtëzon ndryshimin e ndarjes së burimeve sipas nevojës teksa serverat virtualë janë në punë, pa nevojën për ti ristartuar. P.sh. numri i CPU-ve të lidhura me një makine virtuale është i kufizuar vetëm nga kerneli jo nga ndonjë kufizim specifik artificial te implementimit si VMware, Microsoft Virtual Server etj. Po kështu zgjidhja me kernel të vetëm ofron shumë mundësi për optimizime. P.sh. kashe-ja e kernelit shpërndahet midis të gjithë serverave virtual dhe kodet e aplikacioneve dhe librarive mund te shpërndahen në memorie. Kjo mundëson që OpenVZ dhe Virtuozzo të suportojnë efektivisht një numër të madh të serverave virtual në një host fizik te vetëm.

Përdorimi i një pakete virtualizimi në nivel sistemi operativ do të thotë se të gjithë serverat virtual duhet të ekzekutojnë Linux ,sepse OpenVZ dhe Virtuozzo kërkojnë përdorimin e një kerneli të veçantë Linux i cili mundëson shtresën e virtualizimit të kërkuar për të bërë dallimet midis makinave virtuale, për mbështetjen dhe menaxhimin e tyre. Sidoqoftë kjo nuk do të thotë se jemi të kufizuar të përdorim të njëjtin distribucion Linux në secilin nga serverat virtual. Produktet e virtualizimit në nivel Sistemi Operativ shfrytezojnë ndarjen midis kernelit dhe një sistemi skedarësh root, i cili përmban pjesët e hapësirës së përdoruesit dhe kohës së ekzekutimit të cilat njihen si Linux nga shumica e njerëzve. Kjo na mundëson të ekzekutojmë distribucione të ndyshme të Linux në secilin nga serverat virtual. Në varësi të kërkesave për aplikacione dhe librari, servera virtualë të ndryshëm mund të kenë sisteme skedarësh plotësisht të pavarur ose mund të shperndajne aplikacione, librari etj në mënyrë që të reduktojnë

kërkesat tërësore të sistemit. Servera virtual të ndryshem mund të kenë kuota të disqeve të ndara për aq kohë sa pjesë të ndara të disqeve ose volume përdoren për të mbajtur bashkësi skedarësh specifik të serverave.

Zgjidhjet e virtualizimit në nivel sistemi operativ si OpenVZ dhe Virtuozzo mund të reduktojnë totalisht kostot e liçensimit të software-it sepse aplikacionet dhe sistemi root i skedarëve mund të shpërndahen midis serverave të ndryshëm virtual. Termat e liçensave të software-it të lidhura me serverat virtual ndryshojnë për çdo paketë software dhe distribucion të sistemit operativ.

Për të thjeshtëzuar krijimin e serverave virtual unik me karakteristika të ndryshme të kohës së ekzekutimit dhe bashkësi të aplikacioneve, si OpenVZ dhe Virtuozzo sigurojnë një numër modelesh të distribucioneve Linux, të cilat janë bashkësi paketash nga distribucione të ndryshme Linux që mund ti përdorim për të plotësuar një sistem skedarësh apo strukturë direktorie që është specifike ndaj një serveri virtual.

OpenVZ ofron përfitime menaxhimi bazë të ndarjes së burimeve dhe menaxhimit midis serverave të ndryshëm virtual. Zgjidhja komerçiale Virtuozzo nga SWSoft siguron një numër të rëndësishëm të kapaciteteve shtesë të menaxhimit që thjeshtëzojnë sigurimin e serverave, rekuperimin, migrimin, e kështu me radhë.

4.4.6 Parallels Workstation

Parallels është një paketë virtualizimi guest-OS që ekzekutohet në sistemet Linux 32-bit, Microsoft Windows dhe Apple Mac OS X. Prezantuar në 2005, Parallels u bë popullor shumë shpejt si një alternativë komerçiale me kosto të ulët për produktet konkuruese si VMware Workstation. Kur Apple ndryshoi nga PowerPC në proçesoret Intel, Parallels ishin produktet e para të virtualizimit të disponueshme për MAC. Parallels aktualisht zotërohet nga SWSoft, Inc., shitësit e produktit të virtualizimit në nivel sistemi operativ për ndërmarrjet, Virtuozzo dhe versionit të tij open source Open VZ.

Pavarsisht shqetësimeve për koston, Parallels nga pikëpamja e ndërfaqes administrative është një mjedis virtualizimi guest-OS i shpejtë dhe i fuqishëm, njësoj si produktet VMware Workstation, VirtualBox dhe Microsoft Virtual PC.

Parallels përdor një numër të moduleve kernel për të ndërvepruar direkt me hardware-in e sistemit dhe për të caktuar dhe planifikuar efektivisht aksesin e burimeve . Parallels përdor kombinimin e një hypervisori (ngarkuar si një modul kerneli), module të tjera kernel për rrjetin dhe menaxhimin e përgjithshem të hardware-it dhe mbështetje për instruksionet e virtualizimit të hardware-it në proçesorët e selektuar për të siguruar një performancë të lartë të mjedisit të virtualizimit që mund të ekzekutojnë lehtësisht sisteme operative virtuale. Një nga cilësitë më të mira për përdoruesit e desktopit është suporti i i clipboard-it, që e bën më të lehtë per "cut" dhe "paste" mes aplikacioneve që ekezkutohen në sistemin operativ të desktop-it dhe në makinën virtuale. Ngjashmërisht mënyra e koherencës së Parallels na mundëson të ekzekutojmë aplikacione nga një makine virtuale krah për krah me aplikacione që po ekzekutohen në sistemin operativ të desktop-it.

4.5 Përmbledhje

Në këtë kapitull janë dhënë lloje të ndryshme Hypervisorësh, duke filluar nga ai me kryesori, i quajtur XEN dhe më pas kalohet në OpenVZ dhe KVM. Këto hypervisorë janë parë në nivel të virtualizimit të plotë apo para-virtualizim. Më pas këto Hypervisor janë krahasuar me Hypervisor të tjerë, kryesisht komercial si: Hyper-V dhe VMWare.

Në të 3 Hypervisorët të cilët janë mëtë përdorurit në punimin tim, është trajtuar koncepti i Para-virtualizimit, raportet e Sistemit Operativ mbi Host me ato Sisteme të quajtura GuestOS të cilët vendosen në hapsirën e Përdoruesit, në memorjen kryesore. Duke krahasuar Virtualizimin e Plotë me teknikën me Para-Virtualizim, do të shikojmë se nevojiten disa modifikime në kernel, kryesisht në nivelin e virtualizimit te burimeve.

Ndërkohe që në rastin e Virtualizimit të plotë duhet të plotesohen disa kushte Hardware, si p.sh. proçesor plotësisht i virtualizuar, emulim etj.

Një tjetër hapsirë në këtë kapitull rezervohet edhe për virtualizimin në nivel të Sistemit Operativ me perfaqësues Hypervisorin OpenVZ. Ky Hypervisor i trajton Sistemet Operative sikur te ishin "*Conteinera*" që do të thotë se ato mund te trajtohen si aplikacione që ndajnë një grup të njejtë instancash të sistemit. Këto lloj "*conteinerash*" janë shumë të izoluar me njeri tjetrin.

KAPITULLI I PESTË

MIGRIMI LIVE

5.1 Migrimi

Migrimi Proçes: Teknika post-copy është studiuar në mënyra të ndryshme në kontekstin e literaturës së migrimit proces: zbatuar së pari si " Freeze Free " duke përdorur një file server, pastaj vlerësohen me anë të simulimeve, dhe më vonë me anë të implementimit aktual Linux . Ka edhe një zbatim të fundit të post-copy të migrimit proces nën open-Mosix. Në ndryshim nga kjo, kontributet tona janë për të zhvilluar një teknikë të zbatueshme post-copy për migrim live të makinave virtuale. Megjithatë, këto sisteme nuk kanë fituar pranim të gjerë kryesisht për shkak të lëvizshmërisë dhe që kufizohet nga varësia e tepërt. Në kontrast, migrimi VM vepron në të gjithë sistemet operative dhe është natyrisht pa këto probleme.

Migrimi Live VM: Pre-copy është metoda mbizotëruese për migrimin live VM. Këto përfshijnë metodat bazuar në hipervizor nga VMware , Xen dhe KVM, metodat në nivel OS që nuk përdorin hpervisor nga OpenVZ, si dhe zonë të gjërë të migrimit. Të gjitha sistemet e mësipërme aktualisht përdorin migrimin e bazuar në pre-copy. Puna më e ngjashme me teknikën tonë është SnowFlock. Kjo punë ngre grupe aty për aty për të mbështetur paralelisht detyra të vështira llogaritjeje përmes VMs duke klonuar VM burim gjatë rrugës (on the fly). Kjo përmirësohet duke shtyrë në mënyrë aktive memorjen e klonuar përmes multicast nga VM burimi. Ata nuk e përcaktojnë në mënyrë të veçantë VM, as nuk e krahasojnë (ose të optimizojnë) metodën origjinale pre-copy.

Migrimi jo-live VM. Ka disa metoda jo-live për migrimin VM. **Schmidt** propozon përdorimin e kapsulave, të cilat janë grupe të proceseve të lidhura së bashku me IPC e tyre / gjëndjen e rrjetit, si njësi të migrimit. Në mënyrë të ngjashme, Zap përdor grupet e proceseve (pods) së bashku me gjëndjen e kernelit të tyre si njësitë migrimit.

Projekti Denali i drejtohet migrimit të VMs kontrolluara (checkpointed). Pezullimi/rinisja (suspend/resume) e internetit fokusohet në ruajtjen/rikthimin (saving/restoring) e gjendjes së kompjuterit në hardware anonim. Në të gjitha sistemet e mësipërme, ekzekutimin VM pezullohet dhe aplikacionet nuk përparojnë.

Migrimi hibrid live: Modeli hibrid është përshkruar fillimisht për migrimin proces. Funksionon duke bërë vetëm një raund pre-copy në fazën e përgatitjes së migrimit. Gjatë kësaj kohe, VM vazhdon ekzekutimin në burim, ndërsa të gjitha faqet e saj të memorjes janë kopjuar në hostin destinacion. Pas vetëm një përsëritjeje, VM pezullohet dhe gjendja e procesorit dhe faqet dirty non pageable janë kopjuar në destinacion. Si rrjedhojë, VM kthehet në destinacion dhe post-copy i përshkruar më sipër fillon të shtyjë faqet dirty të mbetura nga burimi. Ashtu si me pre-copy, kjo skemë mund funksionojë mirë për ngarkesën e të lexuarit intensiv (read-intensive workloads). Megjithatë, ajo gjithashtu ofron kohën totale përfundimtare të migrimit për ngarkesën e të shkruarës-intensive (write-intensive workloads), si me post-copy. Kjo metodë hibride është duke u zbatuar dhe nuk e mbulon fushëveprimi i këtij materiali.

PrePaging: Prepaging është një teknikë për të fshehur vonesën e gabimeve të faqeve (dhe në përgjithësi akseset I/O në path-in e ekzekutimit kritik) duke parashikuar punën në vazhdim dhe duke ngarkuar e faqet e kërkuara para se ato të aksesohen. Prepaging është i njohur edhe si prefetching i adaptuar ose paging adaptuar së largu. Ajo është studiuar gjerësisht në kontekstin e sistemeve të bazuara disk storage, sepse aksesi në diskun I/O në path-in kritik të ekzekutimit të aplikacioneve mund të jetë shumë e shtrenjtë. Algoritmet tradicionale të prepaging përdorin metoda reaktive dhe të bazuara në histori për të parashikuar dhe bërë prefetch punën e aplikacionit. Sistemi ynë përdor prepaging, jo në kontekstin e prefetching të diskut, por për zgjatjen e kufizuar në kohë të migrimit live në VM për të shmangur vonesat e gabimeve të faqeve të rrjetit nga objektivi për të burimi.

Self- ballooning dinamik (Dynamic Self-Ballooning) (DSB): Metoda e Balonës (ballooning) i referohet memorjes së kërkuar artificialisht brenda një kerneli guest dhe

lëshimin përsëri të kësaj memorjeje në hipervisor. Ballooning është përdorur gjerësisht me qëllim për të ripërmasuar memorjen e VM nga VMWare dhe Xen, dhe lidhet me self-paging në Nemesis. Megjithatë, nuk është e qartë se si ndërveprojnë mekanizmat aktualë ballooning, me teknika të migrimit VM live. Për shembull, ndërsa Xen është i aftë për ballooning vetëm një herë gjatë migrimit dhe dhe kohës së rindezjes së sistemit, nuk ka përdorim të caktuar të balloning dinamike për të reduktuar gjurmë të memorjes para migrimit live. Përveç kësaj, self-balloning ka qenë i angazhuar kohët e fundit në pemë burimin të Xen për të mundësuar një kernel guest të kthehet në mënyrë dinamike në memorje të lirë për te hipervisor pa ndonjë ndërhyrje specifike të njeriut.

5.2 Teknika e Migrimit Live.

Siç kemi përmendur më lart kjo teknikë është një mjet i fortë për transferimin e aplikacioneve, e faqeve të memorjes, gjëndja e CPU, gjëndja e rrjetit etj nga njëra makinë për të të tjerat. Migrimet live mund të realizohet në dy mënyra: me metodën Pre-copy dhe Post-copy. Në metodën e përsëritjes post-copy ka një performancë më të mirë se sa në pre-copy. Kjo ndodh sepse fillimisht transferohet vetëm gjëndja e CPU midis makinave, pastaj faqet e memorjes së lirë dhe faqet Dirty. Një kusht i domosdoshëm për migrimin live midis makinave virtuale është aksesi i tyre në diskun e përbashkët (shared storages). Imazhet e sistemit janë ndërtuar në disk. Nëse një makinë virtuale dështon, automatikisht një kopje të kësaj makine migrohet në makina të tjera virtuale. Ky migrim menaxhohet nga shtresa e hipervisor-it.

Sistemi “Cluster” është një sistem fizik kompjuterash të cilët janë të lidhur me njëri-tjetrin në LAN (Local Area Network). Në figurën 5.1 ne paraqesim një sistem kompjuterik klaster. Janë tre host-e fizikë me disa makina virtuale brenda (VM1, ... VM5). Bazuar në këtë figurë, nëse ndonjë makinë fizike prishet menjëherë, të gjitha informacionet e kësaj makine transferohen nëpër të gjithë rrjetin për te host-et e të tjerë fizikë . Siç thamë më lart, kjo mënyrë është e vlefshme për kontrolluesin e dështimeve, por nëse dështimi është i papritur (arsye aksidentale) nuk transferohet asgjë .

Në figurën 5.1 çdo makinë përdor një pjesë nga Data – Center, SAN (Storage Area Network), e cila quhet qendra virtuale e të dhënave dhe mund të përdoret nga çdo makinë virtuale. Disa aplikacione në qendrën e të dhënat mund të lëvizin nga një pjesë e SAN virtuale në një tjetër.

Një metodë tjetër në migrimin live është menaxhimi i performancës së CPU. Nëse një makinë virtuale është duke kryer shumë detyra, hipervisori lëviz disa nga këto detyra nga makina e caktuar në një tjetër makinë me një ngarkesë më të ulët.

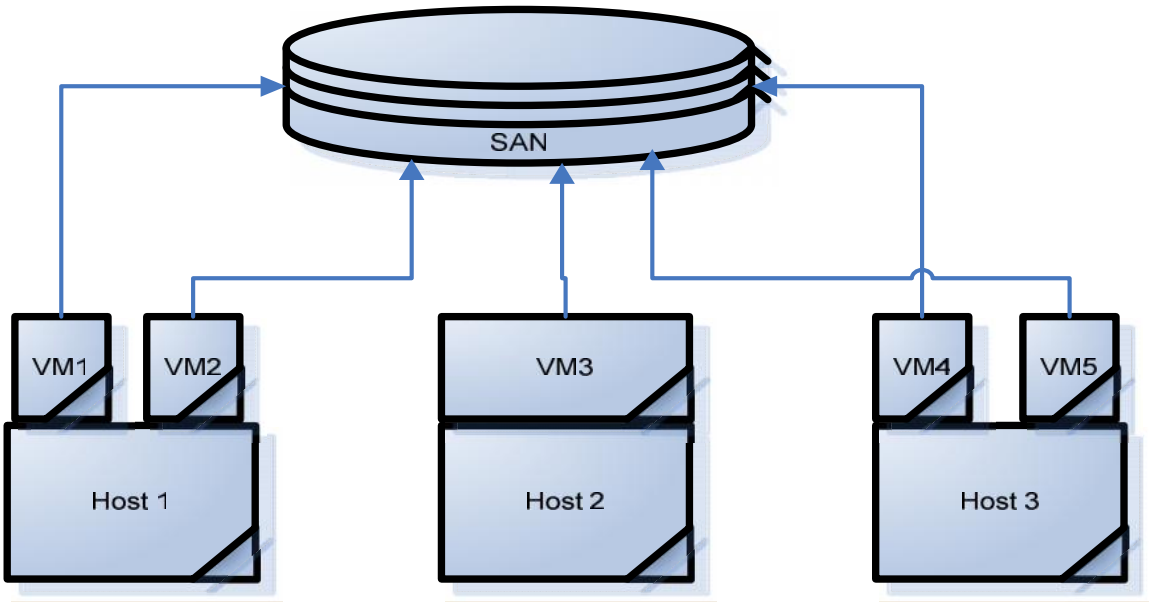


Figura 5.1- Pesë makina virtuale dhe tre hoste fizikë në sistemin klaster të lidhur me SAN

Një tjetër temë kyçe e migrimit live është zbulimin i hotspot-eve. Arkitektura sandpiper e cila zbulon hotspot-in duke mbledhur informacion nga dy mjete, kutia e zezë dhe kutia gri (*black box* dhe *gray box*).

Sandpiper zbaton një algoritëm për zbulimin e hotspot që përcakton se kur duhet të migrojnë makina virtuale. Gjithashtu përcakton se ku duhet të migrojë dhe sa vend

duhet përdorur për memorjen e makinës pas këtij migrimi.
Komponenti i zbulimit të hotspot përdor një motor për monitorimin dhe profilizimin që mbledh statistikatt e përdorimit në servera të ndryshëm virtualë dhe fizikë dhe ndërton profilet e përdorimit të burimeve.

5.2.1 Sistemet "cluster"

Brenda një makinë fizike mund të ndërtojme një sistem klaster (grup). Kjo do të thotë se një numër i madh makinash virtuale (më shumë se 2) mund të komunikojnë me njëra-tjetrën në një formë të sinkronizuar. Sistemi klaster do të thotë se një prej kompjuterëve të zgjedhur nga një grup është *kompjuteri master* dhe është përgjegjës për të menaxhuar shumë aktivitete midis *kompjuterëve slave*. Ky trajtim i jep një performancë më të mirë, sidomos kur jemi futur në aplikime kompjuterike të nivelit të lartë të tilla si shumëzim i matricave të mëdha. Një mjet i cili zvogëlon shpejtësinë e komunikimit është ndërfaqja e rrjetit dhe komunikimi i protokolleve. Nëse kompjuterat slave janë duke komunikuar me ethernet 10/100 Mb performanca e përgjithshme do të degradojë. Por në qoftë se ne përdorim shpejtësi komunikimi gigabit Ethernet koha do të zvogëlohet kështu që performanca do të rritet.

Shumica e protokolleve të komunikimit midis aplikacioneve në hoste të ndryshme janë të TCP dhe UDP. Siç e dimë, UDP është më fleksibël dhe më i shpejt se TCP, por këto analiza nuk marrin në konsideratë përdorimin e sistemit klaster në Xen. Në këtë mënyrë sistemi master-slave, i futur si Makina Virtuale dhe sistem klaster mbi Xen quhen Klaster Virtual. Nëse makina fizike prishet dmth nga avaritë AC apo Bugs të sistemit të gjithë klasterat virtual do të dështojnë.

Në këtë material duhet të kombinojmë klastera fizike dhe virtuale. Duhet të provojmë performancën e CPU në balancimin e ngarkesës gjatë gjenerimit të kërkesave nga makina burimi për të makina objektiv, përdorimi i memorjes fizike dhe virtuale në rast të dështimit të paralajmëruar dhe të gjitha aktivitetet e rrjetit. Dështimi mund të ndodhë rastësisht p.sh nga avari AC, ose për arsye natyrore si: zjarri, tërmeti, tsunami etj,

ose mund të jetë parashikuar nga ne. Ne mund të analizojmë vetëm rastin e dytë. Sistemet klaster duhet të japin karakteristika të një disponueshmërie të lartë të tilla si: Pajisje Backup (HD, DVD, Tape, teknika RAID, SAN etj), Backup i burimeve të energjisë (UPS, inverter, lidhja e një linjë të dytë AC), Data Backup Linjat Broadcast etj.

5.3 Përmbledhje

Megjithëse është kapitulli më i shkurtër, mbetet thelbi i tezës sime, duke qenë se migrimi *live* është baza e punimit të kësaj teze. Ajo që shpjegohet në këtë kapitull është se migrimi *live* në fakt nuk është plotësisht i tillë, pasi vihen re shumë bllokime të quajtura downtime gjatë kohës së migrimit të proçeseve nga një makinë e quajtur burim, në një tjetër të quajtur, destinacion. Koha e bllokimit varet nga shumë faktorë si psh: Numri i proçeseve të cilët po ekzekutohen në çastin e migrimit, numri i faqeve të modifikuara dhe të paruajtura ende në disk, mënyra e dhënies së komandës së bllokimit (të lajmëruar më parë, apo jo), lloji i Hypervisorët ku po punohet, rrjeti i komunikimit etj.

Kur diskutojmë për migrim, kuptojmë një rrjet të mirëfilltë serverash të lidhur me një sistem Baza të Dhënash, ku janë ruajtur imazhet e sistemeve në rrjet. Sisteme të tilla në rrjet formojnë atë që quhet "*High Availability*". Kuptimi i saj është krijimi i grupeve të quajtura "*Cluster*". Me anë të këtyre *Clusterave* vihet re një menaxhim shumë më i mirë në shfrytezimin e burimeve tëpërdorura apo edhe në energjinë e harxhuar.

KAPITULLI I GJASHTË

STUDIMI I PUNIMIT DHE FAZA EKSPERIMENTALE

6.1 Qëllimi i punimit

Qëllimi i punimit tim është të realizohet një migrim i makinave virtuale në një rrjet fizik lokal dhe më gjerë, i cili do të përmirësojë kohën totale të migrimit, redukton bllokimin ose siç quhet ndryshe downtime dhe ul trafikun (overhead) si pasojë e reduktimit të faqeve të modifikueshme në momentin e migrimit të tyre nga një makinë fizike në tjetrën.

Metoda e cila e arrin këtë efiçensë është bazuar në teknikën Pre-Copy. Teknika Pre-Copy ndahet në tre stade:

- Dërgimin e faqeve të lira nga njëra makinë në tjetrën
- Dërgimin e përsëritur të faqeve të modifikuara
- Dërgimin e gjëndjes së CPU-së burim drejt destinacionit.

Një tjetër metodë krahasuese është Post-Copy e cila krijon një bllokim më të vogël duke dërguar fillimisht gjendjen e CPU-së dhe më pas faqet e memorjes nga makina burim drejt asaj destinacion. Gjithsesi kjo metodë është e vështirë të aplikohet në virtualizimin në nivel Sistemi Operativ.

Siç e përmendëm dhe më lart migrim live do të thotë transferim proçesesh nga njëra makinë drejt tjetrës. Kur transferojmë këto proçese bashkë me to do të migrohen edhe burimet ku ato operojnë. Në të kundërt do të kishim vdekjen e proçeseve, cka do të shkaktonte bllokime të ndjeshme.

Duke qënë se do të migrohen proçese kjo do të thotë se destinacioni duhet të jetë gjithmonë gati për të pritur këto proçese, domethënë hosti destinacion duhet të jetë i ndezur gjatë të gjithë kohës.

Migrimin unë do ta studioje bazuar ne dy shtylla:

Në migrim të ***lajmëruar*** dhe të ***pa lajmëruar***.

Migrim i lajmëruar do të thotë që Hypervisori përgatitet dhe njofton marrësin për dërgimin e proceseve të tij. Kjo situatë është e ngjashme me atë që quhet Balancim i Ngarkesës (Load Balancing), teknikë kjo e përdorur për të shfrytëzuar në maksimum burimet.

Vështirësitë janë të mëdha kur nuk kemi migrim të lajmëruar, si psh rasti i një katastrofe në dhomën e serverave (Zjarr, Tërmet, Shkatërrim i rrjetit elektrik etj). Në raste të tilla duhet të gjenden mënyra të asaj që quhet Plani për Vazhdueshmërinë e Aktivitetit (Business Continuing Plan).

Migrimi është i lidhur me atë që quhet Cloud (Reja e të dhënave). Do të bënim punë të kotë nëse hostet janë të vendosur në të njëjtin vend. Po t'i përmbahemi standarteve ISO 27001 janë të përcaktuar qartë distancat midis Serverave që ofrojnë shërbime. Kjo distancë duhet të jetë minimumi 100 km. Lidhja mes tyre mund të jetë rrjet lokal (Local Area Network) përmes tuneleve të komunikimit (Virtual Private Network) ose nëpërmjet (Wide Area Network) përmes ruterave të cilët komunikojnë në shtresën e tretë të modelit OSI. Të dy këto tipe komunikimesh janë trajtuar në temën time. Pavarësisht kësaj në punimin tim për të mos humbur eficensën kjo distancë është reduktuar në 10 km distancë.

Nëse i kthehemi sërish migrimit të *palajmëruar* do të tregojmë se ky migrim shfaq një shkallë të lartë degradimi krahasuar me rastin e parë. Tendeca e këtij punimi është që të reduktojë dhe të ofrojë në një shkallë të pranueshme rezultatet e migrimit në rastin e palajmëruar me atë të lajmëruar më parë.

Në fakt metoda që përdorim për të dy rastet është e njëjtë, por sic trajtuam edhe më lart do të shikohet qëe tendenca do të jetë për të rritur në mënyre të ndjeshme performancën e migrimit, bllokimit dhe trafikut krahasuar me metodat e mëparshme.

Një nga metodat bazë e cila përdoret për të rritur performancën e migrimit të faqeve të modifikuara është Politika e Pastrimit (Cleaning Policy). Le ta shpjegojmë shkurtimisht këtë metodë.

Për ta kuptuar sa më qartë i referohemi migrimit të faqeve të modifikuara brenda një hosti (pra pa marë konceptin e teknikave të Virtualizimit).

Një proces është një aplikacion i cili po ekzekutohet për llogarinë tonë. Teknikat e avancuara e ndajnë memorjen në faqe ku një procesi do t'i rezervohen një madhësi e caktuar faqesh. Këto faqe mund të jenë në memorjen fizike ose Virtuale. Në rast se ato janë në të parën do të thotë se janë të afta të ekzekutohen në cast pa humbje (kryesisht në kohë). Këto faqe ndahen në dy kategori: Faqe të zëna dhe faqe të gatshme. Ekziston dhe një grup faqesh tjetër të cilat quhen faqe të lira. Këto faqe kanë si karakteristikë që janë vetëm të lexueshme ose kanë qënë më parë të modifikuara por tani nuk janë më pasi janë pastruar. Këto faqe quhen shpesh dhe faqe zero, për faktin se e kanë bitin e modifikimit M=0. Arsyeja që nuk janë më faqe të modifikuara, është se ato janë pastruar me anë të teknikave të pastrimit. Pastrim i faqes do të thotë që këto faqe kanë kryer një teknikë të quajtur shkrimi i mëvonshëm (write back) dhe si pasojë e kësaj ato kanë shkuar dhe janë ruajtur në disk. Pastrimi i tyre fillimisht bëhet në zonën SWAP të diskut dhe pasi vdes procesi, faqet largohen që andej drejt hapsirës së gjerë të Diskut të ngurtë.

Sistemet Operative ofrojnë këtë teknikë duke u bazuar në fenomenin e sinkronizimit. Kështu sistemet operative me bazë Linux kanë një bit për faqe të quajtur SYNC, i cili sa herë që merr vlerën "1" bën pastrimin e saj duke e kaluar këtë faqe në pellgun e faqeve të lira. Sistemi Operativ MS Windows e realizon këtë nëpër swape të shpërndarë në disk pa nje hapësire fikse të përcaktuar më parë nga vetë përdoruesi me anë të threadit Write Modifier Page. Si në sistemet Operative me bazë Linux ashtu edhe në Windows ekziston e njëjta llogjikë. Pastrimi i faqeve bëhet në mënyrë periodike nga memorja fizike drejt zonës Swap. Gjithsesi faqet në nivelin aplikacion janë të tipit Swapable, që do të thotë se këto faqe ikin nga memorja fizike sa herë që bëhet një kërkesë nga algoritmat e implementuara në Sistemin Operativ. Ndërkohë faqet në Kernel

janë të tipit SYN, cka do të thotë se ato mund të bëjne shkrim me anë të metodës së quajtur *write through* direkt në memorjen e ngurtë.

Faqet të cilat duhet të largohen gjatë migrimit kanë statusin:

Faqet zero, Standby Dirty dhe Busy Dirty.

Faqet zero mund të largohen pa shkaktuar bllokime, madje me një performancë të mirë në kohën totale të migrimit, gjithcka është e rrezikuar për faqet Standby dhe për më tepër faqet Bussy Dirty. Këto faqe kanë karakteristikën Swapping. Une i kam ndërruar statusin këtyre faqeve duke i kaluar nga Swapping Dirty në Synch Dirty.

Lind një pyetje, cilat mund të jenë informacionet që mund të humbin kur ne jemi duke shkruar një dokument në word pas fikjes së papritur të kompjuterit tonë?

Shkurtimisht mund të themi se të gjitha ato informacione të cilat ne nuk i kemi dhënë më parë ruaj duhet të humbin. Gjithsesi ka një shpresë e cila vjen nga zona Swap, pavarësisht se kjo zonë i referohet këtyre faqeve si pjesë të procesit i cili vdes me fikjen e kompjuterit tonë. Në zonën Swap rezervohet një tjetër zonë e vogël e quajtur Temprorary Zone e cila ka dhe konfigurimin e vet në nivel skedari [16]. Kjo zonë shërben për të bërë të gjitha zhvendosjet e proceseve nga një zonë e memorjes në një tjetër. Bashkë me to rezervohet dhe dokumenti ynë. Gjithsesi nëse mund të marrim dokumentin e plotë apo jo kjo varet nga dy faktorë sic janë:

- Cila zonë temprorary në swap mundi të jetonte edhe pas fikjes së kompjuterit tonë?
- Sa kohë kishte pa u bërë pastrimi i faqeve?

Këtu nuk mund të jepen përgjigje absolute pasi sisteme të ndryshme kanë sjellje të ndryshme. Një faktor tjetër për mosdhënien e një përgjigje të saktë është edhe baza e burimeve hardwerike. Ai që përcakton shpejtësinë e komunikimit me pajisjet është Procesori. Por një faktor tjetër është edhe ndërfaqja e periferikëve si psh ndërfaqja e diskut. Tjetër përgjigje do të ketë ndërfaqja SCSI e një Hard Disku dhe tjetër ajo IDE apo

SATA. Një rol luan dhe lloji i Parent Boardit. Kështu mund të rradhisim një sërë faktorësh dhe burimesh që janë pjesë e komunikimit.

Gjithsesi punimi im nuk merret me pjesën hardwerike, por i jep një përgjigje me ***bazë të gjerë*** fenomenit të migrimit të faqeve të modifikuara, nisur nga disa eksperimente konkrete.

Në këtë punim jo vetëm do të përmiresohen tre kushtet të cilat përmenda pak më lart lidhur me teknikat e virtualizimit, por ky algoritëm mund të futet si pjesë komunikuese e mikrokernelit të Sistemeve Operative me bazë Windows dhe Linux duke ulur humbjet në dokumentat e punonjësve, si pasojë e situatave të ndryshme të paparashikuara në kompjuterat e tyre.

Metoda të ndryshme janë përdorur për uljen e kohës së migrimit në makina të ndryshme. Ndër ato që mund të vlerësohen është një punim i kryer nga [8] të cilët kanë bërë një përpjekje të admirueshme në këtë drejtim. Ato kanë implementuar një teknikë të quajtur Post-Copy e cila redukton kohën totale të migrimit në dëm të overheadit duke eleminuar ripërsëritjen e faqeve të modifikuara. Gjithashtu ato kanë ngritur një mekanizem të quajtur Tullumbacja Dinamike Vetjake (Dynamic Self-Ballooning) duke transferuar faqet e lira nga Guest VM drejt Hypervisorit. Kjo metodë ka dy të meta:

- Funksionon vetëm për raste migrimesh të lajmëruara më pare.
- Eshtë e aplikueshme vetëm për migrime në nivel Para-Virtualizimi si Xen, OracleVM etj.

Një tjetër artikull bazohet në Parashikimin e Performancës së Makinave Virtuale, duke përdorur dy simulator të cilët nxisin rritjen e throughput-it të migrimit me qëllim uljen e kohës totale të migrimit si edhe optimizon në maksimum burimet e migruara. Ky migrim bëhet mbi Hypervisor-in Xen. Në [17] shikohet një përmirësim i ndjeshëm i kohës totale të migrimit. Por ajo që vihet re si problematike në këtë punim është se koha e migrimit ulet jo linearisht me rritjen e throughput-it. Sic e shkruajnë edhe vetë autorët e këtij artikulli, Hypervisori Xen nuk sillet mirë me rritjen e këtij parametri. Kështu që kjo

eficensë degradon me rritjen e numrit të faqeve të modifikuara. Gjithashtu një e metë tjetër e këtij punimi është se në raste të palajmëruara degradimi do të ishte akoma edhe më i keq se rasti mesatar. Kjo do të ishte një shkatërrim i vërtetë për bizneset që mund ta implementonin teknologjinë e propozuar nga autorët e artikullit.

Një artikull i jashtëzakonshëm për nga evolucioni që sjell në përmirësimin e kohës së teknikës së migrimit mbi Hypervisor-in XEN në rrjeta të ndryshme është artikulli [20]. Këta autorë bazohen në migrimin e makinave virtuale në rrjeta të ndryshme mbi Hypervisorin Xen. Duke bërë modifikimin e kernelit të këtij Hypervisori janë implementuar tre teknika të mrekullueshme:

- Parashikimi i teknikës së bllokimit Stop and Copy
- Realizimi i tabelës Hash duke futur një entry për cdo iterim të faqeve të modifikuara. Kjo teknikë quhet Zvoglimi i Përmbajtjes (Content Base Redundancy)
- Përdorimi i Faqeve Deltas [19], i cili hyn deri në përmbajtjen e Faqes së modifikuar. Duke ju referuar studimeve të librit Sistemet Operative Moderne të autorit Andrew Tanenbaum, është parashikuar që vetëm 10% e faqeve modifikohen në pjesën e kohës së punës së procesit. Kjo tregon se 90% e kësaj faqe nuk është e modifikuar. Në këtë mënyrë autorët kanë ngritur një Cache e cila ruan faqen e cila merr përmbajtjen e faqes dhe bën krahasimin me faqen e modifikuar. Kjo metodë bën që të dergohet në Disk vetëm diferenca e faqes së modifikuar.

E meta e këtij punimi është:

- Kërkohet të modifikohet në Kernel, aty ku ndodhet Hypervisori.
- Kjo metodë nuk sqarohet nëse mund të implementohet në sistemet me Virtualizim të Plotë apo ato me Virtualizim në nivel Sistemi Operativ.

Gjithsesi ajo qe vihet re në këtë punim është se ulet ndjeshëm koha e migrimit, po ashtu ulet ndjeshëm edhe bllokimi apo trafiku.

Një tjetër artikull që tregon mënyrën e uljes së trafikut të migrimit midis makinave virtuale në rrjeta të ndryshme është [22]. Ky artikull bazohet te migrimi i gjithë makinave virtuale në rrjeta të ndryshme duke shfrytëzuar dobinë e teknikës së aksesimit direkt të pajisjes I/O në memorje pa trafikun që shkakton Sistemi Operativ. Sipas këtij artikulli trafiku gjatë migrimit me këtë metodë reduktohet deri në 80% krahasuar me metodën standarte TCP/IP. Këtë metodë e kam aplikuar edhe une në artikullin tim nisur nga eficensa e përfituar e sugjeruar nga autorët.

E meta e këtij artikulli është se përsëri i gjithë testimi bëhet vetëm në një Hypervisor, XEN. Dmth nuk janë parë sjelljet e rezultateve në ambjente virtuale të ndryshme.

Duke qënë se unë shumë simulime i kam kryer mbi ambjentin Oracle 10g, një mënyrë tjetër për të minimizuar kohën e Migrimit të Dhënave është përdorimi i Makinës Virtuale Oracle Virtual Machine e cila mbështetet mbi këtë platforme. Në artikullin [24] duke përdorur tabelat e transportueshme të Platformave të Kryqëzuara (Minimizing Data Migration Time, Using Cross-Platform Transportable Tablespace), sipas autorit e redukton kohën e migrimit të dhënave duke zhvendosur databasen nga një platformë në një tjetër me 8 herë më shpejt.

Në artikullin [23], jepet një përshkrim i teknikave të virtualizimit. Ajo që vihet re të ky artikull është përdorimi i metodave të ndryshme të migrimit si psh: Metoda e Kompresimit e cila rrit kohën totale të migrimit por ul trafikun. Gjithsesi kjo metodë mund të implementohet te faqet e lira ose sic njihen faqet zero. Këtë metodë e kam aplikuar dhe une në punimin tim. Metodë tjetër interesante është metoda e parashikimit të Bashkësisë së punës së zvogluar (Minimal Working Set) duke përdorur algoritmrat LRU (The last Recently Used) dhe Splay Tree [20], [21]. Kjo metodë ul kohën e migrimit nëse numri i faqeve të modifikuara është i ulët, apo metodat CR/TR (Check Point Recovery / Trace and Replay) dhe Migrimi duke përdorur skedulimin e CPU-së. Të dyja këto metoda

ulin kohën totale të migrimit dhe bllokimin, por kjo e dyta shoqërohet me një degradim të CPU-së [25], [26].

Gjithsesi për kompresimin e memorjes, artikuj të shumtë tregojnë më qartë metodat e kompresimit të memorjes si psh [27] bazohet kryesisht te metodat Hash. Kjo metodë është implementuar edhe në artikullin [23]. Metoda që përdoret në këtë artikull quhet MECOM, dhe bazohet në kompresimin vetëm të faqeve të lira.

Në artikullin [25] vihet re një pikë shumë interesante të cilën e kam shfrytëzuar une në artikullin tim kur kam eksperimentuar me teknikat në rrjeta të ndryshme. Autorët kanë theksuar se mund të përmirësojnë eficensën e migrimit duke përdorur teknikën SRIOV (Single Root I/O Virtualization). Kjo metodë shoqërohet edhe me reduktim të overhead-it.

Një mënyrë eficente për të ulur kohën totale të migrimit në Hypervisor-at Xen dhe KVM është testuar edhe në artikullin [28]. Këtu eshte futur teknika e ndarjes së Listës së Faqeve Aktive me ato të lira. Kështu gjatë migrimit dërgohen vetëm faqet e lira dhe më pas algoritmi i ndërtuar [29], bën të mundur bllokimin dhe migrimin e faqeve të modifikuara. Sipas tyre do të vihet re një kohe migrimi më e vogël që shoqërohet me një rritje të bllokimit.

Një artikull i jashtëzakonshem për migrimet paralele është [30]. Këtu me anë të tabelave hash të ngritura në memorjen e përbashkët midis disa makinave virtuale që ndajnë të njëjtin host bëhet e mundur të përmirësohet koha totale e migrimit, trafiku në rrjet dhe bllokimi. Duke implementuar një algoritëm në kernelin e Hypervisor-it bëhet e mundur të realizohet metoda e kontrollit të memorjes midis makinave të ndryshme virtuale konkurrente. Faqet e njëjta te të gjitha makinat virtuale gjatë migrimit nuk kanë nevojë të dërgohen në destinacion, duke ulur kështu ndjeshem trafikun, kohën totale të migrimit dhe bllokimin. E meta e këtij aritikulli është se implementohet vetëm në virtualizimin e plotë me KVM – QEMU dhe nuk i jep zgjidhje rasteve të pa lajmëruara.

Një tjetër metodë është përdorur në artikullin [31]. Kjo metodë është një implementim matematikor i cili bazohet në modelin e performancës së aplikacioneve dhe burimeve gjatë migrimit. Duke analizuar këtë performancë kryesisht në Migrimin e CPU-së, Memorjes, Diskut dhe Rrjetit ato kanë ngritur një model i cili parashikon sjelljen e migrimit dhe merr masa për uljet e kohës totale të saj.

Së fundmi një artikull të cilin e kam shfrytëzuar në punimin tim është edhe [32] në të cilin përdoret një metodë e quajtur faza e parangrohjes ("*warm up adaptive phase"*) e cila rrit ndjeshëm kohën totale të migrimit. Gjithsesi kjo metodë është e vlefshme vetëm për rastin kur kemi aplikacione shumë të mëdha. Ky aplikacion është gjeneruar vetëm në Hypervisorin KVM, gjithsesi ai mund të implementohet kollaj dhe në hypervisor të tjerë si XEN, Oracle VM etj.

Nisur nga këto referenca do të shpjegoj ndërtimin e algoritmit tim "***Live Improve***" i cili rrit eficensën e Teknikës së Migrimit Live në rrjetat Lokale dhe më gjerë.

Provat janë realizuar mbi Hypervisor të ndryshëm dhe përkatësisht: XEN në nivelin Para Virtualizim dhe XEN në Full Virtualizim duke implementuar teknikat QEMU. Po ashtu është vepruar dhe me Hypervisorin KVM në Full Virtualizim dhe Para Virtualizim. E së fundmi janë implementuar teknikat e virtualizimit në nivel Sistemi Operativ duke përdorur si Hypervisor OpenVZ. Pra bazuar në këto 5 Hypervisor janë realizuar të gjitha testimet dhe janë bërë të gjitha modifikimet e nevojshme për migrimin e makinave virtuale nga një host fizik në tjetrin, me qëllim rritjen e performancës së parametrave përkatës që i karakterizojnë.

Për matjen e tyre janë përdorur skripte të ndryshme të cilët do t'i trajtojme me rradhë. Të gjithë keto skripte janë ngritur në gjuhën e programimit C ose C++. Gjithashtu janë përdorur edhe mjete të gatshme apo benchmark nga të cilat kemi marre vlerat përkatëse për testet e kryera.

Si është ngritur struktura e algorimit?

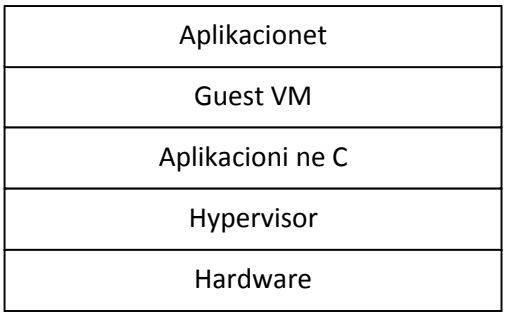


Figura 6.1- Raporti i algoritmit të ngritur me Burimet Hardware, Hypervisorin dhe Aplikacionet.

Kjo figurë tregon anën llogjike të ngritjes së aplikacionit. Ky aplikacion është i ngritur mbi Hypervisor, dmth është në hapësirën përdorues të memorjes fizike. Qëllimi i tij është të krijojë një thread në kernel i cili do të shërbejë për të testuar Tabelën e Faqeve të Memorjes. Në momentin kur shikon që P=1 kjo do të thotë se faqja është në memorjen fizike dhe i perket një procesi i cili mund të jetë në një nga dy gjëndjet: Gati ose Ekzekutim. Pas kësaj threadi teston bitin M. Nëse ky bit është 1 do të thotë që faqja është ekzekutuar dhe modifikuar. Për këtë duhet të dërgohet faqja në Disk. Por sic e përmendem më lart disku fut një penalitet të konsiderueshëm. Dy janë faktorët që fusin këtë penalitet:

- Koha e Rrotullimit
- Koha e kërkimit të sektorëve.

Algoritmi im bën të mundur minimizimin e këtyre dy parametrave. Këtë algoritem e kam quajtur ***Live Improve***. Kështu përsa i përket kohës së rrotullimit do të krijohet një zonë e vazhdueshme sektorësh në disk të cilët mund të kapen me një operacion të vetëm shkrimi dhe leximi. Kjo zonë do të jetë brenda zonës të quajtur SWAP, një zonë e përdorur nga Sistemi Operativ për menaxhimin e proceseve. Zona e përcaktuar nga algoritmi është Temprorary Zone në Swap sic tregohet dhe në [16].

Kjo zonë ofron kohën më të vogël të shkrimit. Ndërkohë për të ulur kohën e kërkimit të sektorëve në cdo Hypervisor është modifikuar kodi burim në XEN, KVM dhe

OpenVZ dhe është ndërtuar një tabelë hash, e cila funksionon në bazë të një funksioni hash. Me anë të kësaj tabele realizohen një grup celësash të cilët ruajnë lidhjen një me një të cdo sektori me një numër. Kështu duke kërkuar celësat ne mund të aksesojmë vetë sektorët, kjo gjë do të ulë ndjeshëm kohën e kërkimit të sektorit.

Ajo që do të modelohet më pas nga algoritmi im do të jetë aktivizimi i një biti të quajtur SYN për cdo faqe. Ky bit implementohet në cdo Sistem Operativ Linux. Ndërkohe që në Sistemin Operativ Windows për të bërë sinkronizimin e shkrimit përdoret një thread i quajtur Write Modified Page. Duke qënë se Hypervisor-at të cilët kam testuar janë me origjinë Linux, biti i cili do të testojmë për sinkronizim do të jetë vetëm SYN. Ky bit nuk bën gjë tjetër, përvec Politikës së Pastrimit të faqeve, duke i kthyer faqet në të ashtuquajtura faqe të lira ose faqe zero. Pavarësisht caktimit të përmirësimit të Kohes së Rrotullimit të Diskut të Ngurtë, pavarësisht dhe kohës së përmirësuar për kërkimin e sektorit duke implementuar tabelën hash, degradimi në shkrimin e cdo faqe nga përdoruesi në Disk është shumë i madh. Për këtë janë impelementuar dhe disa teknika të tjera të cilat kanë ndikuar në uljen e kësaj kohe. Tabelat e përshkruara më poshte kanë treguar dhe përmirësimin në ruajtjen e dokumentave në momentin e një situate kritike.

Të gjitha metodat e studiuara më pas janë testuar vetëm për rastin e migrimit live të makinave virtuale nga një host fizik në një tjetër mbi 5 Hypervisor të ndryshëm.

Metoda të ndryshme janë implementuar me qëllim uljen e kohës totale të migrimit, uljen e kohës së bllokimit dhe të trafikut, por pa krijuar shqetësim në punën e përdoruesit. Një metodë tjetër e cila është implementuar në punimin tim është teknika Delta Compression Memory. Kjo teknikë bazohet në implementimin e një tjetër tabele hash në kernel dhe që teston vetëm faqet e modifikuara. Janë përdorur një sërë testesh dhe aplikacionesh të cilët simulojnë modifikimin dhe shkrimin e faqeve. Në faqet e modifikuara vetëm një pjesë e tyre modifikohet, pjesa më e madhe e saj mbetet e pashkruar. Algoritmi i ngritur nga unë bën të mundur të caktojë një buffer cache ku do të implementohet algoritmi Last Recently Used. Këtu do të ruhen faqet prezente të

modifikuara së fundmi. Qëllimi është të dërgohet në zonën temprorary të Swap-it vetëm diferenca e ndryshuar së fundmi, jo e gjithë faqja. Për këtë në tabelën hash do të ruhet cdo fjalë me një celës integer. Në rastin kur modifikohet një fjalë ndryshon një vlerë në tabelën hash e cila dërgohet në Disk. Zona Cache që implementon tabelën Hash me algoritmin LRU, lokalizohet në Cache Level 2 dhe është e ngjashme me Tabelën TLB (Translation Lookaside Buffer) që ruan entry më të përdorur të Tabelës së faqeve brenda MMU-se. Sipas tabelave të nxjerra nga eksperimentet shikohet një përmirësim i dukshëm i kohës totale të migrimit, kohës së bllokimit dhe trafikut. Gjithsesi shikohet një ngarkesë e madhe e CPU-së dhe sërish një degradim i aktivitetit të përdoruesit. Koha e përgjigjes është në rendin e disa sekondave. Kjo do të thotë që do të ishte e nevojshme fillimisht një teknologji sa më e mirë e CPU-se, Bordit, Memorjes Fizike dhe Ndërfaqes së Diskut.

Meqënëse në shembullin e mësiperm trafiku i gjeneruar nga celësat hash si pasojë e modifikimeve të shumta është i konsiderueshëm, mund të përdorim teknikën MECOM, dmth të kompresimit të memorjes së modifikuar. Rezultatet e nxjerra tregojnë një ulje të trafikut midis Hosteve dhe Storage të Përbashkët, por shikohet një degradim shumë i madh i CPU-së. Kështu që kjo metodë shikohet e pa përshtatshme në punimin tonë.

Faqet e modifikuara në Bufferin Cache L2 i cili do të ruajë të gjitha ndryshimet sic e thamë më sipër mund të implementohet duke përdorur algoritmin LRU për faqet e modifikuara së fundmi. Faqet e modifikuara së fundmi korrespondojnë me Bashkësinë e punës. Një bashkësi sa më eficente e bën penalitetin më të vogël, kjo do të thotë më pak kohë për migrimin e të dhënave, bllokimin e aktivitetit dhe trafikun mes makinave fizike. Duke implementuar në algoritmin tim dy metoda: Metoden Splay Tree dhe LRU shikohet që do të arrihet një ulje e ndjeshme e parametrave të mësipërm.

Duke u bazuar në referencat [20] dhe [21] në këtë algoritëm është implementuar dhe një metodë e re për migrimin e të dhënave. Deri tani ne kemi aplikuar metodën Pre-Copy. Por nëse e modifikojmë disi këtë metodë shohim një përmirësim të lehtë të parametrave sidomos në pjesën e trafikut. Metoda e re që implementohet në Hypervisor-at XEN dhe KVM ka të bëjë me ndarjen e faqeve në Lista Aktive dhe Lista të Lira. Duke

implementuar listat zinxhire do të bëhet fillimisht transferimi i Listave të Lira dhe veteëm me një goditje të vetme do të migrohen Listat Aktive.

Kjo metodë vërtet sjell disa përmirësime por nuk mund të implementohet në teknikat e virtualizimit në nivel Sistemi Operativ.

Për të rritur performancën testet janë kryer dhe mbi një sistem data base Oracle 10g. Në këtë sistem është ngritur makina virtuale Virtual Oracle që nga arkitektura është shumë e ngjashme me XEN. Në këto testime është përdorur teknika Cross Platform Transportable Tablespace - Tabela e Transportueshme e Platformës së kryqëzuar. Eficensa që përfitohet në kohën totale të migrimit është shumë e mirë. E njëjta gjë nuk mund të thuhet për trafikun. Sic do të shikohet nga tabelat reflektohet një kohë migrimi shumë e mirë, po ashtu dhe një bllokim i ulët por trafiku është i rënduar.

Testimet deri në këto momente janë bërë vetëm me anë të migrimit të një Sistemi Operativ nga një host drejt një tjetri. Por sic dihet Virtualizimi lejon përdorimin e njëhershëm të disa makinave virtuale mbi të njëjtën platformë. Eficensa e migrimit duke përdorur migrim sekuencial midis makinave të ndryshme virtuale do të rezultonte një katastrofë e vërtetë. Për këtë na ndihmon migrimi paralel i bazuar në teknikën Live Gang. Kjo metodë ngre një tabelë hash e cila ruan të gjithë informacionet e njëjta në memorjen e përbashkët dhe dërgon vetëm ndryshimet midis makinave të ndryshme virtuale. Nëse një faqe i përket disa makinave ajo do të ruhet në tabelën hash me dy celësa që tregojnë vlerën e faqes dhe numrin e makinës virtuale përkatëse.

Të gjitha teknikat janë implementuar deri më tani në rrjetat lokal. Për të implementuar në rrjeta të ndryshme ose WAN (Wide Area Network) është përdorur një arkitekturë me Fibra optike dhe me Switch-e të implementuara me ndërfaqe fibër [22], [33], [34]. Kjo gjë do të bënte që të mos ndihej vonesa ose ajo të minimizohej në rende të papërfillshme. Një metodë që është përdorur për të implementuar teknikë sa më eficente gjatë migrimit në rrjeta të ndryshme është RDMA – Remote Direct Memory Access. Kjo do të bënte të mundur të minimizonte në maksimum vonesën e shkaktuar nga Stacku TCP/IP duke i dhënë mundësi pajisjeve I/O të aksesojnë direkt memorjen.

Së fundmi një tjetër test është kryer duke krijuar një rrjet të madh hibrid cluster me High Availability. Edhe këtu janë testuar eficensa e migrimit krahasuar me rastin e migrimit 1-1. Rezultatet janë ruajtuar në tabelat përkatëse. Një tjetër krahasim është High Availability duke mos shtuar shtresën e aplikacionit. Diferencat janë shfaqur në formë tabelore.

Aplikacioni që unë kam ngritur mund të implementohet në formë të lajmëruar dhe jo të lajmëruar. Gjithsesi falë algoritmit hapësira e krijuar midis dy teknikave është shumë e vogël, krahasuar me rastin kur ky algoritëm nuk implementohet. Tabelat përkatëse tregojnë parametrat e virtualizimit për të dy këto raste.

Implementimi i algoritmit në nivelin përdorues e bën ate abstrakt ndaj llojit të Hypervisorit. Ajo që i duhet këtij aplikacioni është të jap një ndërfaqe komunikimi me Hypervisorin, ku i kërkohet një shërbim nga kerneli për cdo thirrje nga ky algoritëm. Gjithsesi qëllimi kryesor i punimit është që përdoruesi mos të humbas informacion dhe të jetë transparent pavarësisht problemeve dhe bllokimeve që ndodhin midis Hypervisor-ave. Për këtë është bërë një përllogaritje matematikore duke ngritur tabelën hash dhe është parë që propabiliteti për të humbur një informacion është në nivele < 0.5%.

6.2 Ambienti i Eksperimentit

Le të japim skemën llogjike të punimit tim. Duke i trajtuar hostet si Thin Client, do të ruajmë të gjithë aktivitetin e proceseve në Disk duke bërë sinkronizimin.

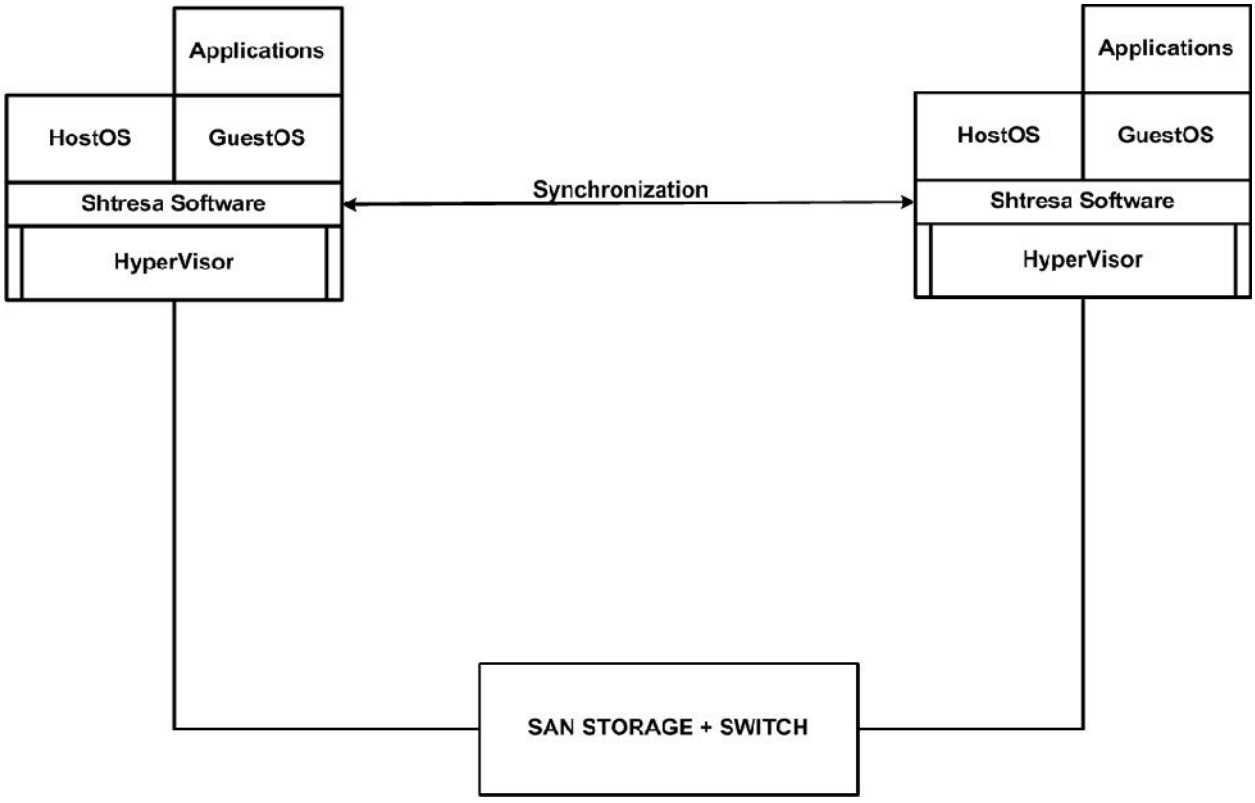


Figura 6.2: Skema Llogjike e komunikimit sinkron midis hostit burim dhe destinacion.

Atëhere cdo aplikacion në Sistemin Operativ Guest do të tentojë të bëjë sinkronizim drejt SAN Storage. Për të ulur degradimin sic e përmendëm dhe në seksionin e mësipërm do të implementojmë disa metoda.

Fillimisht do të testojmë rastin kur kemi një Sistem Operativ Linux CentOS 5.6 në një PC me keto parametra:

- Procesor Core i7-950
- Nr i Bërthamave 4
- Nr i Threadeve 8
- Shpejtësia e Orës 3.06 GHz

- Frekuenca Maksimale e Turbos 3.33 GHz
- Smart Cache 8 MB
- Bashkësia e Instruksioneve 64 bit
- Memorja Fizike 24 GB
- Teknika e Memorjes RAM DDR3
- Suporton Teknologjinë Turbo Boost
- Suporton Teknologjinë Hyper-Threading
- Suporton Teknologjinë e Virtualizimit
- HDD SATA 3 me 64 MB Disk Cache

Atëhere skematikisht do të kemi këtë skicim:

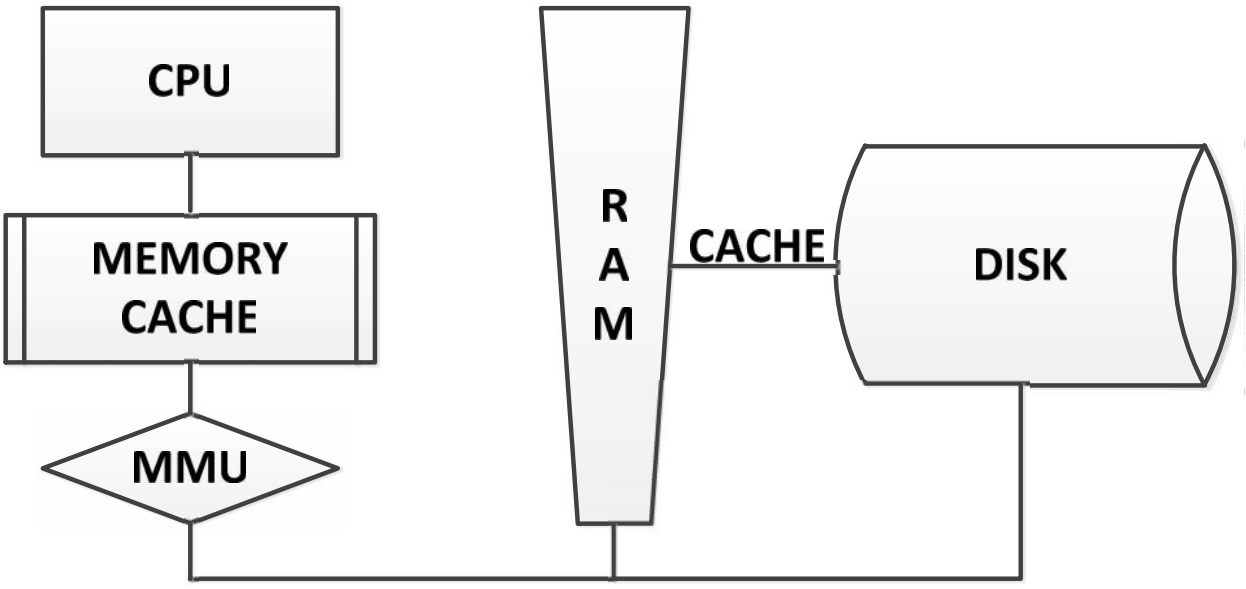


Figura 6.3- Përkthimi i faqeve fizike dhe faqeve virtuale nga MMU. RAM shërben si cache e Diskut.

Kjo është lidhja e brendshme e pajisjeve I/O brenda PC. Këtu sic shikohet nuk kemi ngritur një ambient Virtual por jemi direkt në një komunikim fizik.

Për të shkruar në disk kam implementuar një skript të thjeshtë në gjuhën C++ që shkruan numra të plotë integer nga 0-999 999. Skriptin e kam quajtur *GenNum* dhe sic shikohet shkruan në memorjen fizike deri ne 1 milion numra deri në vdekjen e tij. Ky skript do të ekzekutohet pasi kompilohet në një gjuhë makinë dhe më pas ekzekutohet si kod objekt duke marrë prapashtesën .exe. Në këtë cast sapo është gjeneruar një proces i ri i cili ka një ID. ID e procesit mund të lexohet nga komanda ps dhe më pas të jepet urdhri për vdekjen e tij. Për të bërë vrasjen e procesit do të lindë një proces tjetër në kernel i cili ka privilegjin të zbatojë urdhrin. Pra do të jemi në këtë rrjedhë:

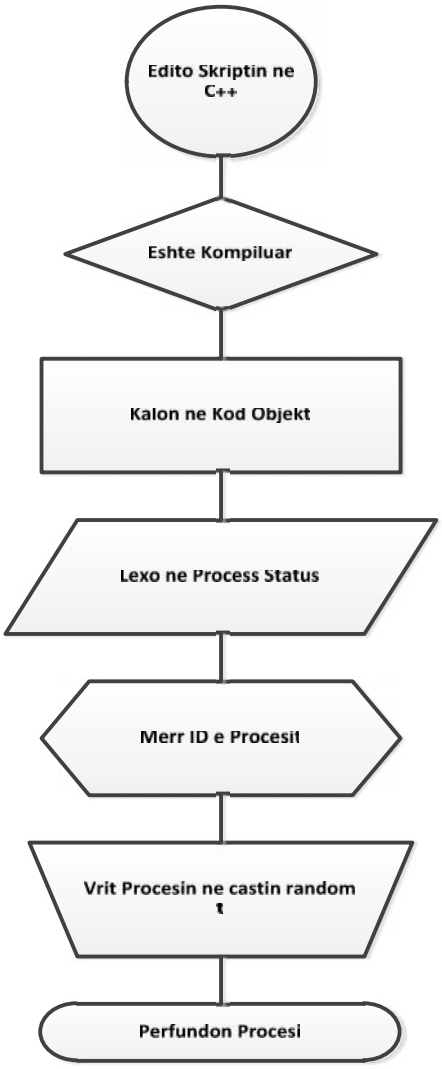


Figura 6.4- Përpunimi i skriptit dhe momenti kur skripti ekzekuton një komandë kill

Pas realizimit të këtij skripti do të masim sa është numri i vlerave të ruajtura në disk. Koha e zgjatjes së procesit është 10 sekonda. Gjithsesi falë një goditje të papritur rastësore, procesi do të vdesë më shpejt. Dihet që Linux përdor bitin SYN për të bërë të mundur sinkronizimin e faqes së modifikuar në zonën Swap në Disk. Do të shohim sa numra janë ruajtur në skedarin e përkohshem në këtë zonë. Nëse kontrollojmë pas përfundimit të procesit sa është numri i ruajtur në këtë file do të shikojmë vlera të

ndryshme. Një faktor që marrim vlera të ndryshme është vetë futja e goditjes vdekjeprurëse rastësore. Sa më vonë të ndodh ajo aq më shumë vlera janë ruajtur në disk, falë teknikës së sinkronizimit në disk. Nëse vlerat e fiksura të kohës i ritestojmë sërish për rastin kur kemi instaluar programin tonë do të marrim vlera të tjera. Rezultatet jepen në tabelën më poshtë:

Tabela 6.1: Diferenca midis numrave të shkruar në swap bazuar në aplikacion dhe ato pa aplikacion në momentin e vdekjes së proçesit të shkrimit.

	Pa Aplikacion		**Me Aplikacion**	
Koha totale e procesit	**Koha e Vdekjes**	**Numrat e shkruar në Swap**	**Koha e vdekjes**	**Numrat e shkruar në Swap**
10 sekonda	**5.7 sekonda**	**< 100 numra**	**5.7 sekonda**	**≈ 150 000**
10 sekonda	**8.9 sekonda**	**< 300 numra**	**8.9 sekonda**	**≈ 400 000**
10 sekonda	**4.5 sekonda**	**< 70 numra**	**4.5 sekonda**	**≈ 120 000**
10 sekonda	**6.6 sekonda**	**< 200 numra**	**6.6 sekonda**	**≈ 250 000**

Pra ajo qe vihet re nga tabela e mësipërme është se Numrat e shkruar në Swap në filen e përkohshme të saj në rastin kur instalojmë aplikacionin është shumë herë më i lartë se në rastin kur nuk ka aplikacion. Kjo do të thotë se më pak informacion mund të na humbë në rastin kur kemi instaluar këtë aplikacion si pasojë e një situate kritike të paparashikuar, si për shembull ndërprerje energjie.

Tabela është përcaktuar me rastin kur kemi implementuar aplikacionin e përshkruar në figurën më poshtë. Algoritmi cakton një zonë temprorary brenda Swap. Kjo madhësi merret sa 0.20-0.25 e madhësisë së RAM-it, kur koha e vdekjes ekzekutohet

vetëm në castin ***4.5 sekonda***. Nëse kjo zonë do të ishte më e madhe do të shikonim një reduktim të numrave të shkruar si në tabelën më poshtë.

Tabela 6.2-Caktimi i madhësisë së zonës Swap në varësi të performancës së shkrimit.

Madhësia e Files Temprorary	**Numrat e shkruar në SWAP**
0.1 RAM	≈ 100 000
0.2 RAM	≈ 120 000
0.3 RAM	≈ 120 000
0.4 RAM	≈ 110 000
0.5 RAM	≈ 80 000
0.8 RAM	≈ 60 000
1 RAM	≈ 20 000

Për të përcaktuar vlerën e files temprorary komandat jepen në skriptin tonë në hapësirën përdorues, por vlerat do të përcaktohen nga proceset në kernel. Për këtë në kernel duhet të krijohet një proces që pointon në limitet e përcaktuar nga skripti. Për ta interpretuar këtë komandë duhet të modifikojmë në Kernelin e Sistemit Operativ. Për këtë rast kam shfrytëzuar Sistemin Operativ CentOS5.6 me kernel në versionin 2.6.18-238.

Po të ndërtojmë skematikisht ngritjen e aplikacionit deri në momentin kur nuk kemi futur konceptin e migrimit të makinave virtuale do ta shfaqim me algoritmin e figurës 6.4.

Gjithashtu për të rritur performancën e përcaktuar në tabelën 6.2, bëjmë një modifikim të skriptit tonë duke ngritur tabelat hash në këtë hapësirë. Kërkimi tashmë në bazë celësash do të bëjë që kerneli t'i përgjigjet kërkesave duke bërë kërkimin me bazë celësash. Për këtë kërkohet një modifikim në kernel jo më në bazë gërmash të alfabetit por celësash. Mënyra se si kerneli do ta bëjë këtë përcaktohet nga metoda e implementuar në këtë skript në hapësirën user. Kështu do të merrnim këtë pamje të tabelës 6.3

Tabela 6.3-Implementimi i tabelës Hash për shkrimin e numrave në zonën Swap

	Pa Aplikacion		**Me Aplikacion**	
Koha totale e proçesit	**Koha e Vdekjes**	**Numrat e shkruar në Swap**	**Koha e vdekjes**	**Numrat e shkruar në Swap**
10 sekonda	**5.7 sekonda**	**< 100 numra**	**5.7 sekonda**	**≈ 180 000**
10 sekonda	**8.9 sekonda**	**< 300 numra**	**8.9 sekonda**	**≈ 800 000**
10 sekonda	**4.5 sekonda**	**< 70 numra**	**4.5 sekonda**	**≈ 150 000**
10 sekonda	**6.6 sekonda**	**< 200 numra**	**6.6 sekonda**	**≈ 260 000**

Pra po të krahasojmë tabelën 6.2 dhe tabelën 6.3 do të shikojmë që numrat e shkruar në filen Temprorary në zonën SWAP janë më shumë duke implementuar tabelen Hash se në rastin kur nuk kemi implementuar këtë tabele, psh në rastin kur procesi vdes

pas **5,7 sekondash** numrat e shkruar janë **180 000** ndersa pa implementimin e tabelës hash janë vetëm **150 000.**

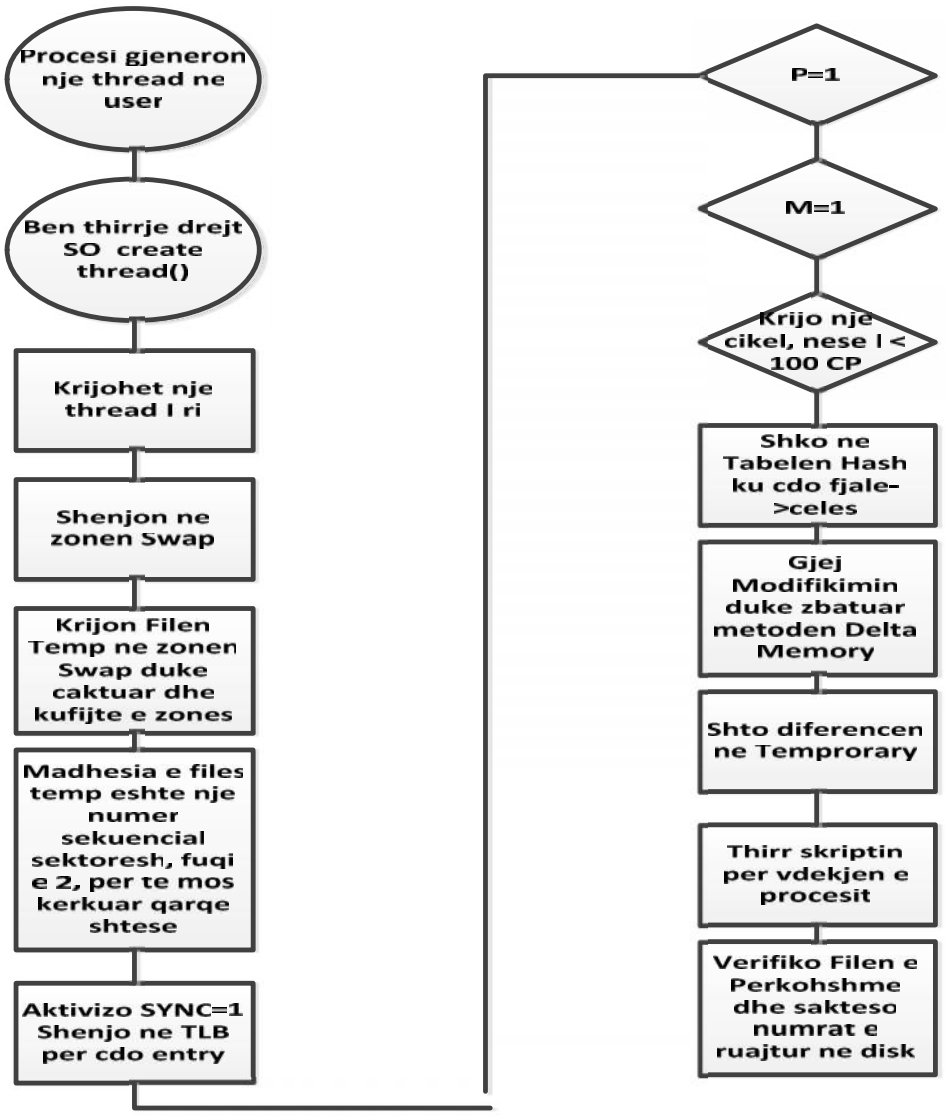


Figura 6.5: Realizimi i Programit në hapësirën Përdorues

Do të tentojmë të reduktojmë kohën e migrimit të makinave virtuale dhe aplikacioneve mbi to fillimisht duke pasur nën kontroll të vazhdueshëm faqet e modifikuara. Sic thamë qëllimi është pastrimi i tyre i menjëhershëm, i cili shoqërohet me degradim të kohës së përgjigjes. Atëhere do ta shikojmë në dy shtylla punimin tim.

Eksperimentet do të organizohen mbi Hypervisor të ndryshëm si: XEN, OpenVZ dhe KVM. Skripti i ndërtuar në nivel aplikacioni do të jap komanda të ndryshme për përmirësimin e eficensës së migrimit. Por shpesh do të na duhet të modifikojmë dhe në hapësirën kernel në mënyrë që t'i ofrohet shërbim skriptit në hapësirën përdorues. Kerneli është përgjegjës për komunikimin me Memorjen, Cachen e Diskut, Rregjistrat e ndërfaqes së Diskut, ndërtimin e mekanizmave të kontrollit të aktivitetit, marrëdhënieve me CPU-në etj.

Para se të trajtojmë migrimin e aplikacioneve në hypervisor-at e ndryshëm po japim skemen e lidhjes së tyre duke filluar brenda një rrjeti fizik lokal.

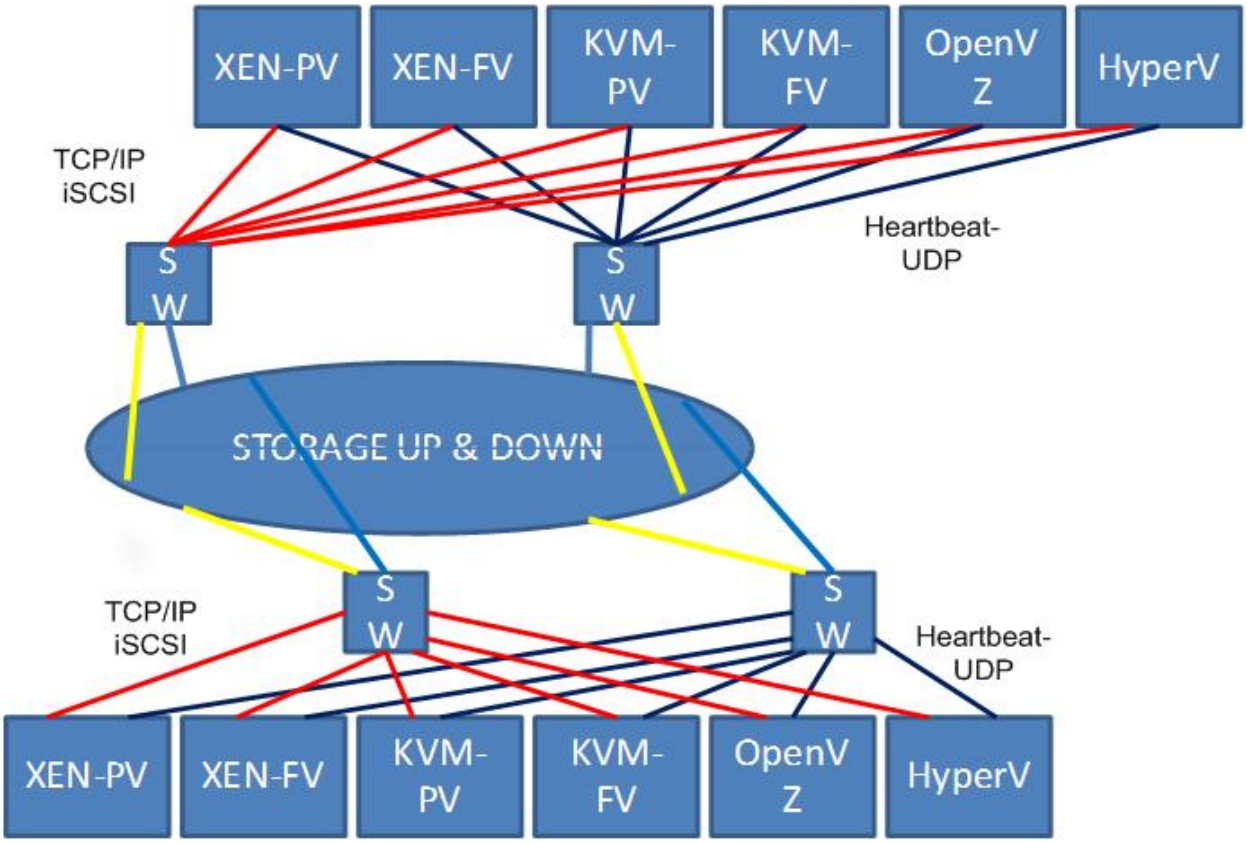


Figura 6.6- Skema e migrimit të Makinës virtuale, mbi Hypervisor të ndryshëm brenda një rrjeti LAN.

Fillimisht të gjithë Hypervisoret ndajnë një Data Storage. Ndarja e tyre bëhet duke instaluar te secili prej tyre protokollin iSCSI. Po fillojmë me parametrat dhe Topologjinë e rrjetit:

Eshtë ngritur një rrjet LAN me 12 PC të lidhur me një Switch të menaxhueshem të tipit CISCO 2960 me TOE Support, 1000 Base-T, Gigabit, 12 PC.

12 PC kanë këto parametra:

- Procesor Core i7-950
- Nr i Bërthamave 4
- Nr i Threadeve 8
- Shpejtësia e Orës 3.06 GHz
- Frekuenca Maksimale e Turbos 3.33 GHz
- Smart Cache 8 MB
- Bashkësia e Instruksioneve 64 bit
- Memorja Fizike 24 GB
- Teknika e Memorjes RAM DDR3
- Karta e rrjetit 10 Gbps over TOE.
- Suporton Teknologjinë Turbo Boost
- Suporton Teknologjinë Hyper-Threading
- Suporton Teknologjinë e Virtualizimit
- HDD SATA 3 me 64 MB Disk Cache
- Sistemi Operativ: CentOS 5.6

Protokollet e komunikimit janë TCP/IP për migrimin e faqeve të proceseve nga një makinë ne tjetrën dhe implementimi i një protokolli të quajtur Heartbeat[1], do të bëhet mbi të njëjtin Switch por bazuar në protokollin UDP sipas [53]. Bazuar mbi këtë punim do të shohim që:

$$\text{Disponueshmëria (Aftësia e një makine virtuale për t'u ngrihet pas migrimit)} = \frac{\text{Koha mesatare para migrimit}}{\text{Koha mesatare para migrimit} + \text{Koha mesatare e riparimit}}$$

(6.1)

Heartbeat e bazon funksionimin e tij tek disa proçese ("daemons") që ekzekutohen në "sistemin e ndërlidhur" dhe që komunikojnë me njëri-tjetrin gjatë gjithë kohës. Informacionet mbi veprimtarinë e këtyre proçeseve ruhen në formën e logeve dhe çdokush mund të mësojë shumë mbi sjelljen e Heartbeat duke u bazuar tek loget.

Nëse do të kishim një cluster të mirëfillte atëhere nëse mbi 50% e nyjeve janë të disponueshme atëherë vetë nyjet zgjedhin njërën prej tyre për të qënë "cluster master". Por në rastin tonë ne nuk do të shfrytëzojme këtë mundësi. Ne do të ngrejmë një skript në gjuhën e programimit C++ të quajtur *cakto_destinacion*, ku për cdo makinë fizike do të përcaktojmë destinacionin e migrimit. Ky skript do të ketë një ndërfaqe komunikimi edhe me protokollin iSCSI si protokolli përgjegjës për menaxhimin e informacionit midis makinave virtuale në hostet fizike dhe Data Bases së përbashkët Storage. Kështu Nyja Master ka nën kontroll burimet dhe nyjet fizike duke përcaktuar cila nyje fizike do të ngrihet pas migrimit të një makine virtuale. Kështu Heartbeat pasi i merr vendimet duke kontrolluar skedaret e konfigurimit urdhëron migrimin e makines virtuale, por fillimisht me anë të një modifikimi në këtë protokoll, dërgohet një sinjal te skripti jonë në C++ i cili pasi lexon informacionin që i jep ky protokoll i rikthen nyjen destinacion. Nëse kjo nyje është e rregullt për protokollin iSCSI atëhere fillon migrimi, në rast të kundërt do t'i kërkohet një ri-interpretim skriptit *cakto_destinacion*. Nëse për 3 herë skripti jep nyjen

[1] Heartbeat është përgjegjës për mbrëvajtjen e migrimit midis makinave fizike duke njoftuar ekzistencën e tyre

destinacion të gabuar atëhere, protokolli iSCSI merr vendimin vetë, duke e migruar makinen virtuale në një destinacion tjetër.

Në sistemin tonë cdo nyje është marrë pa prioritete. Imazhet e makinave virtuale ruhen në "shared storage". Mundësia e ngritjes së të njëjtës makinë virtuale njëkohësisht në më shumë se një nyje fizike mund të cojë në korruptimin e imazhit të makinës virtuale. Heartbeat ofron dy teknika për të shmangur këtë problem: "**node fencing**" dhe "**STONITH (Shot The Other Node In The Head)**".

- "**Node fencing**" ndalon aksesimin e "shared storage" nga disa nyje njëkohësisht. Lejohet të aksesohet "shared storage" vetëm nga nyja që ka nevojë të lexojë imazhin e makinës virtuale që mbart.
- "**STONITH (Shot The Other Node In The Head)**" duket brutale dhe e tillë është. Kur një nyje nuk përgjigjet pra nuk raporton statusin e saj, Heartbeat sigurohet që nyja të ketë "vdekur" duke e "gjuajtur në kokë". Një shembull i "STONITH" do të ishte rasti kur nyjes së supozuar të "vdekur" i pritet rryma elektrike apo aksesi në rrjetin LAN të "cluster-it". Ky veprim bëhet nga nyjet e tjera për të ruajtur intergritetin e informacionit.

Një problem që vihet re në sistemin Cluster është Split Brain [2**] .** Gjithsesi ky problem nuk vlen për ne, pasi këtu nuk kemi një sistem të mirëfilltë Cluster.

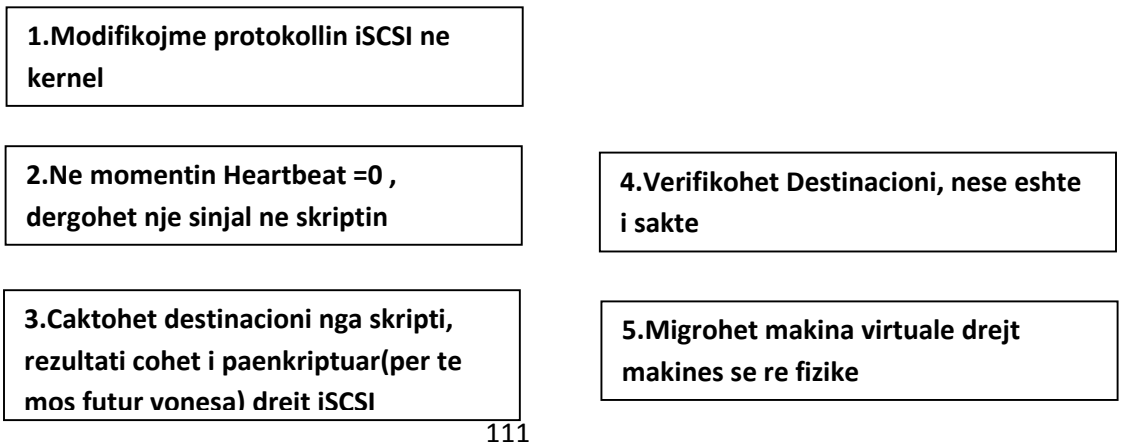


Figura 6.7- Caktimi i destinacionit të makinave fizike dhe momenti kur simulohet një bllokim.

Më poshtë po tregojmë 6 hapat e nevojshëm ku mund të ngrihet teknika Heartbeat:

1. Nyja bllok 0 (psh, gjatë fikjes së kompjuterit) - kontrollon dështimin
2. Në këtë moment heartbeat njofton VCenter duke identifikuar dhe makinën fizike të rënë.
3. Në nyjen përkatëse ekzekutohet bllokimi heartbeat /etc/init.d/x0 stop /etc/init.d/x1 stop, ku do të përfundojnë së punuari makinat virtuale x0 dhe x1 që i perkasin një makinë përkatese fizike.
4. iSCSI dërgon mesazhin e bllokimit të skritpi *cakto_destinacion*.
5. Merret përgjigja nga ky skript dhe fillon migrimi.
6. Heartbeat ekzekuton script në nyjen destinacion /etc/init.d/x0 start /etc/init.d/x1 start.

Bazuar në artikullin [29] rezulton që për rrjetat lokale nuk vihet re ndonjë diferencë midis komunikimit iSCSI dhe atyre me kanale Fiber. Mjafton që të suportohet teknika TCP/IP offload Engines ose TOE me shpejtesi mbi 10 Gbps. Kështu për këto rrjeta do të punojmë me këtë ndërfaqe komunikimi.

Nderfaqja iSCSI punon mbi TCP duke përdorur portat 860 dhe 3260. Ndryshe nga komunikimi paralel në SCSI këtu përdoret një komunikim serial Full Duplex TX dhe RX sikurse edhe komunikimi me kanale Fibër.

Komunikimin me Fiber Optike do ta aplikojmë në rrjetat WAN.

Do të fillojmë me Hypervisorin XEN në nivel Paravirtualizimi dhe Full Virtualizimi duke ngritur emulatorin QEMU. Do të simulojmë një rast të lajmeruar më parë. Më pas do të testojmë edhe rastin pa lajmërim.

Hypervisori XEN sic e kam shprehur edhe më sipër punon mbi teknikat e Paravirtualizimit, çka do të rriste shumë eficensën e tij në lidhje me aktivitetet e disqeve I/O, sidomos po t'i referohemi kohës së shkrimit [35]. Referuar figurës 6.7 do të tentojmë të bëjmë një megrim të lajmëruar duke dërguar një 0 në heartbeat dhe duke aktivizuar skriptin *cakto_destinacion* i implementuar në nivelin përdorues i cili do të bëjë të mundur të komunikojë me protokollin iSCSI në kernel për të caktuar destinacionin.

Këtë shembull do ta implementojmë duke ngritur edhe Hypervisor të tjerë si KVM dhe OpenVZ. Hypervisori KVM së bashku me emulatorin QEMU sic kemi shprehur dhe më parë do të kryejë një virtualizim të plotë, ndërkohë që OpenVZ është një virtualizim në nivel Sistemi Operativ, ku të gjithë conteneirat (SO Guest) do të ndajnë të njëjtin kernel. Sikurse bëmë për hypervisorin XEN veprohet edhe për dy Hypervisor-at e tjerë. Ndërkohë teknologjia na jep mundësi të bëjmë modifikime të Sistemeve Operative Guest mbi KVM duke reduktuar një trap mbi burimin fizik duke kaluar kështu në nivel Paravirtualizimi. Ndërkohë që nëse kemi procesorë AMD-V dhe Intel-VT si dhe pas instalimit të emulatorit QEMU mund të kalojmë pa problem Hypervisorin XEN në nivelin e virtualizimit të plotë.

Fillimisht skripti i ndërtuar në nivelin user thjesht bën sinkronizimin e një numri faqesh < 100 CP nga rregjistrat drejt Files së Përkohshme në zonën temprorary të diskut. Pra fillimisht nuk do të shtojmë metodën Delta Memory.

Për të parë tabelat realizojmë 5000 skripte të thjeshtë në C të cilët gjenerojnë numra random dhe stringa karakteresh. Gjithashtu në këto skripte gjenerohen edhe pamje dinamike grafike 2D dhe 3D mbi ekran. Fillimisht si sistem për menaxhimin e këtyre të dhënave është shfrytëzuar Oracle 11g.

Fillimisht testojmë 65 procese që zënë mesatarisht 3175GB në Memorje. Numri i Standby dhe Busy Dirty është 24000 + 96 me një total 96MB Standby Dirty Page dhe 386KB Busy Dirty Page në memorien e CPU-së (faqja eshte 4KB). Sic shkruam dhe më lart *Heartbeat* skript, simulon batch skript për vrasjen e të gjitha proceseve:

Point -> TLB

Execute ->ps

If process==active{

Send heartbeat =0 to iSCSI

iSCSI <-> cakto destinacion

Kill process}

If P=1&M=1

Set Flag Synch

If Synch Flag in Register <100

Stop Goto Migration:

:Lexo CPU_Consumption

If CPU1>CPU2

Load Balancing – migro

6.2.1 Krahasimi i parametrave me aplikacion dhe pa aplikacion mbi rrjetin ethernet

Realizojmë një tabelë për matjen e kohës totale të migrimit, kohës së bllokimit dhe trafikut, brenda rrjetit lokal LAN në rastin me dhe pa aplikacion.

Me / Pa Soft - ME LAJMERIM – LAN –iSCSI with TOE – SYN

Tabela 6.4-Matja e kohës totale të migrimit, kohës së bllokimit dhe trafikut, brenda rrjetit lokal LAN në rastin me dhe pa aplikacion gjatë migrimit

Lloji i	Nr i	Memorja	CPU	Koha e	Downtime	Over-	Nr i faqeve

HV	proçeseve	Shfrytz.MB	Konsm.%	migrimit (s)	ms	head %	të mbetura
XEN -PV	65-0	3166 3175	10,5 16,2	24,4 1752	300 2435	0,06 6,2	13 – 884
XEN -FV	65-0	3166 3177	14,4 18,7	29,5 1964	440 2964	0,07 8,5	24 – 932
KVM - PV	65-0	3167 3176	13,7 17,6	26,6 1905	410 2822	0,06 8,4	16 – 912
KVM - FV	65-0	3166 3175	14,6 18,8	34,8 2364	660 3799	1,01 0,9	30 - 1036
OPENVZ	65-0	3165 3174	10,1 16,1	26,1 1784	330 2424	0,02 5,1	12 – 876

Me / Pa Soft - PA LAJMERIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim ne ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të mbetura
XEN -PV	65-0	300 2435	17535 Block	0,06 6,2	3,5 87	23 – 7757(8050)
XEN -FV	65-0	440 2964	18642 Block	0,07 8,5	3,7 92	33 – 7956 (8050)

KVM - PV	65-0	410 2822	17974 Block	0,06 8,4	3,5 88	25 – 7810 (8050)
KVM - FV	65-0	660 3799	21005 Block	1,01 10,9	3,9 96	41 – 8020 (8050)
OPENVZ	65-0	330 2424	16604 Block	0,02 5,1	3,3 88	24 – 7715 (8050)

Para Migrimit Me / Pa Soft - LAN –iSCSI with TOE – SYN- Makina Burim / XEN

Memorja Shfrytz.MB (vlera maksimale)	CPU Konsum.% (vlera maksimale)
3182 3175	17,9 13,4
3194 3177	22,5 15,6
3182 3176	20,1 14,7
3179 3175	27,8 16,2
3180 3174	18,0 13,6

6.2.2 Matja e kohës totale dhe kohës së përgjigjes gjatë shkrimit të skripteve

Një tjetër problem këtu është sa do të jetë koha e përgjigjes ndaj përdoruesit. Këtu unë po punoj me sisteme Interaktive dhe ky element ka shumë rëndësi. Koha e përgjigjes nuk mund të bëhet e bezdisur për përdoruesin që po shkruan një tekst të caktuar. Për këtë unë do të ritestoj 5000 skriptet dhe do të mas kohën totale të shkrimit të numrave për

5000 skripte në rastin kur nuk kam instaluar aplikacionin dhe në rastin kur e kam instaluar aplikacionin. Eksperimentin do ta kryej për rastin e shkrimit direkt në një makinë fizike dhe më pas në Hypervizor të ndryshëm. Makinat janë ato të cilat janë përcaktuar në shembujt e mësipërm. Për të matur kohen totale të ekzekutimit si edhe kohën mesatare të përgjigjes do të perdor një skript në gjuhën e programimit në C të quajtur *ResponseC*.

Tabela 6.5- Koha totale e shkrimit të 5000 skripteve në makinën fizike dhe në Hypervisor të ndryshëm

Lloji i Makinës	**Koha totale e ekzekutimit pa aplikacion**	**Koha totale e ekzekutimit me aplikacion**
Shkrimi në Makinë fizike	296 sekonda	316 sekonda
Shkrimi mbi XEN-PV	426 sekonda	766 sekonda
Shkrimi mbi XEN-FV	510 sekonda	1260 sekonda
Shkrimi mbi KVM – PV	490 sekonda	889 sekonda
Shkrimi mbi KVM –FV	604 sekonda	1456 sekonda
Shkrimi mbi OpenVZ	420 sekonda	695 sekonda

.

Nëse testimin do ta verifikonim duke marrë kohën e përgjigjes, që do të ishte shfaqja në ekran e numrave të shkruar duke u mbështetur në karakteristikat Hardware të cituara më sipër, do të merrnim këtë tabelë:

Tabela 6.6- Koha e përgjigjes të 5000 skripteve në makinën fizike dhe në Hypervisor të ndryshëm

Lloji i Makinës	**Koha e përgjigjes pa**	**Koha e përgjigjes me**

	aplikacion	**aplikacion**
Shkrimi në Makinë fizike	60 ms	680 ms
Shkrimi mbi XEN-PV	460 ms	3852 ms
Shkrimi mbi XEN-FV	820 ms	4984 ms
Shkrimi mbi KVM – PV	790 ms	4253 ms
Shkrimi mbi KVM –FV	1140 ms	6650 ms
Shkrimi mbi OpenVZ	410 ms	3760 ms

.

Atëhere le të fillojme me interepretimin e tabelave të mësiperme dhe mjetet që përdora për të marrë këto rezultate.

Fillimisht për të matur konsumimin e CPU-së në Hypervisor-in XEN përdorim komandën xentop –b. Gjithsesi ky skript shfaq rezultatin e konsumimit të saj vetëm për një cast. Për të marrë një mesatare në vlerë të konsumimit të CPU-së përdorim një skript në C të quajtur *AverageXen* [35], i cili rregjistron vlerat e kësaj komande cdo 5 sekonda, duke përdorur funksionin avge. Ky skript është lokalizuar në direktorinë /etc. Për të matur konsumin e CPU-së në XEN-FV përdorim të njëjtën metodë.

Për të matur CPU-në në Hypervisor-in OpenVZ kam ngritur një skript, pasi nuk ka një komande specifike që mund ta bëjë këtë. Skripti të cilin e kam quajtur *traceproc1* është ngritur në C dhe është lokalizuar në /proc/vz/vstat [35]. Ky skript rregjistron proceset në gjendje ekzekutimi ose gati, në memorien e përbashket midis Conteneirat dhe Hypervisor. Skripti skanon në këtë memorie statusin e cdo procesi në skedarin vstat. Cdo proces ka një bit wakeup në Rregjistrin e Statusit të Procesit. Nëse biti ka vlerën 1 procesi është duke u ekzekutuar dhe nëse ka vlerën 0, procesi është në gjendje gati. Në këtë skript të lokalizuar në /proc/vz është impelementuar formula:

$$\text{Vlefshmeria e Proceseve} = \frac{\text{Koha per cdo proces aktiv}}{\text{Koha totale e CPU - se}} \times 100\% \quad (6.2)$$

Shuma e proçeseve aktive të mundshme = Aktivitetin e CPU-së.

Gjithsesi sic jepen në artikullin [35] përsëri nuk mund të masim dot vlerën e konsumimit në CPU, meqënëse kur proceset janë në gjendje idle ato sërish shpenzojnë procesim. Kështu ndërtohet një semafor [36] për ta derguar në gjumë procesin, në këtë mënyrë nuk konsumohet CPU-ja. Variablat semafor janë ngritur në C në skriptin e quajtur *Semafore* dhe do të ruajnë ID për të gjitha proceset idle. Informacioni është marrë nga skripti *Traceproc1*.

Për çdo proces pasiv gjenerohet një thread i cili dërgon një sinjal te këto procese. Në këtë mënyrë proceset pasive transformohen në procese gjumë - "sleep". Në momentin kur CPU dërgon një mesazh Interrupt, për zgjimin e një nga proceset në gjendjen gjumë, skripti *Semafor* merr kete sinjal, lexon ID e procesit të thirrur duke e rregjistruar adresën dhe vlerën e tij në një rregjistër të përkohshem dhe më pas thërritet thread-i pergjegjës. Në këtë mënyrë thread-i i sapokrijuar nga vlera e variablit mutex=1, do të zgjojë procesin në gjumë.

Në fakt kjo është një ndërmarrje e vështirë pasi thread-i është impelementuar në hapësirën përdorues, cka do të thotë se procesi mund të kalojë në gjumë përgjithmonë. Pas modifikimit të skriptit *traceproc1*, konsumimi i CPU-së në Hypervisorin OpenVZ do të jetë:

$$\text{Konsumimi i CPU} = \frac{\text{Shuma e proceseve aktive}}{\text{Numri Total i Proceseve}} \quad (6.3)$$

Figura më poshtë jep një skemë grafike të qartë për matjen e konsumimit të CPU-së.

Së fundmi për të matur konsumimin e CPU-së në Hypervisorin KVM përdoret një mjet i quajtur *SAR Utility* [35]. Sikurse *XenTop* për matjen e proçesimit në XEN ashtu edhe ky mjet mat kohën e proçesimit mbi Hypervisorin KVM për një çast të caktuar. Në

këtë mënyrë për matjen e vlerës mesatare të konsumimit të CPU-së, kam krijuar një skript në C te lokalizuar në */dev/kvm* të quajtur *AverageKVM*. Ky skript do të përdor nje funksion te quajtur ***avge*** per konsumimin e CPU-së. Ky skript mund të implementohet si në rastin e virtualizimit të plotë ashtu edhe të para-virtualizuar.

Për të matur konsumimin e memorjes së 65 proçeseve tona të gjeneruara si pasoj e një dyzine me skripte në C, mbi CentOS do të nisim fillimisht me Hypervisorin XEN.

Nisur nga artikulli [38], përdor një mjet të quajtur *MemAccess*. Ky mjet mund të shfrytëzohet dhe në migrimin live të kombinuar me një skript i cili përshkruan një vlerë mesatare. Skriptin e emëroj *MemCXen*.

Në OpenVZ përdor një mjet të quajtur *stream_tool* [37].

Edhe ketu si në të gjitha rastet e tjera jepet vetëm një vlerë "flash" e shfrytëzimit të memorjes. Por ky skript fatkeqësisht nuk mund të masë ndryshimet e menjëhershme dinamike që vijnë si pasojë e fluksit të vazhdueshem të përseritjes së faqeve të modifikuara gjatë kohës së migrimit[35]. Për këtë arsye fillimisht kam ngritur një skript të quajtur *MemOVZ* në gjuhën C i cili llogarit numrin e faqeve të modifikuara dhe e shumezon atë me madhësinë e faqes. Gjithsesi ky skript nuk jep një vlerë të mirë dhe të saktë në momentin kur kemi një dështim (page-fault). Atëhere implementoj një mjet të quajtur *Bonnie++*[39] të cilin e përshtas për të matur bandwidth-in midis dy makinave fizike gjatë migrimit. Duke shënuar me RAM_V1 memorjen në makinën burim dhe RAM_V2 në makinën e destinacion, mund të llogaris numrin total të faqeve të migruara:

$$(Nr.i\ Faqeve\ të\ Migruara) \quad = \frac{Madhësinë\ Totale(B)}{Madhësinë\ e\ faqes\ (B)} \qquad (6.4)$$

Duke u nisur nga benchmarku stream në çastin fillestarë do të kemi:

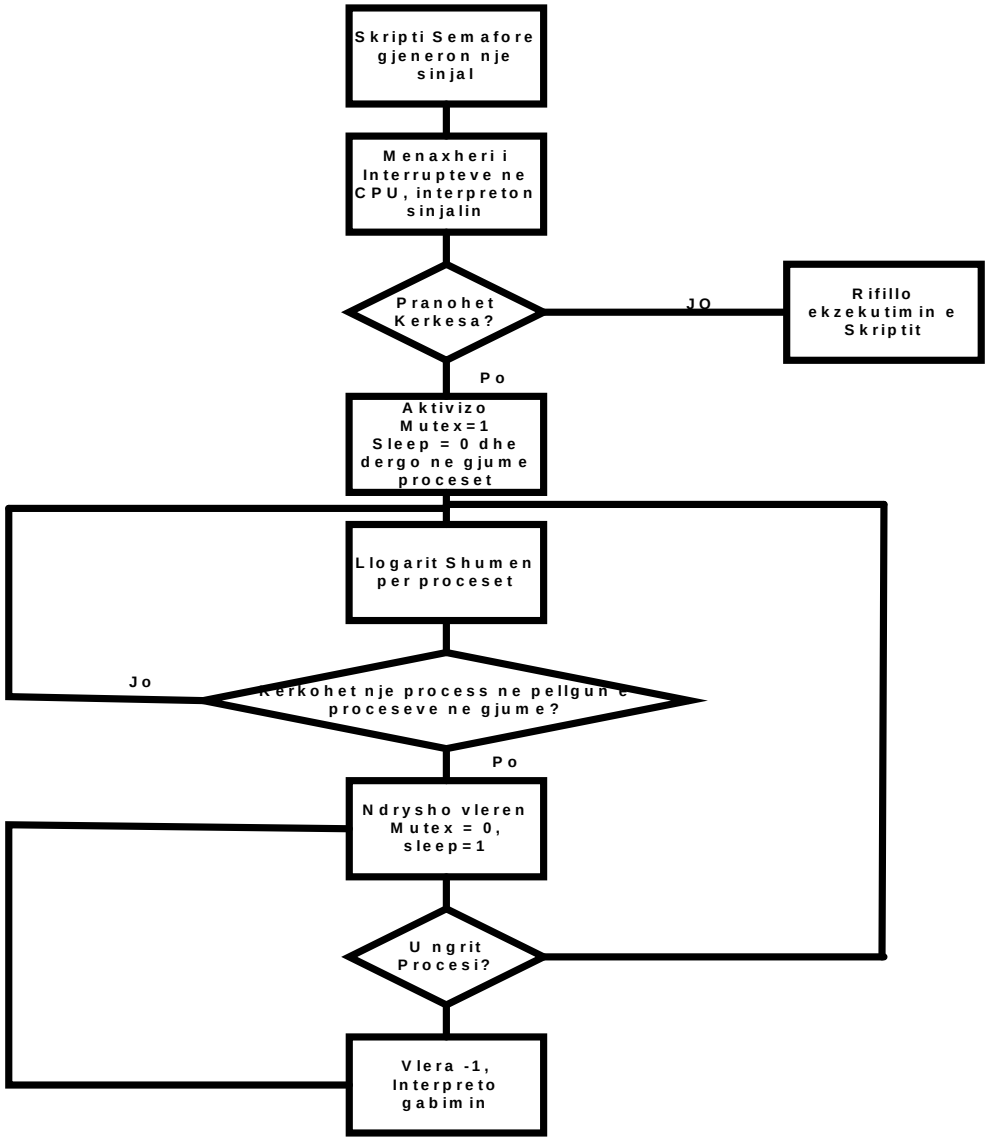


Figura 6.8-Algoritmi i llogaritjes së faqeve të migrura në Hypervisorin OpenVZ

(Memorja e Shfrytëzuar) $= Nr.e\ faqeve\ të\ lira + Nr.e\ faqeve\ të\ modifikuar * Nr.e\ faqeve\ të\ dështuara * Ma\ hësinë e\ faqes$ (6.5)

Ndërkohë për të matur vlerën e memorjes së shfrytëzuar në KVM implementoj një mjet open-source të quajtur *stress-tool*. Këtë mjet e kam përdorur duke ju referuar artikullit [40].

Për të matur kohën totale të migrimit dhe downtime do të përdorim një simulator të quajtur *Stress tool* [18]. Gjithashtu për matjen e downtime do të përdorim dhe një skript te quajtur DownC i cili llogarit kohën e bllokimit në momentin e dërgimit të statusit të ekzekutimit nga makina burim drejt asaj destinacion. Ky skript është i shkruar në gjuhën C dhe përdor si mjet vlerat e mara nga mjeti *Bonnie+*.

Për matjen e numrit të proçeseve mjafton të shtypim komanden ***ps***

Ndërkohë për të matur numrin e faqeve të mbetura pas iterimit të parë do të përdorim një skript i cili numëron faqet e modifikuara që në momentin e migrimit. Karakteristikat e tyre janë:

1.Ekzekutimi në rregjistrat e Proçesorit

2. Modifikimi.

Skripti quhet *Iterim+* dhe bazohet në një tabelë *Hash* e cila nga ana e vet ka të ruajtur të gjitha rekordet për faqe. Dmth kemi një kopje të tabelës së faqeve të hashuar në indekse. Çdo numër i plotë simbolizon një record te quajtur *Entry*. Ky script kryen dy funksione:

Të bëj diferencen midis faqeve të migruara dhe atyre qe u modifikuan gjatë proçesit të migrimit.

Për të matur kohën totale të ekzekutimit të skripteve në Diskun lokal, mund të përdorim mjete të ndryshme dhe ti përshtasim ato. Gjithsesi për të matur kohën e shkrimit në disk kam ndërtuar një tjetër skript. Ky skript mban adresat e të gjitha aktiviteteve të shkruara në disk nga momenti t0. Ky moment përkon me fillimin e programit të parë të përcaktuar nga ne. Skripti *ResponseC* gjeneron një thread agjent. Për këtë arsye nevojitet ndihma e kernelit. Ky thread do të përgjojë momentin fillestarë dhe perfundimtarë të

shkrimit të programeve në disk. Në momentin kur shkruhet programi i fundit, kjo kohë caktohet si koha t1. Në këtë çast threadi i cili gjatë gjithë kohës ka qenë në gjendje gjumi do të gjurmojë shkrimin e programit të fundit në disk. Pas kësaj ai sinjalizon skriptin për mbarimin e ekzekutimit. Skripti i cili ka rregjistruar kohën fillestare t0 dhe të mbarimit t1 bën diferencën dhe gjen sa është kjo kohë.

Kërkesa e vetme për të matur këto kohë është që programet e percaktuara në skript të ekzekutohen sekuencialisht. Modifikimi në kernel është shumë i lehtë dhe unik për të gjithë Hypervisorët. Thjeshtë do të përdoren dy thirrje *sleep* dhe *wake up* dhe një funksion *thread_create* për krijimin e threadit që do të caktojë fillimin dhe fundin e shkrimit. Skripti është quajtur *TimeC+*.

E së fundmi për të matur kohën e përgjigjes do të realizojmë sërish një skript i cili do të bashkërendojë me programin tonë. Ky skript do të mat kohën nga momenti kur shkruajmë në një faqe dhe deri kur ajo ruhet në disk. Pas kësaj do të gjejmë një mesatare të kohës së përgjigjes për çdo fjalë. Në fakt, siç parashtrohet edhe më sipër, janë gjeneruar numra dhe në bazë të tyre janë caktuar kohët përkatëse. Këtë skript do ta quajmë *ResponseC*, për shkak se edhe ky skript është shkruar në gjuhën e programimit C. Skica e lidhjes së tij me programin kryesore do të jetë si më poshtë:

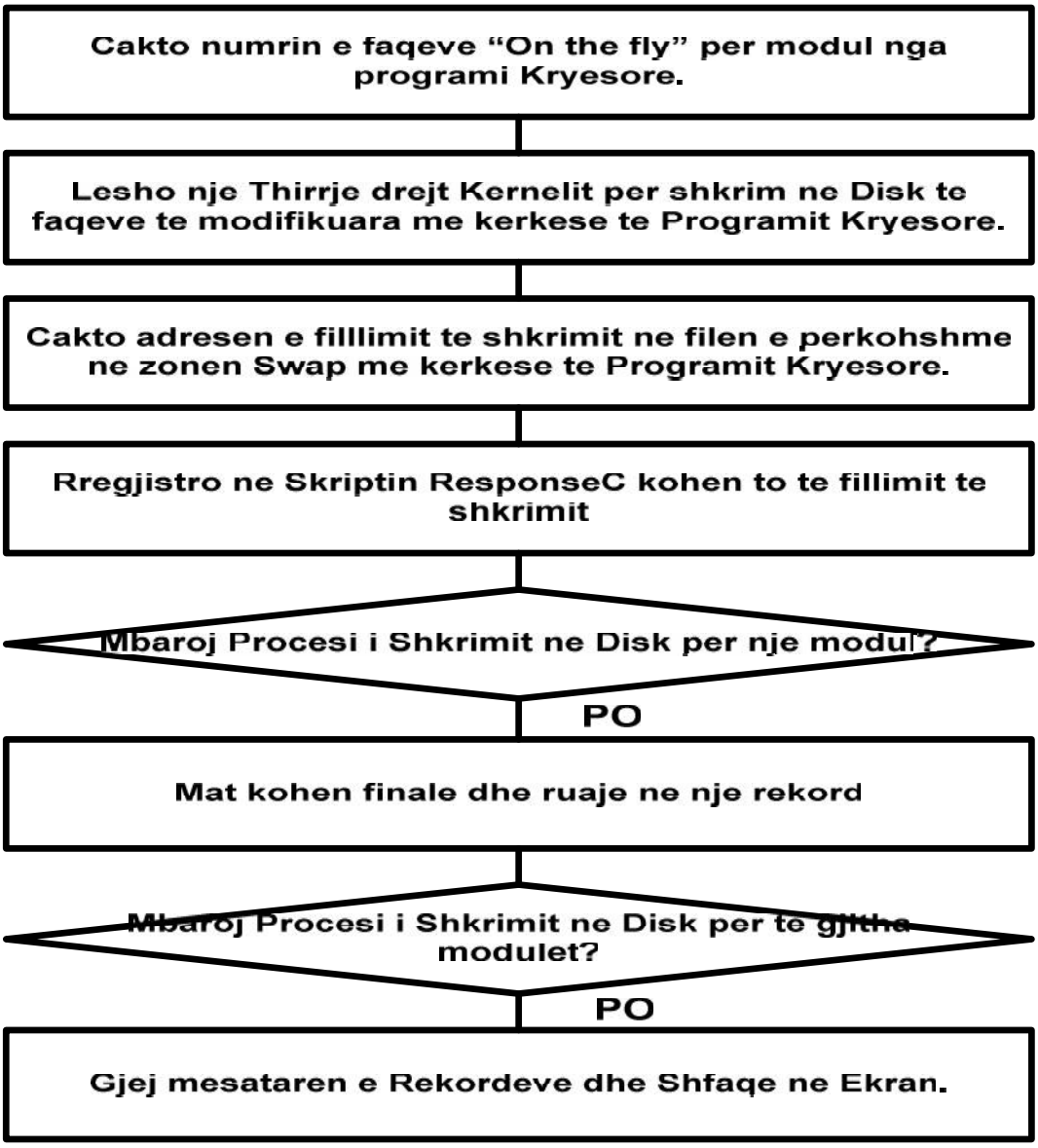


Figura 6.9-Skripti për matjen e kohës së përgjigjes së shkrimit të numrave rastësor në disk.

Le të fillojmë me interpretimin e tabelave të nxjerra nga eksperimentet e mësipërme:

Siç u përmend më sipër janë ekzekutuar 5000 skripte që rendisin rreth 65 proçese. Siç shikohet në tabelat 5.5 dhe 5.6, në të gjitha rastet kur kemi ngritur aplikacionin tonë,

ndjehet një rritje e lehtë e ngarkesës në memorjen fizike, krahasuar me rastin pa këtë aplikacion. Arsyeja është numri i faqeve shtesë që fut proçesi i menaxhimit të aplikacionit në hapsirën përdorues. Ajo që vihet re është se ngarkesa duket më e lartë në Hypervisorin XEN-FV, pasi këtu duhet emuluar aplikacioni QEMU dhe driverat e kartës së rrjetit *e1000*, të cilat rezervojnë një vend shtesë në memorje.

Ndërkohë ajo që vihet re më shumë është ngarkesa shtesë e CPU-së që shkaktohet nga shkrimi "*write back*" në disk për një numër të caktuar faqesh të modifikuara. Pra disku perveç se duhet te menaxhojë memorjen fizike për shkrimin e numrave random, duhet dhe të menaxhoje memorjen sekondare së bashku me një proçes shtesë për të shkruajtur në të. Ajo qe vihet re është rritja shumë e lartë në Hypervisorin KVM sidomos në teknikën me Virtualizim të plotë. Arsyeja është se vetë ky Hypervisor ka një performancë të keqe në aksesimin e disqeve Hyres/Dales [4].

Por të metat që shikohen te shfrytëzimi i memorjes dhe i proçesorit, përmisohen në kohën e Migrimit. Nëse bëjmë një krahasim të rastit me aplikacion dhe atij pa aplikacion te kohës totale të Migrimit shikohet një përmisim i ndjeshëm krahasuar me rastin e dytë. Kjo vjen për faktin se koha totale e migrimit tani do të jetë shumë e:

1.Kohës së migrimit të faqeve të lira

2. Kohës së migrimit të faqeve të modifikuara

3. Kohës së migrimit të faqeve të iteruara

Por hapi i tretë që është dhe rasti më i keq në kohën Totale të migrimit, falë ngritjes së aplikacionit, do të reduktohet në përsëritjen e një numri mjaft të vogël të faqeve të modifikuara. Numri i faqeve do të reduktohet në një numër dy shifror.

E njëjta gjë vihet re edhe për zvoglimin e kohës së bllokimit. Kjo kohë reduktohet mjaft pasi tani do të transferohet një numër i vogël faqesh të modifikuara shoqëruar me disa Byte të gjendjes së proçesorit.

Referuar pikes së tretë, do të shpjegohet dhe fakti i reduktimit të trafikut të migrimit dhe numrit të faqeve të mbetura.

Nëse do të shkojmë në tabelën 5.4 në rastin e migrimit pa lajmerim, vihet re një degradim i parametrave të mësiperm. Kjo gjë është mëse normale pasi po i referohem situatës së një katastrofe si: rënie zjarri, nderprerje energjie, tërmet etj. Kjo do të thotë që proçesi *heartbeat* i cili sinjalizon për avari nuk do të funksionojë. Kjo do të thotë që të gjitha faqet e modifikuara dhe te pashkruara *write back* në disk do të humbasin. Falë aplikacionit të ndërtuar ne hapsirën përdorues, ky numër është shumë më i vogel se ne rastin kur nuk kemi ngritur aplikacionin.

Nëse shkojmë në tabelën 5.5, do të shohim se koha totale për ekzekutimin total të aplikacioneve është më e madhe në rastin kur ngrihet aplikacioni se në rastin kur nuk e kemi atë. Kjo për vetë faktin e ngritjes së metodës *write-back* për një numër të kufizuar faqesh të modifikuara.

Dhe së fundi, për të testuar kohën e përgjigjes, e cila mund të quhet si një nga pikat më kritike, kalojmë në tabelën 5.6. Në këtë tabelë vihet re një rritje e lartë e kohës së përgjigjes së skripteve në rastin kur kemi instaluar aplikacionin tonë gjatë gjenerimit të numrave të rastësishëm. Kjo është normale pasi duhet të shkruajmë në disk për një numër të kufizuar faqesh te modifikueshme. Përdorimi i shpeshtë i metodës *write-back* normalisht e rrit së tepërmi kohën e përgjigjes për shkrimin në disk.

6.2.3 Përmirësime të Algoritmit Live Improve

Programit tonë do ti bëjmë dy përmirësime:

Ngritja e metodës Delta Memory

Ngritja e metodës Cold Migration ose Stop and Copy.

Siç e përmendëm më sipër kjo metodë do të shërbejë për ngritjen e një tabele hash brenda faqes së memorjes ku do të bëjë diferencën midis faqeve të modifikuara së fundmi. Siç dihet nga [10], gjatë shkrimit në një faqe një pjesë e saj mund te

modifikohet, kjo do të thotë që nuk është e nevojshme të dërgojmë të gjithë faqen nga e para, por do të mjaftonte vetëm pjesa e modifikuar të ruhej në disk. Për këtë duhet të bëjmë disa modifikime në kernel. Tabela hash e ngritur në nivel aplikacioni do të diktojë në aktivitetin e një threadi në kernel i cili do të rregjistrojë të gjitha ndryshimet brenda faqes. Kontrolli i faqes është një proçes rutinë i cili nuk varet nga lloji i Hypervisorit që kemi ngritur. Kjo do të na çojë në një unifikim të modelit të kontrollit në kernel.

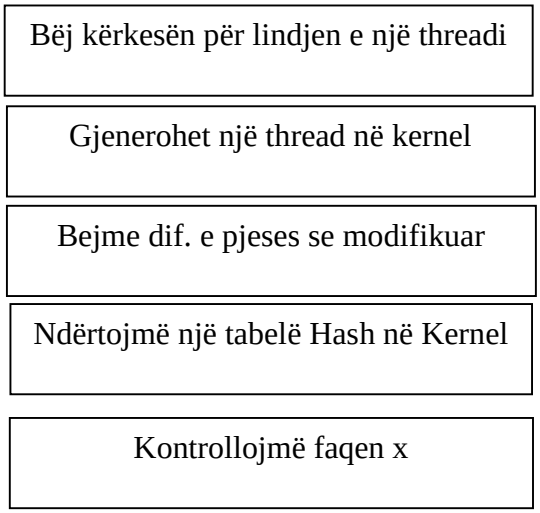


Figura 6.10-Struktura e ngritjes së tabelës Hash në skriptin që kontrollon ndryshimet e reja në faqe

Gjithashtu për faqet e lira kemi implementuar metodën e kompresimit të faqeve siç e trajtuam më lart.

Nga ky modifikim do të marrim tabelat e mëposhtëme:

Tabela 6.7-Ngritja e teknikës Delta Compression në një rrjet LAN

Me / Pa Delta Compression - ME LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	**Nr i proçeseve**	**CPU Shfrytz.%**	**Koha e migrimit**	**Downtime ms**	**Over-head %**	**Nr i faqeve të**

			(s)			mbetura.
XEN -PV	65-0	17,9 20,1	24,4 21,5	300 235	0,06 0,04	13 – 10
XEN -FV	65-0	22,5 24,7	29,5 26,7	440 395	0,07 0,05	24 – 22
KVM - PV	65-0	20,1 22,8	26,6 25,2	410 385	0,06 0,04	16 – 12
KVM - FV	65-0	27,8 30,6	34,8 32,9	660 640	1,01 1,00	30 – 26
OPENVZ	65-0	18,0 20.4	26,1 23,2	330 300	0,02 0,01	12 – 9

Me / Pa Delta Compression - PA LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim ne ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të mbetura
XEN -PV	65-0	300 235	17535 8964	0,06 0,04	3,5 2,8	23 – 18

XEN -FV	**65-0**	**440** **395**	**18642** **10500**	**0,07** **0,05**	**3,7** **3,1**	**33 –** **30**
KVM - PV	**65-0**	**410** **385**	**17974** **9964**	**0,06** **0,04**	**3,5** **2,9**	**25 –** **22**
KVM - FV	**65-0**	**660** **640**	**21005** **14250**	**1,01** **1,00**	**3,9** **3,2**	**41 –** **36**
OPENVZ	**65-0**	**330** **300**	**16604** **8690**	**0,02** **0,01**	**3,3** **2,7**	**24 –** **21**

.

Pra siç shikohet nga tabela ka një përmisim të ndjeshëm në Kohën Totale të Migrimit, Overheadin, Kohën e Bllokimit dhe Nr e faqeve te mbetura, duke konsumuar disi më shumë proçesor dhe memorje. Vlerat permisohen edhe për rastin kur nuk kemi bërë një lajmërim me pare.

Nëse e zgjeroj algoritmin duke përdorur metodën pa iterime të quajtur Stop and Copy, ose sic quhet ndryshe metoden Cold Migration te inicuar nga autoret: Timothy Wood, Prashan Shenoy nga Universiteti i Masachusas në vitin 2010. Kjo metodë do të reduktojë në miminum numrin e iterimeve, pasi dërgohen faqet e lira dhe të modifikuara, bëhet dërgimi i gjëndjes së CPU-së duke krijuar bllokimin e makinës burim, menjëherë. Në këtë metodë bëhet ndarja e faqeve të lira dhe të modifikuara në dy vargje të ndarë. Kjo do të thotë që në aplikacionin “Live Improve” do të shtoj një variabël *semaforë*. Ky variabël do të marrë vlerën 0 në momentin kur jepet urdhëri për migrim. Menjëherë sapo kjo vlerë të bëhet 0, do të gjenerohet një thread në kernel i cili do të dërgoj një sinjal bllokimi të aktivitetit të CPU-së.

Kjo do të thotë se me faqet e migruara, do të migrohet edhe flamuri i gjendjes së CPU-së.

Duke implementuar këtë metodë do të marrim rezultate të tjera:

Tabela 6.8-Ngritja e teknikës Cold Migration në një rrjet LAN

Pa / Me Cold_Migration - ME LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Humbura.
XEN -PV	65-0	20,1 - 20,1	21,5 20,3	235 247	0,04 0,03	10
XEN -FV	65-0	24,7 – 24,8	26,7 25,8	395 428	0,05 0,04	22
KVM - PV	65-0	22,8 – 22,8	25,2 24,5	385 432	0,04 0,03	12
KVM - FV	65-0	30,6 – 30,5	32,9 32,1	640 661	1,00 0,08	26
OPENVZ	65-0	20.4 – 20,4	23,2 22,2	300 318	0,01 0,005	9

Pa / Me Cold Migration - PA LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim në ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të humbura
XEN -PV	65-0	235 227	8964 9255	0,04 0,03	2,8 2,7	18
XEN -FV	65-0	395 388	10500 11389	0,05 0,04	3,1 3,0	30
KVM - PV	65-0	385 382	9964 10355	0,04 0,03	2,9 2,7	22
KVM - FV	65-0	640 631	14250 14636	1,00 0,08	3,2 3,0	36
OPENVZ	65-0	300 288	8690 8874	0,01 0,005	2,7 2,6	21

.

Nëse verifikojmë tabelat e mësiperme shikohet se me implementimin e teknikave Me Cold Migration ka një përmirësim shumë të lehtë në kohën totale të migrimit dhe trafikut,duke rritur kohën e bllokimit. Koha e bllokimit shtohet për arsye të humbjes së faqeve të modifikuara së fundmi. Gjithsesi ajo që do të fitohet është trafiku, i cili do të reduktohet lehtësisht për shkak të mungesës së përsëritjeve me faqet e modifikuara. Edhe këtu shikohet që virtualizimi i plotë fut një trafik më të lartë, siç është shprehur edhe më sipër, arsyeja kryesore mbetet ekzekutimi i dy instruksioneve trap nga aplikacioni përdorues, drejt GuestOS dhe më pas që aty drejt Hypervisorit.

Nëse e krahasojmë studimin tonë me artikullin ku është implementuar kjo metodë, shikojmë një diferencë. Arsyet janë:

1.Testimi në artikull është bërë vetëm për Hypervisor XEN-PV

2. Testimet janë bërë duke implementuar metodën në nivel kerneli dhe jo në nivel aplikacioni. Ngritja në nivel aplikacioni e ul shume efiçensën, pasi futet kompleksitet në kohë dhe në ekzekutim nga shtimi i një instruksioni shtesë “Trap”.

Ajo që vlen të theksohet është numri i faqeve të humbura në kolonën e fundit. Gjë që nuk vihej re te metodat e mëparshme. Kjo vjen sepse ne rastet e metodave të implementur më herët do të viheshin re përseritje ose iterimi. Ndërkohe që në metodën “*Cold Migration*” nuk kemi këtë fenomen. Kjo do të thotë që faqet e fundit të modifikuara nuk do të kenë shansin të ridërgohen në makinën destinacion, çka do të thotë se ato do të humbasin përgjithmonë. Siç shikohet po të krahasojmë tabelat 6.8, numri i faqeve të humbura në metodën *Cold Migration* është i barabartë me numrin e faqeve të mbetura në metodat paraardhëse. Kjo është normale pasi faqet e mbetura, nuk janë gjë tjetër, veçse faqet e modifikuara që duhet të ridërgohen sërish në destinacion.

Duke u nisur nga artikulli [20], arsyetoj per përmirësimin e algoritmit ***Live Improve***. Siç e kemi përmendur edhe më sipër implementimi i kësaj teknike është ngritur në Hypervisoret Xen dhe KVM duke modifikuar kernelin përkatës. Do ta zgjeroj modifikimin e kësaj metode edhe për Virtualizime në nivel Sistemi Operativ, duke modifikuar në OpenVZ. Modifikimi këtu do të jetë duke bërë një modifikim në kernel dhe përkatësisht duke ndarë në dy lista të pavarura, faqet e lira nga faqet aktive. Kështu gjatë migrimit do të dërgohen faqet e lira ndërkohë që faqet e modifikuara duke ndjekur metodën *Cold Migration* do të bllokohen.

Metoda që kam ngritur është bazuar në Listat e Lidhura Zinxhir sipas këtij algoritmi:

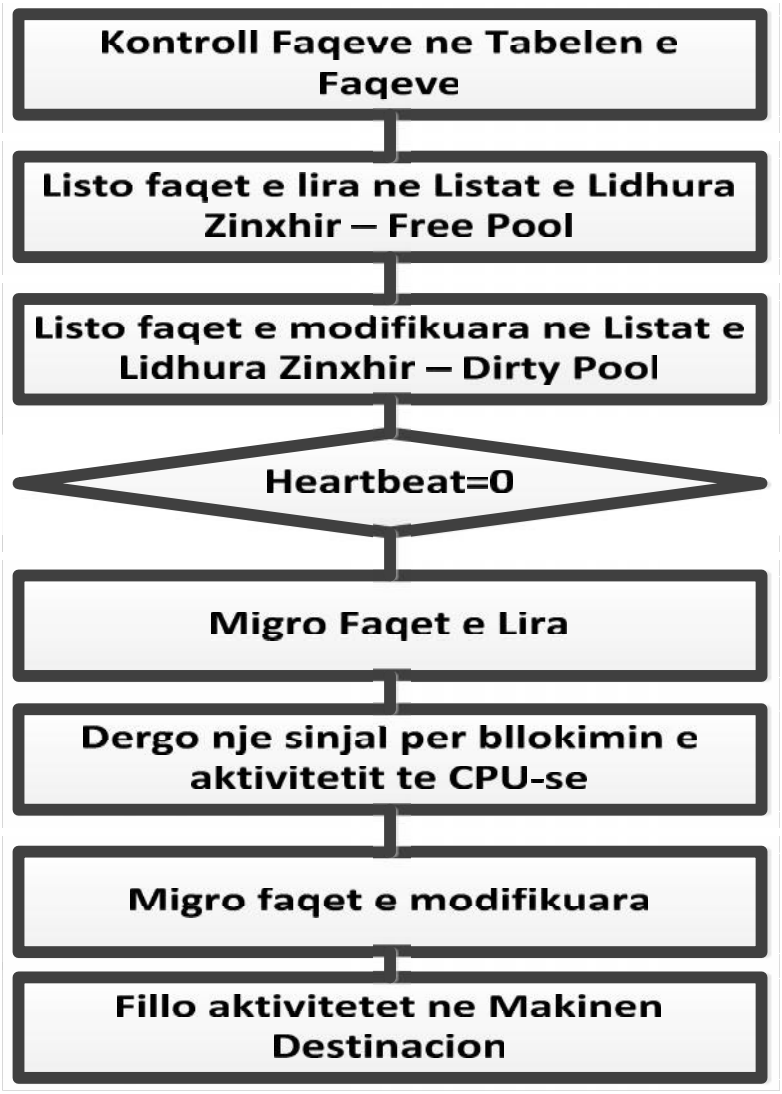


Figura 6.11-Algoritmi i teknikes Cold Migration midis Hypervisoreve te ndryshem duke implementuar formen me lista te lidhura zinxhir.

Faqet e lira të futura në një pellg të përbashkët ndjekin rregullat zinxhir. Kjo do të thotë se kosto e krijimit apo e fshirjes së tyre është e ulët. E njëjta gjë ndodh dhe për faqet e modifikuara. Kur një faqe pastrohet ajo futet në listen e faqeve të lira. Kjo nuk shton koston në mënyrë të ndjeshme, pasi vetëm duhet të shtohet një pointer.

Nga kjo metodë kemi nxjerrë keto rezultate:

Tabela 6.9-Ngritja e teknikës me caktimin e listave të lidhua zinxhir në metodën "Cold Migration" në një rrjet LAN

Me / Pa Lista Aktive - ME LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve te Humbura.
XEN -PV	65-0	20,5 - 20,1	17,6 20,3	282 247	0,02 0,03	10
XEN -FV	65-0	25,6 – 24,8	22,9 25,8	495 428	0,04 0,04	22
KVM - PV	65-0	24,1 – 22,8	22,5 24,5	499 432	0,02 0,03	12
KVM - FV	65-0	32,2 – 30,5	30,7 32,1	730 661	0,08 0,08	26
OPENVZ	65-0	20,6 – 20,4	19,1 22,2	376 318	0,04 0,05	9

Me / Pa Lista Aktive - PA LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim	Downtime pa lajmërim	Overhead me lajmërim	Overhead pa lajmërim	Nr i faqeve te

		ms	ne ms	%	%	humbura
XEN -PV	65-0	282 247	9644 9255	0,02 0,03	2,6 2,7	18
XEN -FV	65-0	495 428	12150 11389	0,04 0,04	3,0 3,0	30
KVM - PV	65-0	499 432	11462 10355	0,02 0,03	2,6 2,7	22
KVM - FV	65-0	730 661	15610 14636	0,08 0,08	3,0 3,0	36
OPENVZ	65-0	376 318	8961 8874	0,04 0,05	2,5 2,6	21

.

Nga të dhënat e tabelave të mësipërme ajo që vihet re është se fitojmë kohë në migrimin total të faqeve por kjo shoqërohet me një rritje të ndjeshme të Trafikut dhe kohes se bllokimit. Arsyeja e saj është se do të na mungojnë iterimet. Mos gjetja e faqeve të modifikuara së fundmi shkakton penalitet shtesë. Një reduktim i lehtë vihet re te trafiku i shkaktuar. Kjo pasi organizimi i faqeve në dy pellgje ndikon në uljen e tij, sidomos të sistemeve të paravirtualizuara, për shkaqe që i sqaruam më siper. Nderkohë, do të vihet re rritje te proçesimi i faqeve. Kjo gjë ndodh pasi kërkohet një punë shtesë nga proçesori për të bërë të mundur organizimin e tyre sipas bitit të modifikimit.Kontrolli i këtij biti dhe dërgimi i pointerit përkatës shkaktojnë një rritje të aktivitetit të CPU-së.

6.2.4 Përmirësime të algoritmit duke përdorur metodën e kompresimit

Çfarë ndodh nëse faqet e lira do të kompresoheshin, nisur kjo nga artikuj të ndryshëm të cilët i referohen kësaj metode. Por konkretisht kjo metodë është përshkruar qartë te [23], [27]. Do ta implementojmë këtë metodë ekstra, tashmë jo vetëm për Hypervisoret XEN por edhe për KVM dhe OpenVZ. Algoritmi *Live Improve* do të modifikohet disi, duke shtuar dhe gjetjen e biteve të shkrimit dhe prezencës, konkretisht bitet M dhe P, për M=0 dhe P=1 per çdo faqe të caktuar. Algoritmi i kompresimit MECOM [27],[38]. është shumë i përdorshëm dhe bazohet në disa koncepte matematikore *hash*. Dy metoda kompresimi të marra nga [19] për kompresim janë: *zlib* dhe *xdelta*. Algoritmet e implementuar në artikujt [20],[23],[26],[27], nuk kërkon asnjë modifikim për rastet e Hypervisoreve OpenVZ dhe KVM, pra kodi burim i këtyre hypervisoreve nuk është e nevojshme të modifikojë algoritmin, me disa perjashtime të vogla që konsistojnë në ndërfaqen e komunikimit të tij me këtë algoritëm, dmth në mesazhet e transmetuara *get* dhe *post* midis tyre. Një tjetër ndryshim është dhe lokalizimi i këtij algoritmi në Hypervisor të ndryshëm dhe konkretisht. Në hypervisorin XEN algoritmi do të modifikohet në */etc/xen* në KVM në pathin /dev/kvm dhe në OpenVZ do të përcaktohet duke konfiguruar në OpenVZ /etc/sysctl.conf=1 dhe /etc/vz.

Një modifikim tjetër do të bëhet në kernelin e Linux duke modifikuar librarinë LZMA duke përdorur metodën Pdelta [19].

Pas këtyre modifikimeve do të kemi një ulje të lehtë të trafikut por që do të shoqërohet me rritje të ndjeshme të konsumimit të CPU-së. Parametrat e tjerë ndikohen pak ose aspak nga ky modifikim. Për shembull Downtime ulet shumë shumë pak, gati e pandjeshme, pasi kompresimi nuk ndikon në faqet e modifikuara. Duke qenë se *Downtime* i referohet kryesisht faqeve të modifikuara, kjo do të thotë se ky parametër do të jetë i pandjeshem nga impelemtimi i këtij algoritmi. Gjithsesi faktori kryesore që përmirësohet është trafiku. Kjo vjen sepse reduktohet numri i faqeve zero në maksimum.

Tabela 6.10.A-Ngritja e teknikës Me Kompresim në nje rrjet LAN.

Pa / Me Kompresim - ME LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Humbura.
XEN -PV	65-0	20,5 - 24,8	17,6 17,2	282 266	0,02 0,008	10
XEN -FV	65-0	25,6 – 33,7	22,9 21,8	495 478	0,04 0,018	22
KVM - PV	65-0	24,1 – 29,2	22,5 21,6	499 481	0,02 0,007	12
KVM - FV	65-0	32,2 – 38,8	30,7 28,8	730 711	0,08 0,025	26
OPENVZ	65-0	20,6 – 23,4	19,1 18,2	376 358	0,04 0,015	9

Pa / Me Kompresim - PA LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim në ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të humbura

XEN -PV	**65-0**	**282** **266**	**9644** **9472**	**0,02** **0,008**	**2,6** **2,7**	**18**
XEN -FV	**65-0**	**495** **478**	**12150** **11876**	**0,04** **0,018**	**3,0** **3,0**	**30**
KVM - PV	**65-0**	**499** **481**	**11462** **11340**	**0,02** **0,0077**	**2,6** **2,7**	**22**
KVM - FV	**65-0**	**730** **711**	**15610** **15425**	**0,08** **0,025**	**3,0** **3,0**	**36**
OPENVZ	**65-0**	**376** **358**	**8961** **8895**	**0,04** **0,015**	**2,5** **2,6**	**21**

Kjo metodë është absurde të aplikohet në faqet e modifikuara pasi kjo do të shkaktonte një penalitet të kohës totale të migrimit, bllokimit dhe një degjenerim të CPU-së. Kjo gjë do të shkaktohej sepse, para se një grup faqesh të pastrohen, fillimisht ato duhej të kompresoheshin. Ajo që do të merrnim do të ishte katastrofë në lidhje me kohën e përgjigjes. Kështu ky eksperiment nuk vlen të merret në konsideratë. Të njëjtit mendim janë dhe autorë të ndryshëm të cilet kanë punuar me kompresimin e faqeve në makinat te ndryshme virtuale. Gjithsesi nëse do të testojmë vetëm kohën e përgjigjes do të bindemi sa e dëmshme do të ishte kjo metodë po të implementohej në faqet e modifikuara:

Pa / Me Kompresim të faqeve të modifikuara - ME LAJMËRIM – LAN –iSCSI with TOE, SYN.

Tabela 6.10.B Ngritja e teknikës Me Kompresim në një rrjet LAN duke përfshirë edhe faqet e modifikuara

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Humbura.
XEN -PV	65-0	20,5 - 46,8	17,6 210	282 4111	0,02 0,002	10
XEN -FV	65-0	25,6 – 67,8	22,9 256	495 6328	0,04 0,005	22
KVM - PV	65-0	24,1 – 59,1	22,5 224	499 5759	0,02 0,002	12
KVM - FV	65-0	32,2 – 71,2	30,7 337	730 8224	0,08 0,011	26
OPENVZ	65-0	20,6 – 42,3	19,1 217	376 4078	0,04 0,003	9

Po të shikojmë në tabelën e mësipërme kuptojmë që për një overhead të mirë do të degradojmë kohën totale të migrimit, shfrytëzimin e CPU-së, kohën e bllokimit. Gjithashtu përdorimi i metodës së kompresimit në rastin e kompresimit të faqeve nuk ndikon në uljen e numrit të faqeve të humbura, kjo për faktin se penaliteti që shtohet nga kompresimi i faqeve të modifikuara balancohet me numrin e madh të migrimit të tyre.

Ajo që vlen të shikohet është dhe koha e përgjigjes. Sa do të jetë kjo kohë në rastin kur shtojmë kompresimin e faqeve të modifikuara? Për këtë vlen tabela e mëposhtëme:

Tabela 6.11-Caktimi i kohës së përgjigjes në një rrjet LAN për teknikën me Kompresim

Lloji i Makinës	Koha totale e përgjigjes me kompresim të faqeve të modifikuara	Koha totale e përgjigjes me kompresim të faqeve të pa-modifikuara
Shkrimi mbi XEN-PV	11658 ms	3852 ms
Shkrimi mbi XEN-FV	15320 ms	4984 ms
Shkrimi mbi KVM – PV	13665 ms	4253 ms
Shkrimi mbi KVM –FV	21194 ms	6650 ms
Shkrimi mbi OpenVZ	10250 ms	3760 ms

.

Matjen e kohës së përgjigjes e masim me të njejtat metoda siç përdorëm më parë. Nga ajo që shikohet është degjenerimi i kohës së përgjigjes. Ajo shkon deri në rendin e 10 sekondave. Kjo vjen për shkak të kompleksitetit të kompresimit të futur në faqet e modifikuara. Nga kjo tabelë kuptojmë se nuk është aspak e nevojshme ndërtimi i një sistemi të tillë për faqet e modifikuara.

Duke u nisur nga artikulli [35] një metodë tjetër është ndërtimi një bashkësie pune efiçente duke përdorur algoritmat LRU dhe Splay Tree. Algoritmi Splay Tree është një pemë kërkimi e avancuar që fut disa avantazhe si:

Implementim i thjeshtë, Performancë të mirë në rastin mesatarë, memorje të vogël *footprint*, mundësi për të krijuar një strukturë të dhënash të qëndrueshme etj [21].

Me anë të këtij algoritmi do të bëhet i mundur kërkimi më i shpejtë i faqeve të lira dhe atyre të modifikuara, duke përdorur rregullat e pemës binare.

6.2.5 Përmirësime të algoritmit duke përdorur metodën e parashikueshmërise së bashkësisë së punës.

Duke implementuar dhe algoritmin LRU ndërtojmë metodën e parashikueshmërisë së bashkesisë së punës, duke u bazuar në konceptin e kohës dhe vendodhjes [20]. Me anë të kësaj metode tentojmë të zvoglojmë numrin e faqeve të modifikuara në një bashkësi pune të quajtur *Working Dirty Page*.

Por përveç faqeve të modifikuara në bashkësinë e punës futen edhe faqet e lexuara, dmth të pa modifikuara. Këto faqe i fusim në një bashkësi tjetër pune të quajtur *Working Zero Page*.

Modifikimi do të bëhet në algoritmin tonë ku gjatë kërkimit të faqeve të modifikuara dhe atyre të lira do të kërkohet nga threaded në kernel të ndërtojnë grupin e faqeve të lira dhe atyre të modifikuara. Këto faqe do të lexohen nga Tabela e Faqeve në MMU (Njësine e Menaxhimit të Memorjes). Më pas do të kërkohet implementimi i algoritmit LRU. Ky algoritëm do të implementohet së bashku me algoritmin Splay Tree dhe do të fillojë të parashikojë faqet e kërkuara me anë të metodës *demand paging* [41].

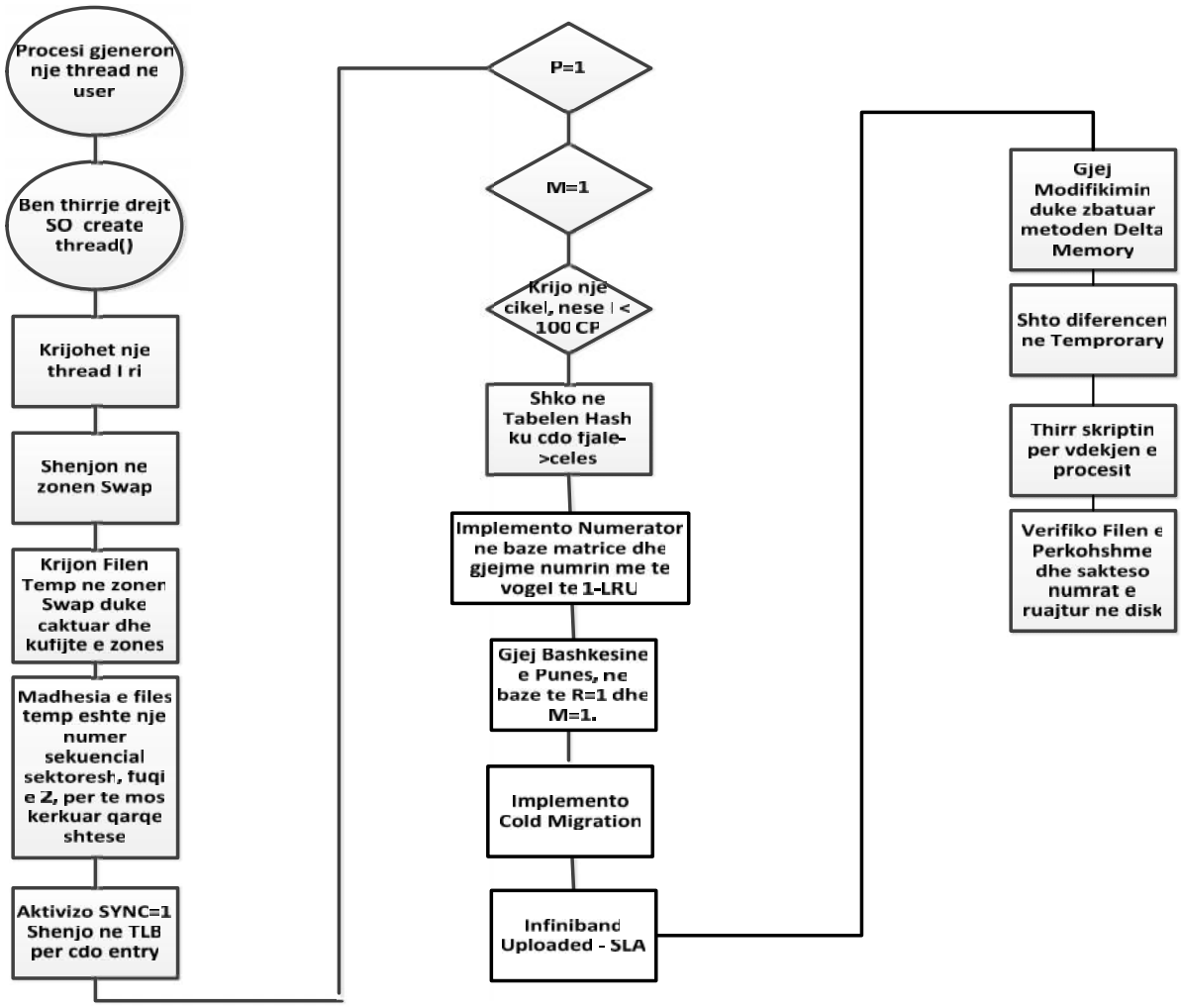


Figura 6.12-Implementimi i algoritmit LRU në metodën tonë për minimizimin e bashkësisë së punës.

Modifikimi në kernel këtu do të implementohet për Hypervisor të ndryshëm si: OpenVZ, KVM dhe XEN. Modifikimet do të jenë mbi CENTOS5,6 të HostOS për të tre Hypervisorët. Këto tre Hypervisor nuk do të ndikohen nga modifikimet në kernel. Në bazë të algoritmit që ngritëm do të kemi këto vlera:

Tabela 6.12-Ngritja e teknikës në një rrjet LAN duke caktuar bashkesinë minimale të punës me anë të algoritmit LRU

Me / Pa Parashikim (LRU+Splay Tree) - ME LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Humbura.
XEN -PV	65-0	27,4 - 24,8	15,6 17,2	240 266	0,007 0,008	8 10
XEN -FV	65-0	36,6 – 33,7	20,2 21,8	435 478	0,016 0,018	20 22
KVM - PV	65-0	32,5 – 29,2	18,9 21,6	466 481	0,007 0,007	11 12
KVM - FV	65-0	41,4 – 38,8	26,4 28,8	686 711	0,021 0,025	25 26
OPENVZ	65-0	26,4 – 23,4	16,1 18,2	301 358	0,011 0,015	8 9

Me / Pa Parashikim (LRU+Splay Tree) - PA LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim	Downtime pa lajmërim	Overhead me lajmërim	Overhead pa lajmërim	Nr i faqeve të

		ms	ne ms	%	%	humbura
XEN -PV	65-0	240 266	9231 9472	0,007 0,008	2,6 2,7	15 18
XEN -FV	65-0	435 478	11699 11876	0,016 0,018	3,0 3,0	27 30
KVM - PV	65-0	466 481	11115 11340	0,007 0,007	2,6 2,7	18 22
KVM - FV	65-0	686 711	13624 15425	0,021 0,025	2,9 3,0	33 36
OPENVZ	65-0	301 358	8645 8895	0,011 0,015	2,5 2,6	18 21

.

Shënim: Metoda e kompresimit siç duket edhe nga tabela e mësiperme është zbatuar vetëm për faqet zero.

Sikurse shikohet nga tabelat e mësipërme me krijimin e dy bashkësive të punës veç për faqet zero dhe veç për ato të modifikuara, duke ndjekur rregullat e arta të Listave të Lidhura zinxhir do të shikohet një degradim i proçesorit i cili shoqërohet me një përmirësim të kohës totale të migrimit dhe downtime. Kjo për arsyen se gjetja e një bashkësie pune të vogël ul penalitetet shtesë të kërkimit, si rrjedhojë edhe migrimi do të jetë më i vogël në kohë. Ajo që nuk ndikohet është trafiku i krijuar për arsye se, numri i faqeve të lira dhe të modifikuara nuk ndryshon, ajo që ndryshon është vetëm organizimi i tyre.

Ajo që vihet re më shumë interes në këtë tabelë është ulja e faqeve të humbura. Kjo vjen si pasoj e përcaktimit të bashkësise së punës me të vogël. Parashikimi i sukseshëm i bashkësisë së punës do të thotë se do të migrohen vetëm ato faqe që e kanë të nevojshme.

6.2.6 Përmirësime të algoritmit duke përdorur metodën e parashikueshmërise së bashkesisë së punës e kombinuar me algoritmat LRU dhe Splay Tree.

Nëse i referohemi artikujve [21] dhe [23] shikohet se me rritjen e numrit të faqeve të modifikuara, kjo metodë nuk është efiçente. Për këtë mundohemi të testojmë rastin kur në vend të 5000 skripte të cilët gjenerojnë grafike dhe numra të rastesishëm, do të ngremë 10000 prej tyre. Kjo do të rrisë numrin e shkrimeve në memorjen fizike dhe do të shoqërohet me rritje të faqeve të modifikueshme. Do të shikojmë si do të ndryshojnë parametrat e tabelave të mësiperme:

Tabela 6.13-Ngritja e teknikës në një rrjet LAN duke caktuar bashkësinë minimale të punës me ane të algoritmit LRU me dyfishimin e faqeve në memorje

Me Parashikim (LRU+Splay Tree) - ME LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Humbura.
XEN -PV	65-96	27,4 - 27,8	15,6 17,6	240 272	0,007 0,014	8 10
XEN -FV	65-96	36,6 – 36,9	20,2 21,9	435 480	0,016 0,028	20 21

KVM - PV	65-96	32,5 – 32,9	18,9 21,9	466 483	0,007 0,015	11 12
KVM - FV	65-96	41,4 – 41,7	26,4 29,1	686 710	0,021 0,035	25 27
OPENVZ	65-96	26,4 – 26,6	16,1 18,3	301 360	0,011 0,020	8 9

Me Parashikim (LRU+Splay Tree) - PA LAJMËRIM – LAN –iSCSI with TOE, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim ne ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të humbura
XEN -PV	65-96	240 272	9231 9455	0,007 0,014	2,6 2,7	15 17
XEN -FV	65-96	435 480	11699 11722	0,016 0,028	3,0 3,1	27 29
KVM - PV	65-96	466 483	11115 11225	0,007 0,015	2,6 2,7	18 21
KVM -	65-96	686	13624	0,021	2,9	33

FV		**710**	**14310**	**0,035**	**3,0**	**36**
OPENVZ	**65-96**	**301** **360**	**8645** **8790**	**0,011** **0,020**	**2,5** **2,6**	**18** **20**

Siç shikohet ne tabelë megjithëse u dyfishua numri i skripteve, numri i proçeseve të lindura u rrit vetëm me gjysmën e tyre. Kjo vjen për shkak se një pjesë e proçeseve ekzekutojnë skripte të të njejtës natyrë dhe mund ta menaxhojnë këtë skript duke shtuar numrin e threadeve të tyre.

Ajo që dallohet është një rritje e lehtë e parametrave matës. Nëse i referohemi kohës totale të migrimit ajo rritet me shtimin e numrit të faqeve të modifikuara, por jo linearisht me to. Ndërkohë që e njëjta gjë mund të thuhet për të gjithë parametrat e tjerë të tabelës.

Ndërkohë që për tu nënvizuar është se degjenerimin më të madh e ndjen Hypervisori KVM në nivel të virtualizimit të plotë. Kjo përputhet plotësisht me artikuj të ndryshem si [35],[42],[43],[44].

6.2.7 Përmirësime të algoritmit duke përdorur metodat e mësiperme mbi arkitekturën RDMA

Do të kërkojmë një ndryshim rrënjësor të arkitekturës së komunikimit midis hosteve në LAN, duke ruajtur të njejtën Topologji por tani duke përdorur një metodë të quajtur RDMA (Remote Direct Memory Access) [21], [22],[23],[29] dhe [32]. Referuar artikujve të cituar, kjo teknikë rrit së tepërmi efiçensën e migrimit dhe përkatësisht në: Kohën Totale të Migrimit, Trafikun dhe Kohën e Bllokimit. Për të implementuar teknikën RDMA duhet të ndërtoj një ndërfaqe komunikimi me hardware Infiniband dhe protokoll iWARP (Internet Wide Area RDMA Protocol). Një pamje e nderfaqes së Infiniband jepet në figurën e mëposhtëme:

Figura 6.13- Ndërfaqja e komunikimit Infiniband 4x

Në fakt parametrat që janë përdorur për Infiniband janë Enhanced Data Rate me shpejtësi 4x, duke arritur kapacitetin 100 Gbps. Bazuar në [33] mund të permend se teknika RDMA ofron një shpejtësi shumë të lartë në transferimin e memorjes nga një makinë fizike në tjetrën pa përfshirjen e CPU-së dhe Sistemit Operativ. Bazuar në artikullin [22] mos perfshirja e kompleksitetit të Sistemit Operativ çon në eliminimin e dy kopjeve në kernel dhe zevendësimin e tyre me funksionet *map* dhe *umap* drejt memorjes përdorues. Kjo gjë bën që implementimi i protokollit RDMA të lejoj dërgimin e një file nga një memorje në tjetrën. Ajo që humbasim këtu është shfrytëzimi i Memorjes si pasojë e shumfishimit të kopjeve në të. RDMA ndryshe mund te thuhet se suporton rrjetin *zero-copy* [34] për shkak të fuqisë që i jepet kartës së rrjetit për të transferuar të dhëna direkt drejt një aplikacioni pa i kopjuar me parë ato në kernel. Të gjitha teknologjitë e bazuara mbi teknikën *Infiniband* kanë si karakteristikë RDMA. RDMA bën që të shmangim teknikën klasike të komunikimit me TCP/IP. Futja e kësaj teknike kërkon vetëm një instalim rutine në Sistemin Operativ Linux CentOS 5,6 që kam përdorur në punimin tim. Ndërkohë ajo që duhet të instaloj është protokolli iWARP. Pas ketyre modifikimeve do të përsëris eksperimentin me 5000 skripte dhe do të shikoj të gjithe parametrat përkates:

Tabela 6.14-Ngritja e teknikës në një rrjet LAN duke caktuar bashkësinë minimale të punës me ane të algoritmit LRU mbi metodën RDMA.

PA / ME RDMA - ME LAJMËRIM – LAN –iSCSI-RDMA with Infiniband - iWARP, SYN.

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Humbura.
XEN -PV	65-0	27,4 - 15,7	15,6 9,4	240 205	0,007 0,003	8 4
XEN -FV	65-0	36,6 – 26,4	20,2 14,5	435 402	0,016 0,012	20 11
KVM - PV	65-0	32,5 – 20,5	18,9 12,3	466 430	0,007 0,003	11 6
KVM - FV	65-0	41,4 – 30,9	26,4 21,1	686 654	0,021 0,018	25 14
OPENVZ	65-0	26,4 – 13,3	16,1 9,7	301 268	0,011 0,006	8 3

PA / ME RDMA - PA LAJMËRIM – LAN –iSCSI-RDMA with Infiniband - iWARP, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim	Downtime pa lajmërim	Overhead me lajmërim	Overhead pa lajmërim	Nr i faqeve të

		ms	ne ms	%	%	humbura
XEN -PV	**65-0**	**240** **205**	**9231** **8591**	**0,007** **0,003**	**2,6** **2,1**	**15** **10**
XEN -FV	**65-0**	**435** **402**	**11699** **11052**	**0,016** **0,012**	**3,0** **2,6**	**27** **22**
KVM - PV	**65-0**	**466** **430**	**11115** **10569**	**0,007** **0,003**	**2,6** **2,2**	**18** **14**
KVM - FV	**65-0**	**686** **654**	**13624** **10002**	**0,021** **0,018**	**2,9** **2,6**	**33** **30**
OPENVZ	**65-0**	**301** **268**	**8645** **7968**	**0,011** **0,006**	**2,5** **2,1**	**18** **12**

Rezultatet që përfitojmë janë shumë impresionuese. Kështu falë teknikës RDMA dhe implementimit hardware mbi Infiniband arrijmë të përfitojmë një përmisim në të gjithë faktorët përkatës të migrimit. Ndërkohë, ajo që vihet re më konkretisht është reduktimi i numrit të faqeve të humbura gjatë migrimit pa lajmërim, duke implementuar metodën *Cold Migration*.

Shënim: Në artikullin [22] është futur dhe koncepti i migrimit *Pipeline* por që në eksperimentet e mia nuk është i nevojshëm pasi aplikacionet zënë një hapsirë që nuk kalon madhesinë fizike të memorjes.

Duke ju referuar artikullit [32], vihet re përfshirja e një metode e quajtur *warm up*. Me anë të kësaj metode bëhet e mundur transferimi i memorjes në avancë. Kjo fazë

ekzekutohet në background pa shkaktuar ndërprerje të shërbimit. Kjo mënyrë siç shprehet dhe në artikullin [32] do të ofrojë lehtësira në migrimin *High Availability*. Kjo metodë në fakt është efiçente në aplikacione të mëdha, gjithsesi do të provoj efiçensen e saj në rastin kur madhësia totale e faqeve nuk zë hapsirë të madhe.

6.2.7 Përmirësime të algoritmit duke përdorur arkitekturen RDMA me metodën Warm-Up

Bëjmë modifikim në kernel përkatësisht për KVM – FV duke modifikuar disa komanda në QEMU si në [32]. I njejti modifikim bëhet edhe për XEN-FV. Ndërkohë që për XEN-PV , KVM –PV dhe OpenVZ duhet të modifikojmë libraritë në të. Duke ju referuar serish [32] krijojmë një algoritëm shtesë në kernel që do të ruajë skedarët e referencës, do të përfshijë metodën *copy on write*, hijet ose "*shadowing"*. Kështu faqet e fundit të modifikuara do të ruhen në një cache të quajtur *Adaptive Replacement Cache* brenda memorjes së Hypervisorëve. Kjo cache është një zonë e rezervuar në memorje ku ruhen versionet e faqeve të referencës. Algoritmi është marrë nga [27] dhe është si më poshtë:

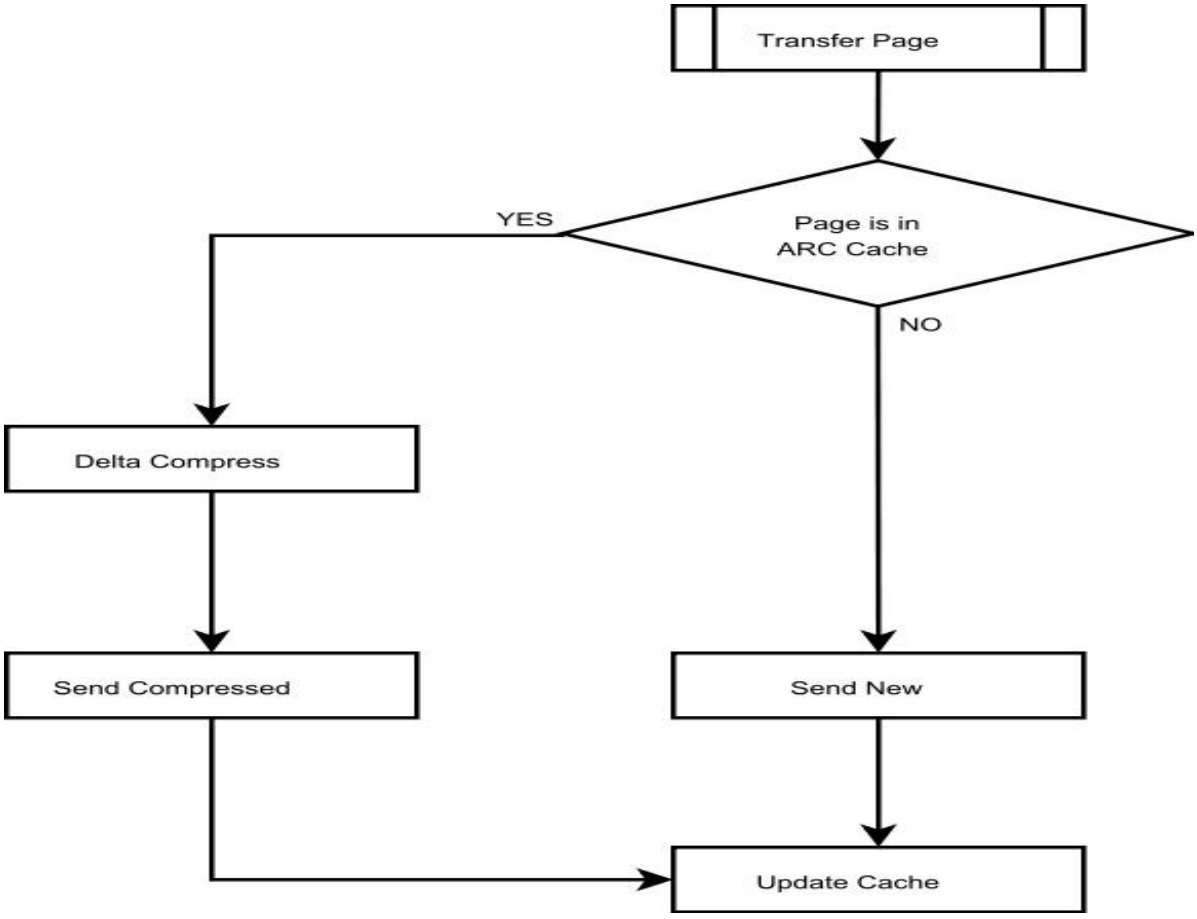


Figura 6.14-Skica e implementimit të algoritmit të Cache-së ARC (Adaptive Replacement Cache) në kernel

Në bazë të ketij algoritmi do të marrim këto vlera nga tabelat e mëposhtëme:

Tabela 6.15-Ngritja e teknikës në një rrjet LAN duke caktuar bashkësinë minimale të punës me anë të algoritmit LRU mbi metodën RDMA dhe Warm-UP (Para-Ngrohjes)

ME Warm-UP / PA Warm-UP - ME LAJMËRIM – LAN –iSCSI-RDMA with Infiniband - iWARP, SYN.

Lloji i	Nr i	CPU	Koha e	Downtime	Over-	Nr i

HV	proçeseve	Shfrytz.%	migrimit (s)	ms	head %	faqeve të Humbura.
XEN -PV	65-0	25,4 - 15,7	8,6 9,4	200 205	0,003 0,003	4 4
XEN -FV	65-0	36,2 – 26,4	12,6 14,5	399 402	0,011 0,012	11 11
KVM - PV	65-0	31,1 – 20,5	10,1 12,3	430 430	0,003 0,003	5 6
KVM - FV	65-0	40,8 – 30,9	19,5 21,1	650 654	0,017 0,018	13 14
OPENVZ	65-0	22,8 – 13,3	8,4 9,7	261 268	0,005 0,006	3 3

ME Warm-UP / PA Warm-UP - PA LAJMËRIM – LAN –iSCSI-RDMA with Infiniband - iWARP, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim ne ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të humbura
XEN -PV	65-0	200	8580	0,003	2,1	10

		205	8591	0,003	2,1	10
XEN -FV	65-0	399 402	11044 11052	0,011 0,012	2,6 2,6	21 22
KVM - PV	65-0	430 430	10498 10569	0,003 0,003	2,2 2,2	13 14
KVM - FV	65-0	650 654	9994 10002	0,017 0,018	2,6 2,6	30 30
OPENVZ	65-0	261 268	7960 7968	0,005 0,006	2,1 2,1	11 12

Siç shikohet nga tabelat e mësipërme shikohet që CPU-ja ka një rritje të ndjeshme, për arsye se gjatë gjithë kohës proçesori duhet të organizojë faqet e memorjes dhe duhet ti migrojë ato drejt një makine destinacion pa ardhur komanda e migrimit të tyre. Por ajo që përfitojmë është në kohën totale të migrimit dhe deri diku edhe në kohën e bllokimit. Por nuk përfitojmë asgjë nga trafiku, dmth nuk na reduktohet ai në momentin e migrimit. Kjo gjë për arsye se numri i faqeve të modifikuara i referohet momentit të iterimit të fundit dhe nuk ka lidhje me fazën *warm-up*.

Një përmirësim i artikullit [39] është bërë nga unë në dy drejtime:

1.Implementimin e teknikave të migrimit në Hypervisor të ndryshëm

2. Implementimin e teknikës edhe në rastet e pa-lajmëruara.

Referuar [39], do të bëhet e mundur instalimi jo më i një makine mbi Hypervisor, por i dy të tilla. Kjo do të bëjë që të migrojmë në rast dështimi të një nyje jo një makine virtuale “Guest” por dy njëherësh. Për këtë situatë, në kernel është ngritur një algoritëm i

cili nuk është gjë tjetër veçse një tabelë hash e cila eliminon kopjimet e faqeve të njëjta. Faqet e njëjta vendosen në një zonë të përbashkët të memorjes dhe marrin statusin si faqe te përbashkëta. Këto faqe mund të aksesohen nga makina të ndryshme. Faqet personale të çdo makine virtuale i takojnë adresave specifike të secilës. Tabela Hash do të ketë gjithashtu edhe vetinë e metodës *Delta*, që do të thotë se në faqet e modifikuara dërgohen vetëm pjesa e modifikuar, duke bërë diferencën midis gjithë faqes dhe asaj që është modifikuar së fundmi. Duke qenë se do të bëhet një kontroll në tabelën e faqeve në kernel do të ishte më kollaj që këtë algoritëm të ngrihej në kernel. Të gjithë Hypervisorët do të kishin të njejtën sjellje ndaj këtij algoritmi, pasi pjesa e kontrollit të faqeve të memorjes dhe konkretisht e biteve P dhe M nuk ndryshon shumë në Hypervisor të ndryshëm. Algoritmi është i ngjashëm me algoritmin tonë në hapsirën përdorues, por veçse tani ky algoritem i përket vetëm rasteve të makinave paralele të migruara brenda një hosti të caktuar. Nëse këtë algoritëm do ta zgjeronim në algoritmin tonë kryesore, pra në hapsirën përdorues, do të ishte shumë e dobishme pasi do të ruhej abstraksioni, por e meta do të ishte se algoritmi yne do të bëhej shumë kompleks dhe veshtirësia për ta mirëmbajtur do të rritej ndjeshëm. Kjo ishte arsyeja pse vendosa të krijoj një algoritem tjetër tani në kernel për të bërë të mundur menaxhimin dhe efiçensën e memorjes së makinave virtuale gjatë migrimit të tyre.

Fillimisht mund të shkruajë se si GuestOS të dytë kam përdorur një makinë me Ubuntu 10.10 Desktop. Kështu do të kemi dy GuestOS njëra me CentOS5,6 dhe tjetra me Ubuntu 10.10 Desktop si ne figure:

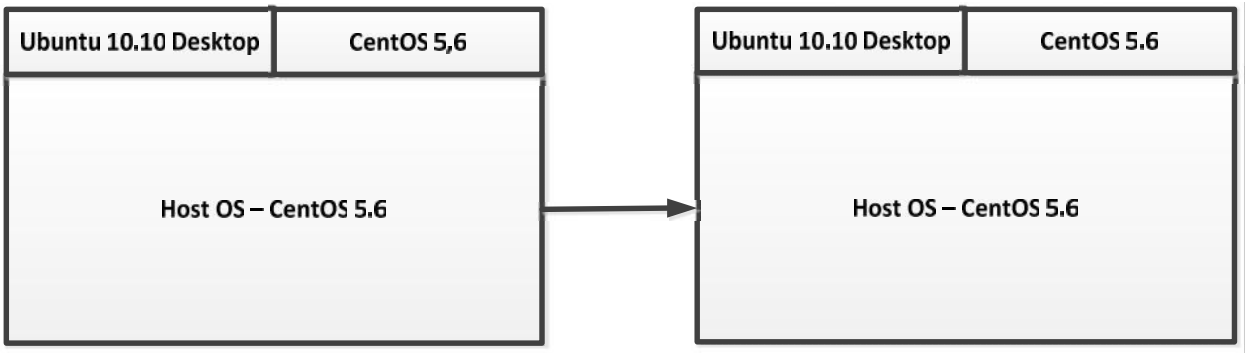


Figura 6.15-Koncepti i komunikimit Gang Live midis dy hosteve.

Të njejta skripte i ndërtojmë edhe në makinën e dytë virtuale, duke kërkuar që të dy-fishojmë në 130 numrin e proçeseve. Këto proçese do të jenë të ndara në makinat respektive “GuestOS”.

Algorimin në kernel po e shfaq në mënyre skematike sipas figurës më poshtë:

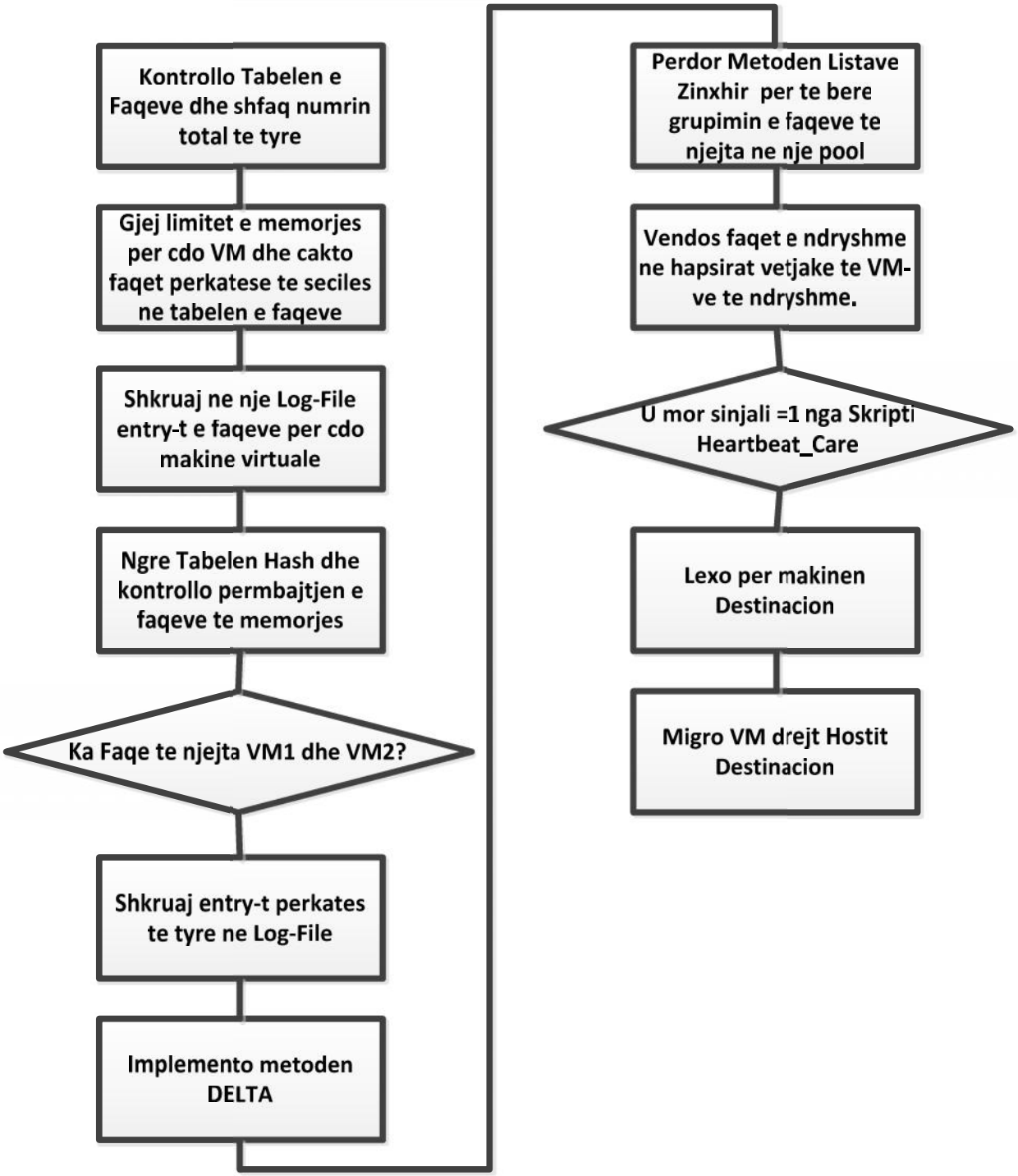

Kontrollo Tabelen e Faqeve dhe shfaq numrin total te tyre
Gjej limitet e memorjes per cdo VM dhe cakto faqet perkatese te seciles ne tabelen e faqeve
Shkruaj ne nje Log-File entry-t e faqeve per cdo makine virtuale
Ngre Tabelen Hash dhe kontrollo permbajtjen e faqeve te memorjes
Ka Faqe te njejta VM1 dhe VM2?
Shkruaj entry-t perkates te tyre ne Log-File
Implemento metoden DELTA
Perdor Metoden Listave Zinxhir per te bere grupimin e faqeve te njejta ne nje pool
Vendos faqet e ndryshme ne hapsirat vetjake te VM-ve te ndryshme.
U mor sinjali =1 nga Skripti Heartbeat_Care
Lexo per makinen Destinacion
Migro VM drejt Hostit Destinacion

Figura 6.16-Agoritmi mbi implementimin e teknikës Gang Live midis makinave fizike me 2 VM në to.

6.2.8 Simulime paralele në Makinat Virtuale

Eksperimentin do ta bëjme pak më ndryshe. Do të provojmë migrimin e makinave virtuale pa përdorur algoritmin në kernel që shkurtimisht po themi *Pa Live Gang* dhe më pas do të implementojmë këtë metodë në migrimin e njëhershëm të dy makinave virtuale duke përdorur këtë metodë.

Pas realizimit të kësaj metode do të marrim këto vlera:

Tabela 6.16 Ngritja e teknikës në një rrjet LAN duke caktuar bashkësinë minimale të punës me anë të algoritmit LRU mbi metodën RDMA dhe Warm-UP (Para-Ngrohjes mbi Live Gang.

PA LIVE GANG / ME LIVE GANG - ME LAJMËRIM – LAN –iSCSI-RDMA with Infiniband - iWARP, SYN.

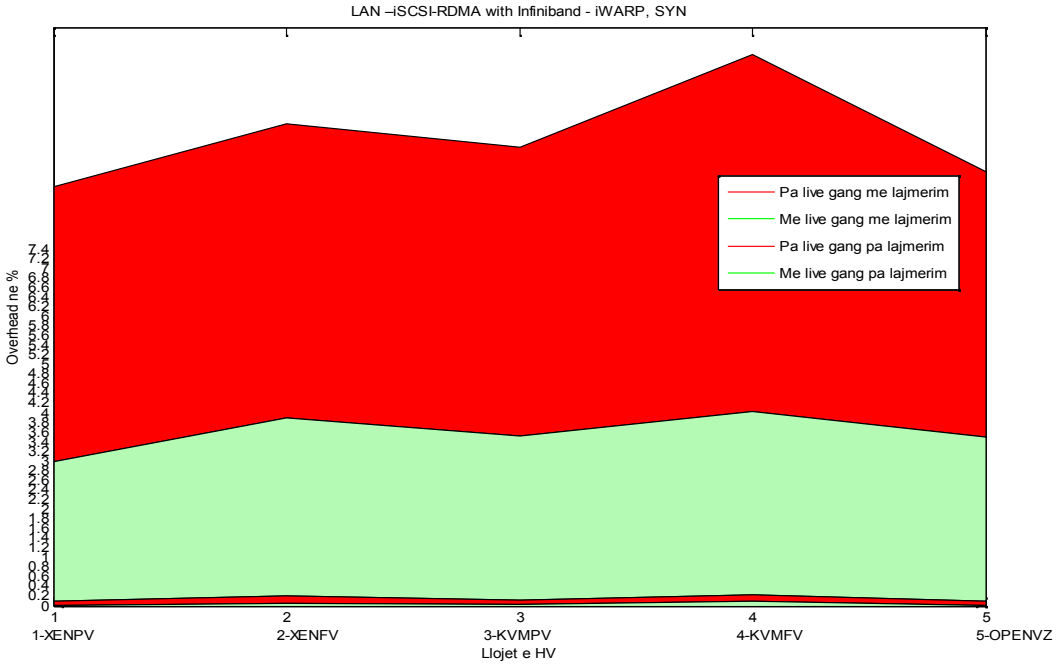
Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Humbura.
XEN -PV	130	45,8 - 37,2	16,9 8,9	590 290	0,07 0,02	12 6
XEN -FV	130	48,5 – 39,7	28,8 15,4	887 402	0,14 0,06	44 25
KVM -	130	54,9 –	21,1	995	0,08	15

PV		**44,8**	**12,8**	**430**	**0,04**	**7**
KVM - FV	**130**	**63,2 – 54,8**	**40,2 22,3**	**1456 654**	**0,15 0,09**	**31 20**
OPENVZ	**130**	**41,5 – 33,6**	**15,8 10,6**	**505 268**	**0,08 0,02**	**11 6**

PA LIVE GANG / ME LIVE GANG - PA LAJMËRIM – LAN –iSCSI-RDMA with Infiniband - iWARP, SYN.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim në ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të humbura
XEN -PV	**130**	**590 290**	**36924 10566**	**0,07 0,02**	**5,7 2,9**	**37 22**
XEN -FV	**130**	**887 402**	**48562 14365**	**0,14 0,06**	**6,1 3,7**	**92 45**
KVM - PV	**130**	**995 430**	**46225 13214**	**0,08 0,04**	**6,0 3,4**	**97 31**
KVM - FV	**130**	**1456 654**	**42868 13101**	**0,15 0,09**	**7,4 3,8**	**110 71**

OPENVZ	**130**	**505**	**268**	**34998** **9685**	**0,08**	**0,02**	**5,5** **3,4**	**44** **26**



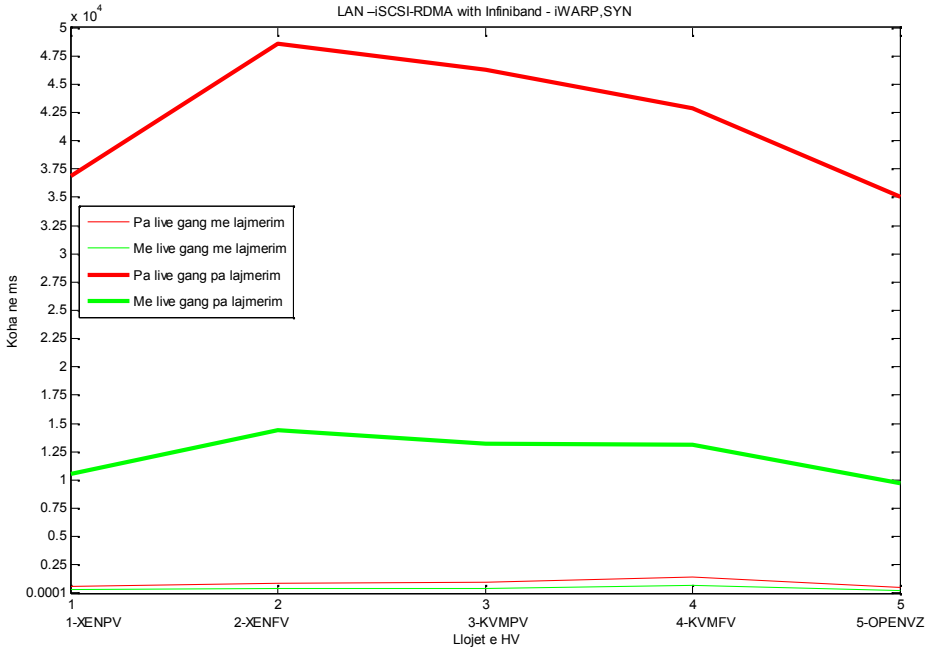


Figura 6.17- Grafiku për shfaqjen e Teknikës Paralele Me/Pa Live Gang në metodat Me/Pa Lajmërim

Duhet të shënoj se të gjitha metodat e shkruajtura deri më tani janë përdorur për të dy rastet e eksperimenteve.

Siç duket dhe nga tabelat e mësiperme, duke shtuar numrin e makinave virtuale do të shtohen koefiçentët e parametrave të migrimit. Ajo që vihet re është përmirësimi i ndjeshëm që përfitohet duke përdorur metodën Live Gang. Me anë të kësaj metode parametrat i ofrohen shumë rastit të migrimit me një makinë të vetme virtuale. Kjo është falë ngritjes së dy parametrave në ketë metode:

a.Tabelës hash për faqet e njejta

b.Implementimit të metodës Delta.

Duke përdorur këtë metodë shikohet një përmisim edhe në humbjen e faqeve të modifikuara.

6.2.9 Simulime paralele me Makinat Virtuale të impelentuara në Databasen Oracle 11g.

Metoda e fundit e implementuar i përket ngritjes së migrimit tashmë të dy makinave virtuale vetëm mbi Hypervisorin OracleVM. Referuar [35]. Për këtë si database është implementuar ambjenti Oracle10g. Duke përdorur metodat e tabelave të transportueshme në Platforma të Kryqëzuara dhe duke ngritur një aplikacion online Faturimi të quajtur *Billing* i cili do të rregjistrojë faturat për 30000 abonentë të gjeneruar rastësisht në bazë emrash dhe mbiemrash nga një skript i shkruar në ***C#*** të quajtur *BillingC*. Emrat do të plotësojnë një formë dhe do të siglojnë disa imazhe mbi këtë formë.

Testimet janë bërë për dy PC të lidhur në një rrjet me një Switch Gigabitësh me një SAN Storage si ai i përdorur në eksperimentet e deri tanishme. Nga testimet do të kemi keto rezultate:

PA VM-Oracle / ME VM-Oracle - ME LAJMËRIM/PA LAJMËRIM – LAN – iSCSI-RDMA with Infiniband.

Tabela 6.17-Ngritja e teknikës në një rrjet LAN me metodat e tabelave të transportueshme në Platforma të Kryqëzuara në mjedisin Oracle.

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Mbetura.
OracleVM	130	32,4 - 35,6	16,2 3,4	380 175	0,06 0,01	11 5
OracleVM	130	34,3 - 37,3	154 76	32620 5146	0,06 0,01	72 25

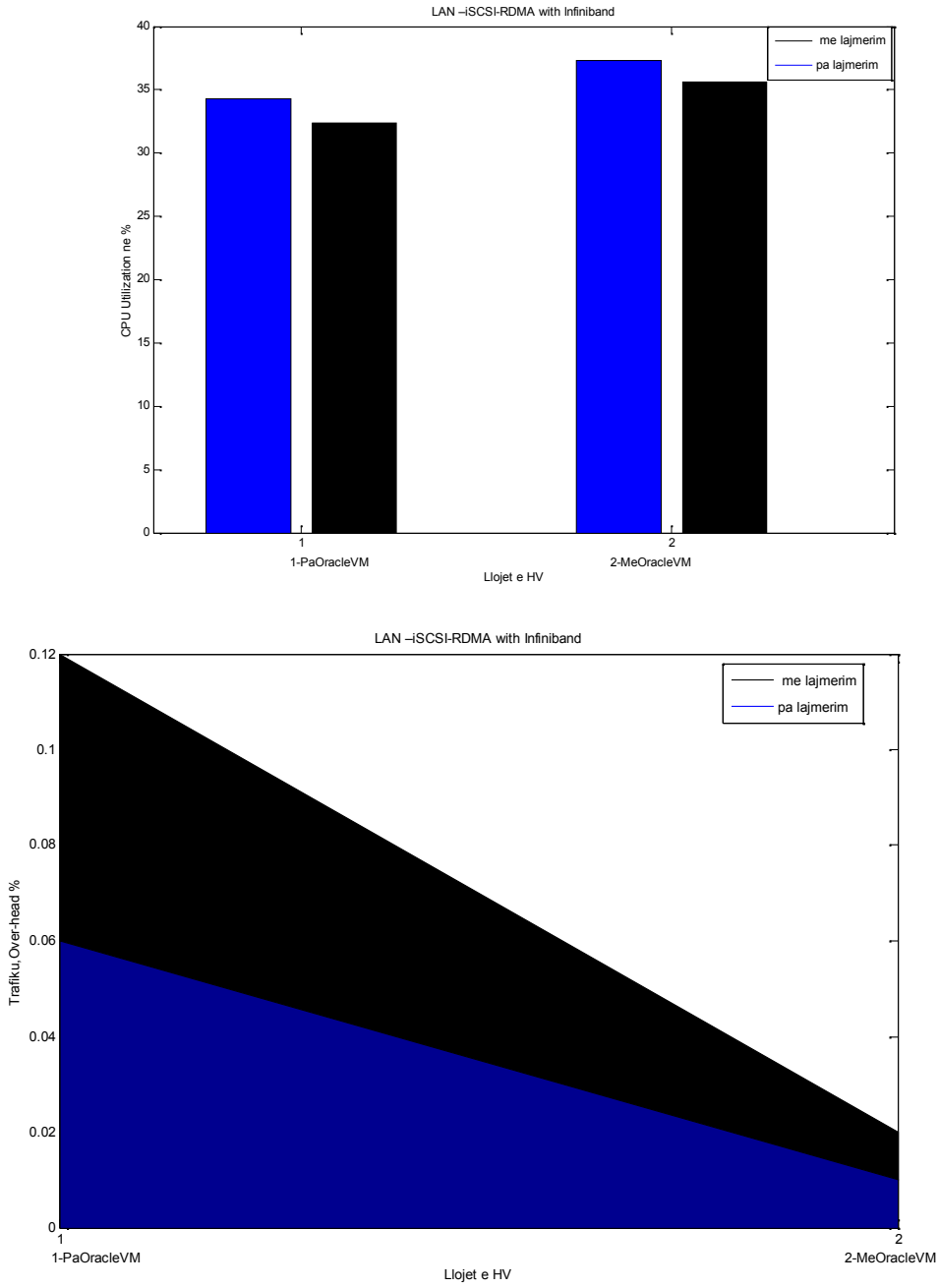


Figura 6.18-Teknika RDMA në sistemet *ORACLE VM*, ME/PA Lajmërim

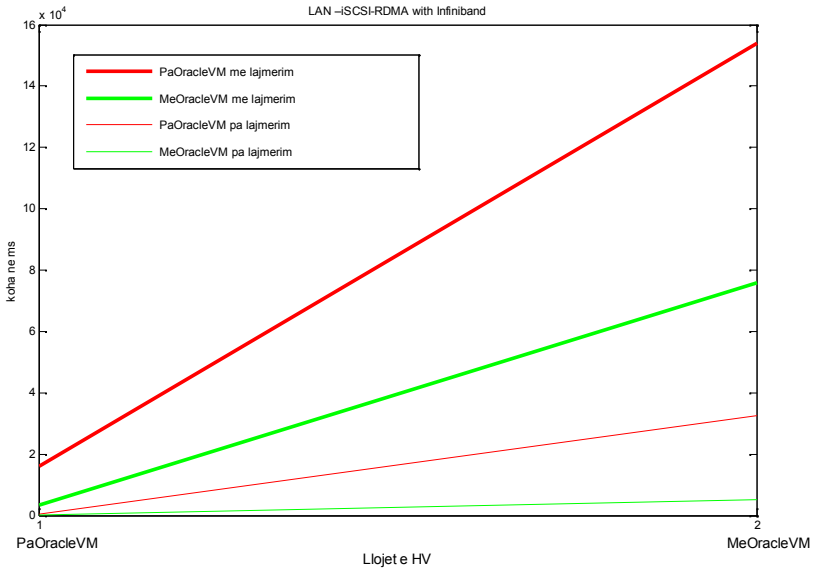


Figura 6.19-Teknikat Me/Pa Oracle VM në teknikat Me/Pa Lajmërim në rrjetat LAN.

Mund të përmendim që makina virtuale OracleVM është me të njejtën strkturë si ato me XEN [45]. Pra siç shikohet dhe nga tabela e mësipërme përmirësimi i dukshëm vjen nga platforma në Oracle. Kjo vjen për vetë strukturën e këtij database. Gjithashtu mund të deklaroj se ngritja e makinave virtuale Oracle Virtual Machine me platforma të kryqëzuara [35] bën të mundur uljen e kohës totale të migrimit, kohës së bllokimit dhe ul numrin e faqeve të humbura.

6.2.10 Simulime paralele në Makinat Virtuale në rrjetin e gjerë WAN

Nëse dy simulimet e fundit i implementojmë në një rrjet WAN të lidhur përmes kanaleve fiber. Do të tentoj të bëj migrimin Live Gang dmth, të disa makinave virtuale nga një host drejt një hosti tjetër. Për këtë do të ngremë dy routera të tipit CISCO 7301 me ndërfaqe Fibër dhe Switch WS-C3750x-48T-L me të njejtën ndërfaqe në mënyre që të mos kemi humbje në kohë dhe në trafik. Lidhja në WAN do të aktivizohet për 21 km distancë.

Figura 6.20-Pamja e pajisjeve router të tipit CISCO 7301 me ndërfaqe Fiber dhe Switch WS-C3750x-48T-L

Skica përkatëse bën të mundur komunikimin në një rrjet WAN midis makinave përkatëse:

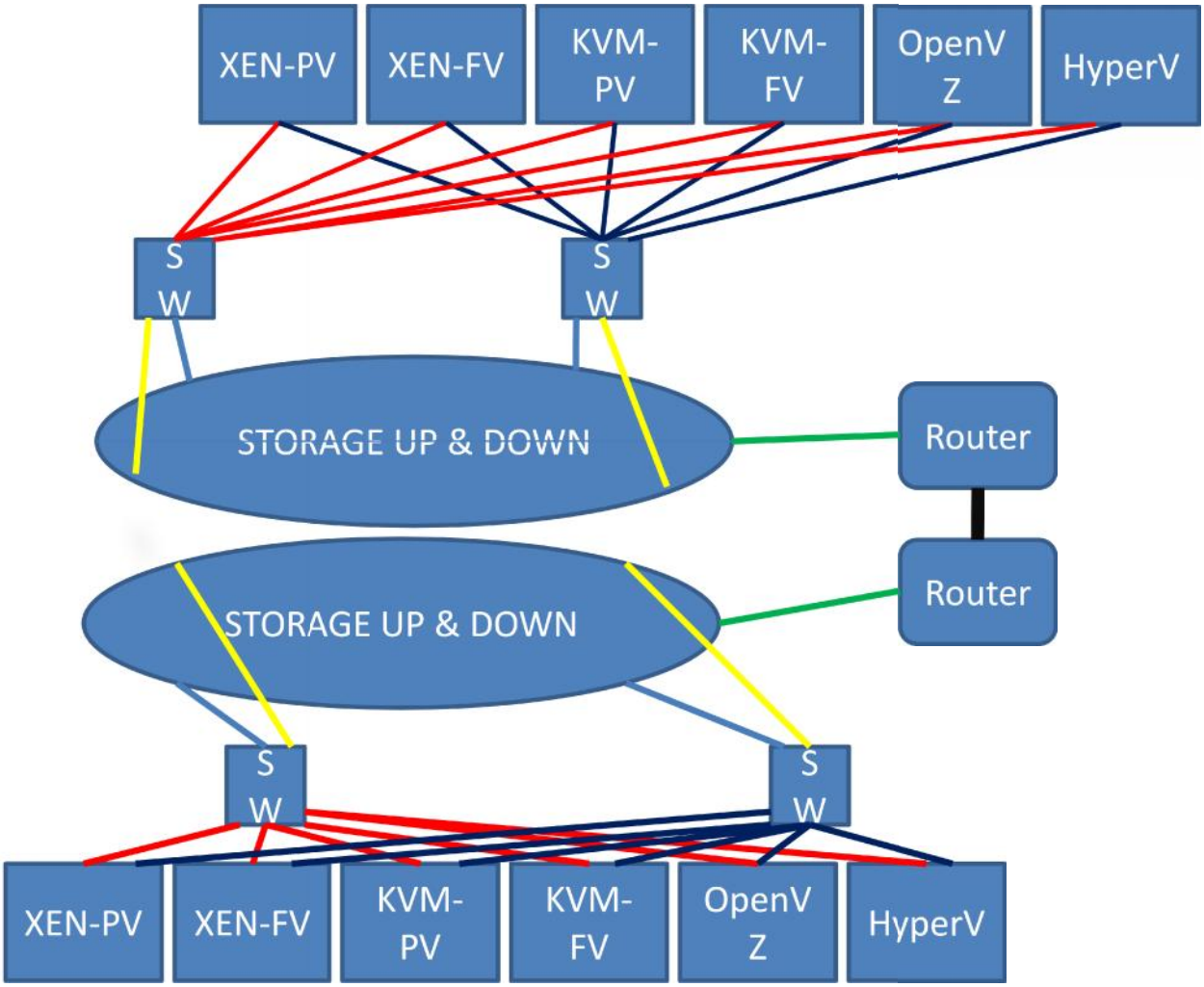


Figura 6.21: Skica që bën të mundur komunikimin në një rrjet WAN midis makinave përkatëse

PA LIVE GANG / ME LIVE GANG - ME LAJMËRIM – WAN –Fibre Channel, SYN.

Tabela 6.18. Krahasimet në metodat Pa Live Gang dhe Me Live Gang në metoden me Lajmerim në rrjetat WAN me Kanale Fiber.

Lloji i	Nr i	CPU	Koha e migrimit	Downtime	Over-	Nr i faqeve të

HV	proçeseve	Shfrytz.%	(s)	ms	head %	Humbura.
XEN -PV	130	46,9 - 38,7	17,5 9,5	890 560	0,08 0,03	13 7
XEN -FV	130	49,4 – 41,2	30,5 16,5	1165 705	0,16 0,08	45 26
KVM - PV	130	55,6 – 44,8	22,0 13,7	1255 740	0,09 0,05	16 8
KVM - FV	130	64,5 – 55,7	42,1 24,1	1744 961	0,16 0,10	32 21
OPENVZ	130	42,6 – 34,7	16,7 12,1	822 544	0,09 0,03	12 7

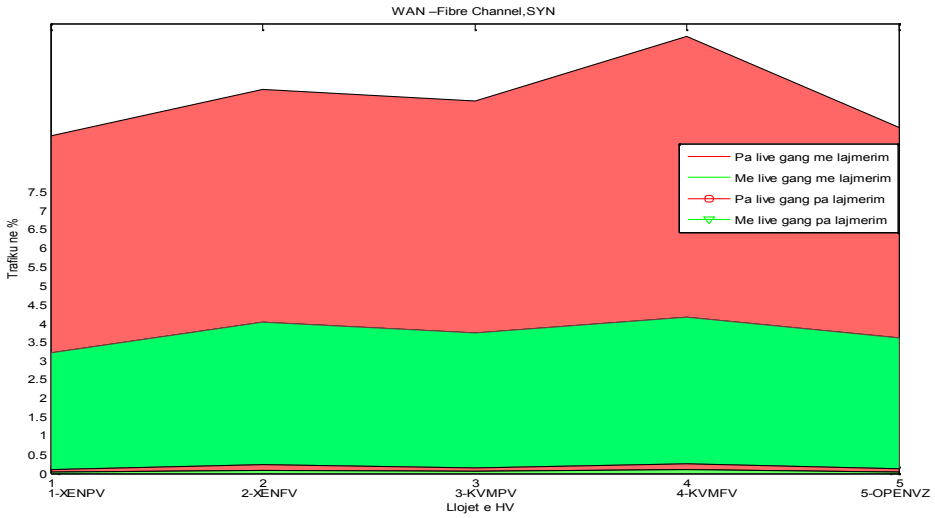


Figura 6.22-Teknikat Me/Pa Oracle VM në teknikat Me/Pa Lajmërim në rrjetat WAN.

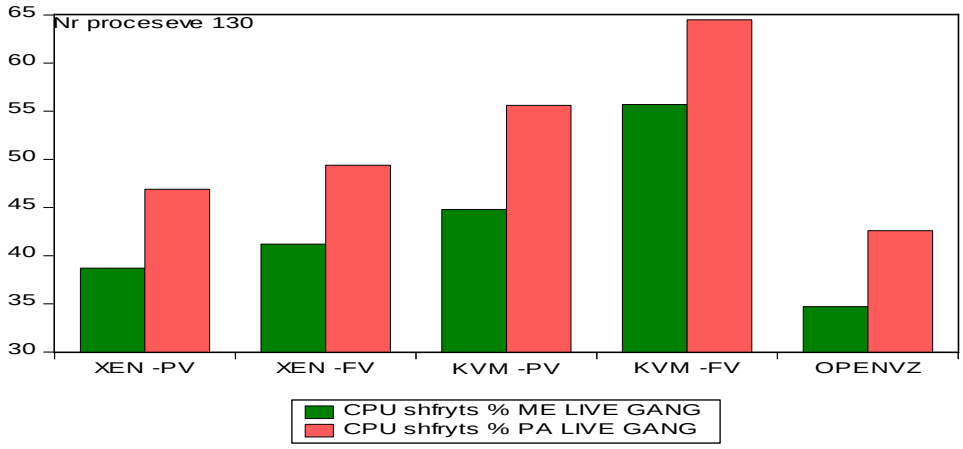


Figura 6.23- Shfrytëzimi në % i CPU-së Me/Pa Live Gang

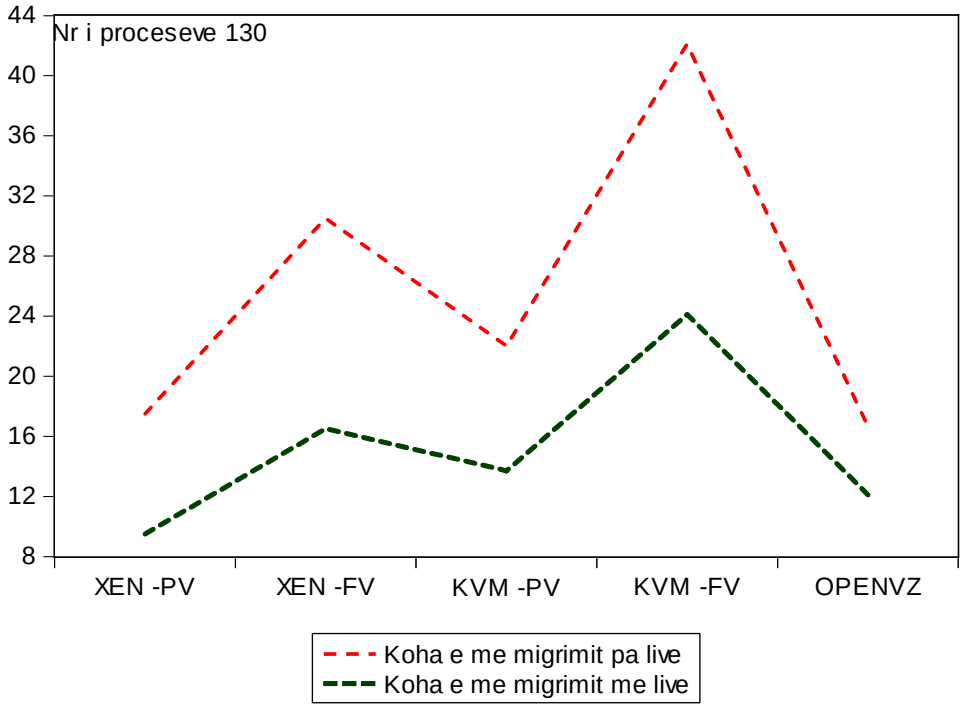


Figura 6.24- Koha e Migrimit Me/Pa Metodën Live Gang për 5 Hypervisor të ndryshëm

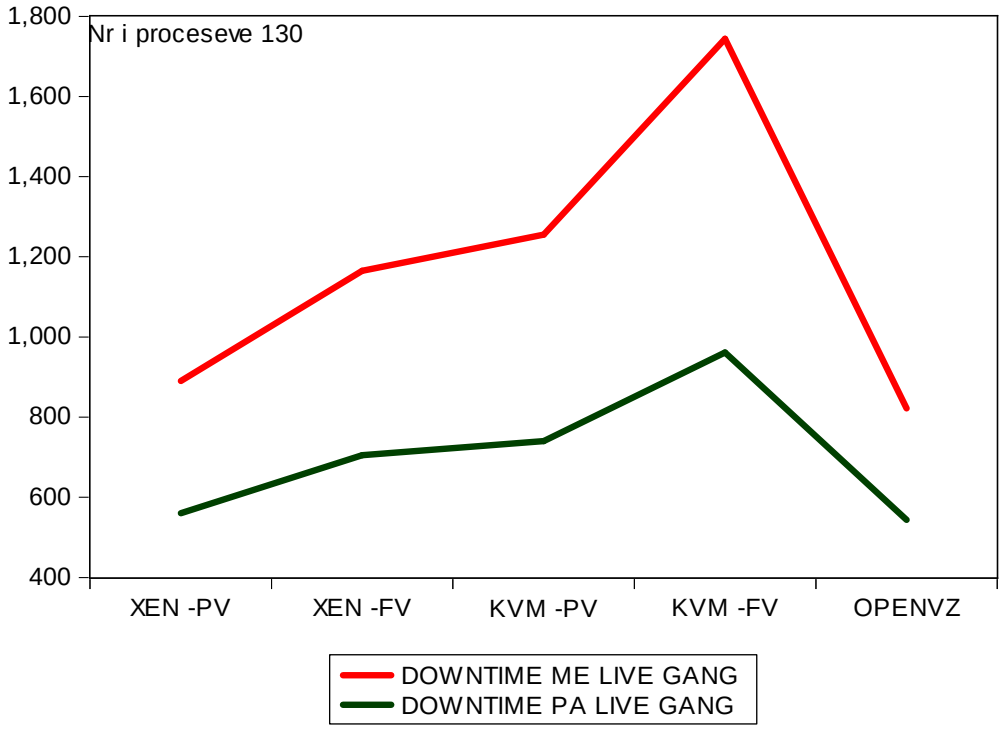


Figura 6.25 - Downtime Me/Pa Metodën Live Gang për 5 Hypervisor të ndryshëm

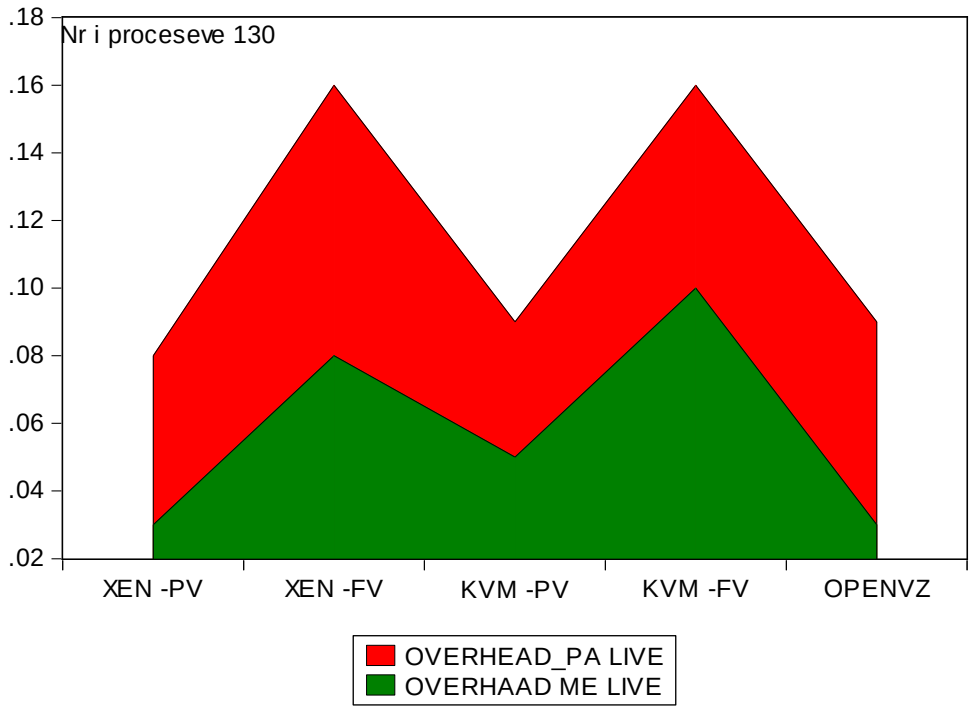


Figura 6.26 - Overhead Me/Pa Metodën Live Gang për 5 Hypervisor të ndryshëm

PA LIVE GANG / ME LIVE GANG - PA LAJMËRIM – WAN –Fibre Channel, SYN.

Tabela 6.19-Ngritja e teknikës në një rrjet WAN duke caktuar bashkesinë minimale të punës me anë të algoritmit LRU me kanalet fibër mbi metodën Live Gang.

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim në ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të humbura
XEN -PV	130	890 560	42524 13245	0,08 0,03	5,8 3,1	39 24

XEN -FV	**130**	**1165** **705**	**52644** **17581**	**0,16** **0,08**	**6,2** **3,8**	**94** **47**
KVM - PV	**130**	**1255** **740**	**50124** **16247**	**0,09** **0,05**	**6,2** **3,6**	**101** **34**
KVM - FV	**130**	**1744** **961**	**46954** **16247**	**0,16** **0,10**	**7,5** **3,9**	**113** **74**
OPENVZ	**130**	**890** **560**	**38864** **12547**	**0,09** **0,03**	**5,6** **3,5**	**45** **27**

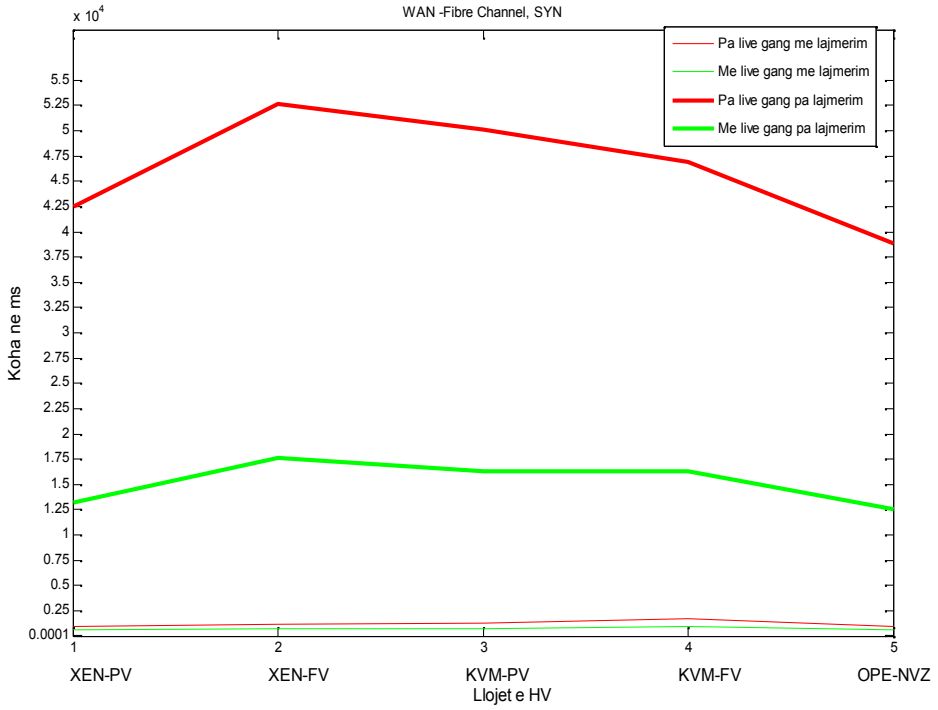


Figura 6.27: Teknikat Pa/Me Live Migration në një rrjet WAN dhe matja e kohës së migrimit duke u implementuar në këtë teknikë

Nga tabelat e mësipërme vëmë re se implementimi i migrimit Live Gang midis makinave fizike në rrjetin WAN, ndikon shumë pak në konsumimin e Proçesorit, por mund të rris mjaft numrin e kohës së bllokimit, numrin e faqeve të humbura. Ndërkohë që trafiku nuk ndikon për arsye se vetë routerat dhe kanali me fibër nuk shton elementë shtesë enkriptimi në kanal, pasi trafiku nuk është i enkriptuar. Ndërkohë që koha totale e migrimit shtohet për shkak të kohës shtesë së futur nga buferat e ruterave.

6.2.10 Simulime paralele në Makinat Virtuale në rrjetin e gjerë WAN mbi Database Oracle 11g

Nëse e bëjmë provën duke implementuar mbi hypervisorin OracleVM do të marrim këtë tabelë

PA VM-Oracle / ME VM-Oracle - ME LAJMËRIM/PA LAJMËRIM – WAN – Fibre Channel, SYN.

Tabela 6.20 Ngritja e teknikës në një rrjet WAN me metodat e tabelave të transportueshme në Platforma të Kryqëzuara në mjedisin Oracle.

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Mbetura.
OracleVM	130	35,4 - 39,4	19,2 5,5	425 192	0,08 0,03	14 7
OracleVM	130	38,7 - 41,5	161 81	35887 5635	0,35 0,05	77 28

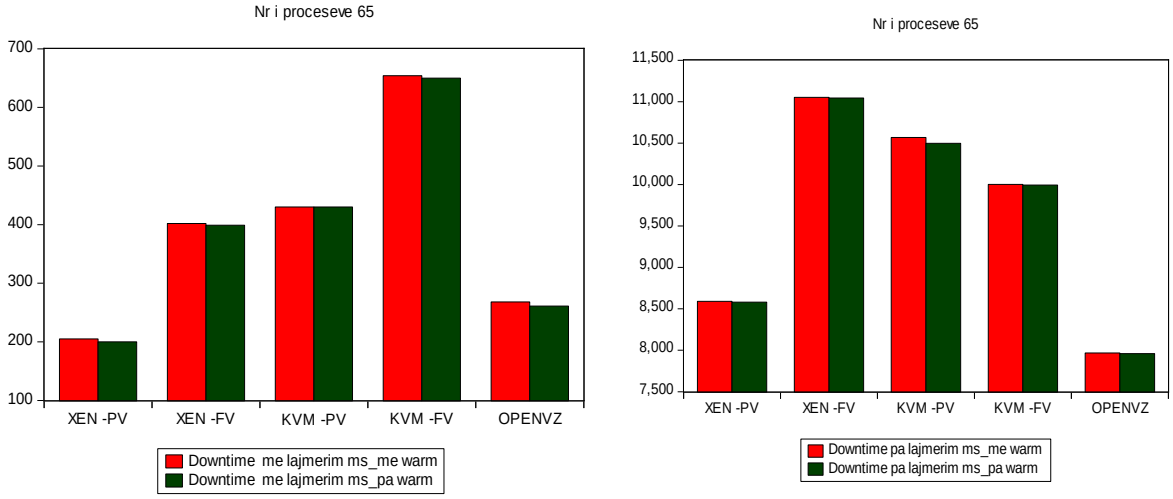


Figura 6.28- Teknikat Downtime Me/Pa lajmërim duke implementuar metodën Me/Pa Warm-UP

6.3 Përfundime

Kapitulli i gjashtë është kapitulli kryesorë i tezës, pasi këtu jepet e gjithë llogjika teorike e shtrimit të problemit, si edhe eksperimentet dhe rezultatet përkatëse. Ky kapitull ka një vlerë të jashtëzakonshme, pasi tregohet me hapa ngritja e metodës e cila përmirëson kohën totale të migrimit, kohën e bllokimit dhe trafikun e shkaktuar si pasojë e këtij migrimi.

E veçanta e kësaj metode është se ngrihet në hapsirën përdorues, çka do të thotë se ndërfaqja e komunikimit të tij është unike pavarësisht llojit të Hypervisorit të përdorur.

Gjithsesi për të përmirësuar këto parametra, janë integruar në këtë metodë teknika të ndryshme të studiura nga inxhinjerë dhe shkenctarë anembanë. Disa teknika që mund të përmenden janë:

a. Teknika RDMA
b. Teknika Warm-UP
c. Teknika e Parashikueshmërise së Bashkësisë së Punës duke përdorur algoritmat *LRU* dhe *Splay-Tree*

d. Teknikat e kompresimit dhe tabelave *Hash* në përmbajtjen e faqes
e. Teknika e Platformave të Kryqëzuara mbi Data-Base Oracle.

Teknika e fundit është edhe më e fuqishmja në përmirësimin e parametrave të migrimit.

Një tjetër vend zë edhe strukurat Hardware të cilat duhet të merren në konsideratë si psh:

a. Llojet e komunikimit SCSI apo me kanale Fiber.
b. Llojet e Storage
c. Llojet e Switcheve apo Routerave në rrjetat LAN dhe WAN.

KAPITULLI I SHTATË

KONKLUZIONET

Nga ato që dolën në këtë punim mund të jap këto përfundime:

Mund të arrihen të reduktohen humbjet e të dhënave duke implementuar një metodë apo algoritëm i cili do të bëjë të mundur të ulë në maksimum këtë kosto nga katastrofa të paparashikuara si: tërmetet, zjarret, dëmtimet e diskut të ngurtë, bordit të kompjuterit, bllokimeve të programeve etj. Ky algoritëm mund të impelementohet në çdo makinë ose mund të bëhet pjesë e vetë Sistemeve Operative, sidomos atyre me kod të hapur si open source. Kjo do të bënte që shumë njerëz të shpëtojnë nga depresioni.

Ndërkohë reduktimi i humbjes së të dhënave mund të implementohet siç treguam edhe më sipër në teknikat e virtualizimit. Kjo teknikë mund të bejë të mundur të ul në maksimum kohën totale të migrimit të makinave virtuale si pasojë e një defekti të parashikuar ose të pa lajmëruar më parë. Përveç kohës totale të migrimit që është një parametër kyç në botën e virtualizimit do të reduktohej ndjeshëm koha e bllokimit dhe trafiku i gjeneruar. Algoritmi që ngre e kam quajtur ***Live Improve*** dhe rrit në mënyrë të ndjeshme performancën e migrimit në platforma *open source* duke ulur koston e parametrave të mësipërm, në kurriz të një proçesimi të rritur lehtësisht.

Metodat të cilat janë përdorur për matjen e ketyre parametrave janë bazuar në mjete tradicionale por edhe skripte të cilat janë në seksionin e mëposhtëme shtonjcë.

Ngritja e algoritmit ***Live Improve*** do të sjellë padyshim një hap të madh cilësor në fushën e migrimit dhe ruajtjes së të dhënave pasi tani faqet e modifikuara do të jenë pjesë e diskut të ngurtë të qëndrueshëm dhe jo e memorjes së avullueshme. Ato do të shkojnë atje si pjesë të proçeseve në ekzekutim pa ndërhyrjen tonë dhe për më shumë përdoruesi nuk do të ndjejë thuajse asgjë nga ky element shtesë.

Në rast kur programi do të bllokohet për një arsye të veçantë dhe do të nevojitet ristartimi i sistemit nga e para, përdoruesi do të shikojë punën e tij të ruajtur automatikisht në PC e tij.

E njëjta gjë mund të thuhet edhe për migrimin. Migrimi duke qenë se është për një numër më të madh aplikacionesh (pasi migrohet e gjithë makina fizike), shkalla e vështirësisë këtu është më e madhe, pasi duhet të merren parasysh driverat e sistemit, gjendja e tyre, proçeset në kernel etj. Gjithsesi edhe këtu mund të realizohet një rritje e shkëlqyer e performancës duke implementuar metoda të ndryshme si:

1.Ngritja e metodës Stop and Copy

2.Kërkimi në bazë të numrave të plotë duke implementuar tabelat hash në nivel përdorues dhe kernel.

3. Ngritja e metodës së listave aktive duke i ndarë faqet zero, të lira dhe të modifikuara

4. Ngritja e metodës së ngjeshjes për faqet e lira, faqet zero, faqet në kernel.

5. Ngritja e metodës së parashikimit të bashkësisë së punës duke impelementuar algoritmat LRU dhe Splay Tree.

6. Ngritja e metodës Live Gang për një numër me më shumë se një aplikacioni për makinë dështimi

7. Ngritja e metodës RDMA për të migruar faqet me ndihmën e proçesorit ndihmës “Direct Memory Access” në rrjeta lokale dhe WAN

8. Përmirësimi i arkitekturës në aspektin e kohës së rrotullimit të diskut të ngurtë me sa më pak lëvizje dhe sektorë të kokës magnetike në të, duke kërkuar ndihmën e një sistemi “Temprorary” brenda zonës Swap të diskut i cili implementon file sistemin e vet me metodën “Warm UP”

9. Ngritja e teknikave të mësiperme në protokollin iSCSI me teknikën “Infiniband with TOE”

10. Ngritja e teknikave të mësiperme në platforma të kryqezuara në Oracle 10g duke impelementuar një sistem faturimi në rrjeta lokale dhe WAN me kanale Fibra lokalizohet brenda një rrezeje prej rreth 2 km.

Renditja e pikave të mësipërme u bë për të treguar dhe rritjen e performancës së sistemit nga një pikë në tjetrën. Ajo që vihet re në fund është se koha totale e migrimit, koha e pergjigjes, koha e bllokimit dhe trafiku ulen ndjeshëm po të krahasohen me pikën e parë. Kjo do të thotë që për çdo pikë që ne lëvizim shikohet një rritje e performancës siç jepet në tabelat në kapitullin V. Mjaftojmë të bëjmë vetëm një krahasim midis tabelave 7.1 dhe 7.2.

Tabela 7.1: Matja e kohës totale të migrimit, kohës së bllokimit dhe trafikut, brenda rrjetit lokal LAN në rastin me dhe pa aplikacion në metoden Me-Lajmërim

Lloji i HV	Nr i proçeseve	Memorja Shfrytz.MB	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të mbetura.
XEN -PV	65-0	3182(3175)	17,9 13,4	24,4 1752	300 2435	0,06 6,2	13 – 884
XEN -FV	65-0	3194(3177)	22,5 15,6	29 ,5 1964	440 2964	0,07 8,5	24 – 932
KVM - PV	65-0	3182(3176)	20,1 14,7	26,6 1905	410 2822	0,06 8,4	16 – 912
KVM -	65-0	3179(3175)	27,8 16,2	34,8	660	1,01	30 -

FV				2364	3799	0,9	1036
OPENVZ	65-0	3180(3174)	18,0 13,6	26,1 1784	330 2424	0,02 5,1	12 – 876

Tabela 7.2: Matja e kohës totale të migrimit, kohës së bllokimit dhe trafikut, brenda rrjetit lokal LAN në rastin me dhe pa aplikacion në metodën Pa-Lajmërim

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim në ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të mbetura
XEN -PV	65-0	300 2435	17535 Block	0,06 6,2	3,5 87	23 – 7757(8050)
XEN -FV	65-0	440 2964	18642 Block	0,07 8,5	3,7 92	33 – 7956 (8050)
KVM - PV	65-0	410 2822	17974 Block	0,06 8,4	3,5 88	25 – 7810 (8050)
KVM - FV	65-0	660 3799	21005 Block	1,01 10,9	3,9 96	41 – 8020 (8050)
OPENVZ	65-0	330 2424	16604 Block	0,02 5,1	3,3 88	24 – 7715 (8050)

Tabela 7.3: Matja e kohës totale të migrimit, kohës së bllokimit dhe trafikut, brenda rrjetit lokal LAN në rastin me dhe pa aplikacion në metodën Me-Lajmërim

Lloji i HV	Nr i proçeseve	CPU Shfrytz.%	Koha e migrimit (s)	Downtime ms	Over-head %	Nr i faqeve të Humbura.
XEN -PV	65-0	25,4 - 15,7	8,6 9,4	200 205	0,003 0,003	4 4
XEN -FV	65-0	36,2 – 26,4	12,6 14,5	399 402	0,011 0,012	11 11
KVM - PV	65-0	31,1 – 20,5	10,1 12,3	430 430	0,003 0,003	5 6
KVM - FV	65-0	40,8 – 30,9	19,5 21,1	650 654	0,017 0,018	13 14
OPENVZ	65-0	22,8 – 13,3	8,4 9,7	261 268	0,005 0,006	3 3

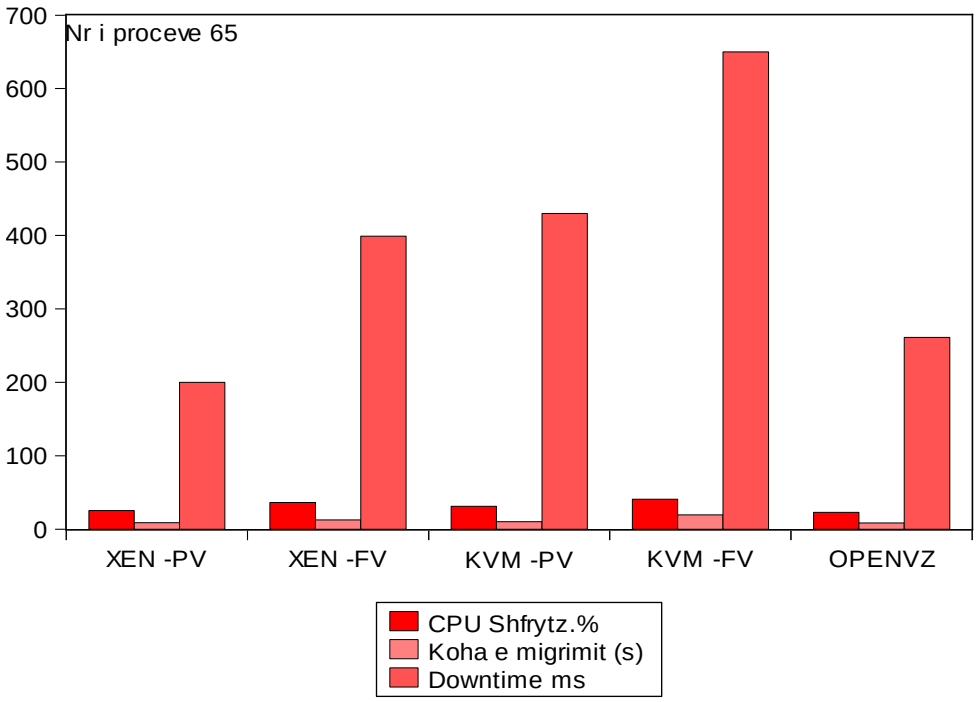


Figura 7.1 :Shfrytëzimi i CPU-se, Kohes Totale te Migrimit dhe Bllokimit ne rrjetin lokal LAN ne rastin me dhe pa aplikacion ne metoden Me-Lajmerim me tekniken Warm-UP

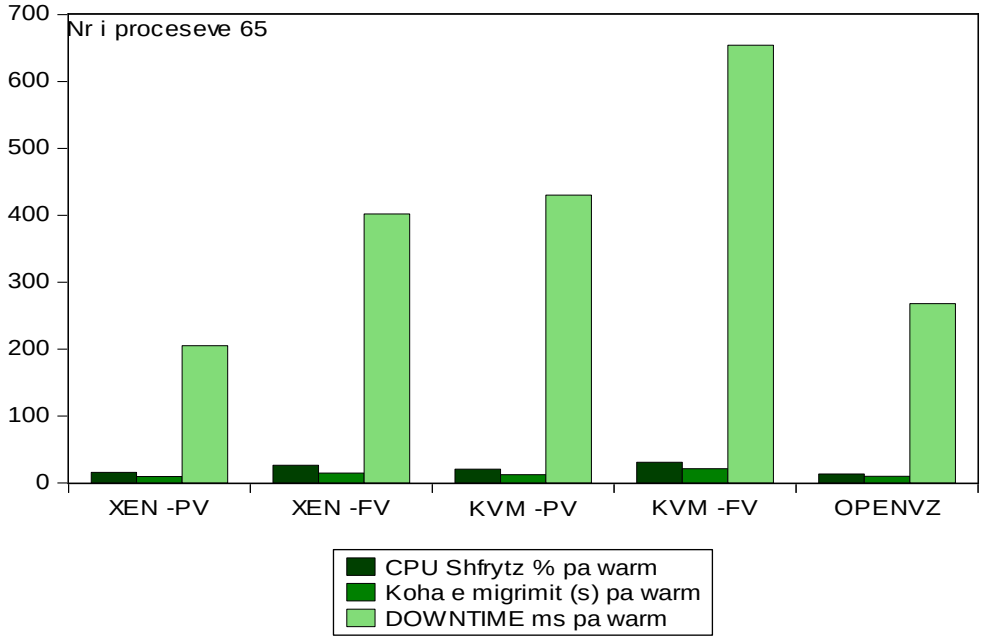


Figura 7.2 :Shfrytezimi i CPU-së, Kohës Totale të Migrimit dhe Bllokimit në rrjetin lokal LAN në rastin me dhe pa aplikacion në metodën Me-Lajmërim pa përdorur tekniken Warm-UP

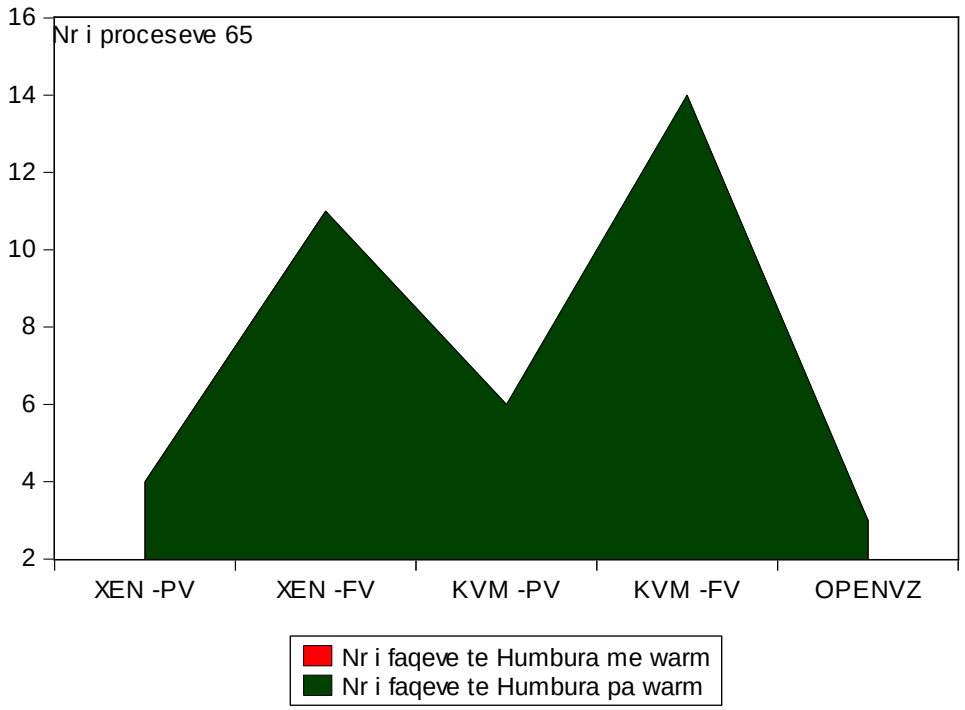


Figura 7.3 Përcaktimi i numrit të faqeve të humbura për rastin me dhe pa aplikacion duke krahasuar teknikat me/pa Warm-UP

Tabela 7.4 Ngritja e teknikës në një rrjet LAN duke caktuar bashkësine minimale të punës me anë të algoritmit LRU mbi metodën RDMA dhe Warm-UP (Para-Ngrohjes) në metoden Pa-Lajmërim

Lloji i HV	Nr i proçeseve	Downtime me lajmërim ms	Downtime pa lajmërim në ms	Overhead me lajmërim %	Overhead pa lajmërim %	Nr i faqeve të humbura

XEN -PV	**65-0**	**200** **8580**	**205** **8591**	**0,003** **2,1**	**0,003** **2,1**	**2,1** **2,1**	**10** **10**
XEN -FV	**65-0**	**399** **11044**	**402** **11052**	**0,011** **2,6**	**0,012** **2,6**	**2,6** **2,6**	**21** **22**
KVM - PV	**65-0**	**430** **10498**	**430** **10569**	**0,003** **2,2**	**0,003** **2,2**	**2,2** **2,2**	**13** **14**
KVM - FV	**65-0**	**650** **9994**	**654** **10002**	**0,017** **2,6**	**0,018** **2,6**	**2,6** **2,6**	**30** **30**
OPENVZ	**65-0**	**261** **7960**	**268** **7968**	**0,005** **2,1**	**0,006** **2,1**	**2,1** **2,1**	**11** **12**

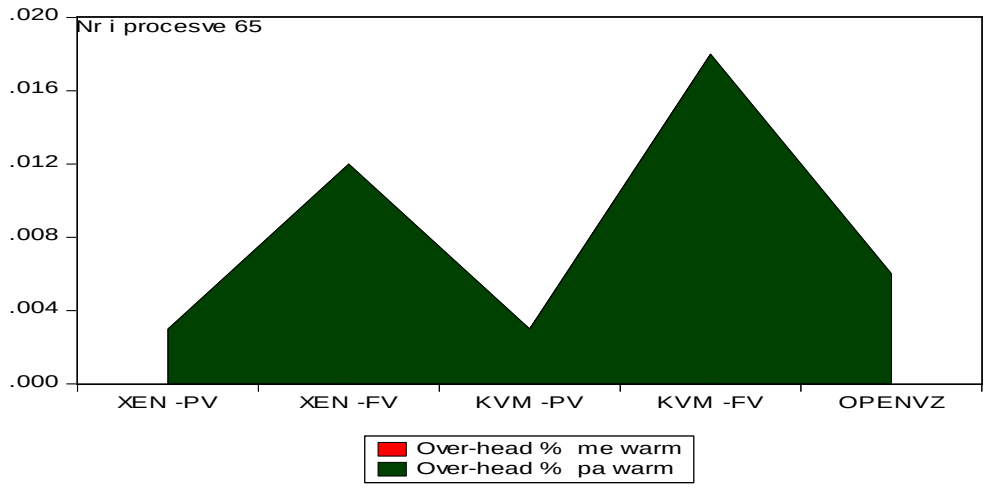


Figura 7.4 Përcaktimi i overhead-it për rastin me dhe pa aplikacion duke krahasuar teknikat me/pa Warm-UP në tekniket Me Lajmërim

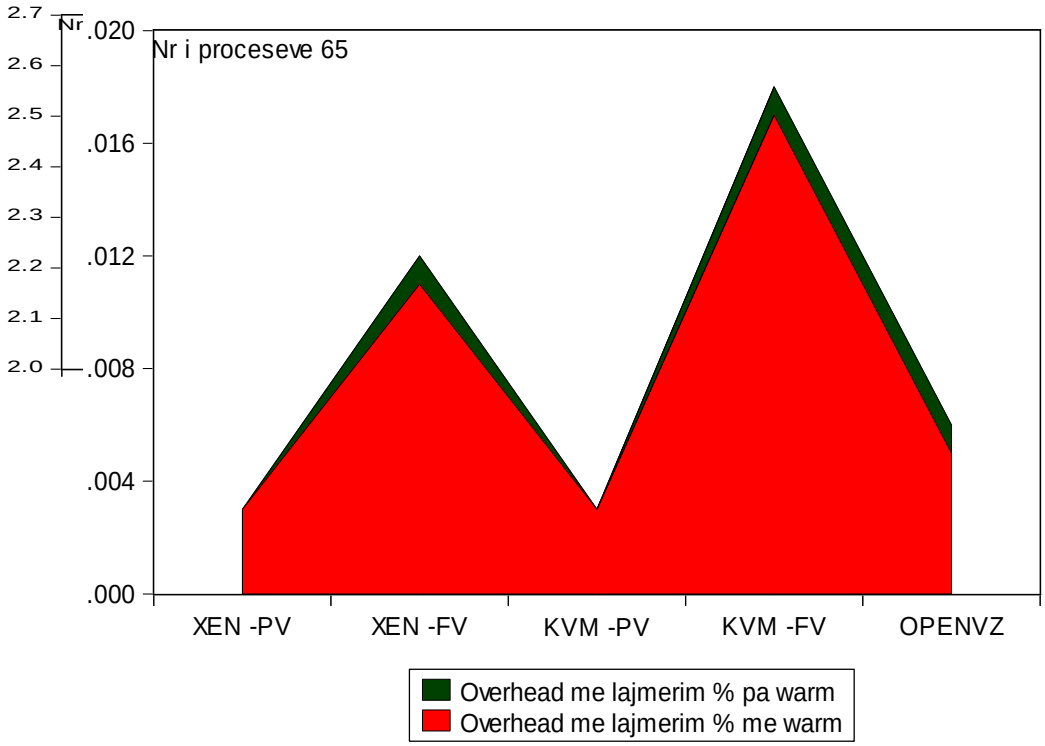


Figura 7.5 Përcaktimi i overhead-it për rastin me dhe pa aplikacion duke krahasuar teknikat me/pa Warm-UP në teknikat Pa Lajmërim

Krahasimi i këtyre dy tabelave është një shembull i gjallë i evoluimit të efiçensës së algoritmit tonë ***Live Improve***. Me anë të këtij algoritmi kuptojmë që të gjithë parametrat e përmendur më sipër janë rritur në menyrë të shkëlqyer.Kjo është dhe fuqia e këtij algoritmi.

Ajo që mbetet për të ardhmen është ngritja e këtij algoritmi në sitemet me kod të mbyllur dhe nëse i referohemi teknikave të virtualizimit, mund të përmend supervizorin Hyper-V që është dhe përfaqesuesi i Microsoft. Ne kodin e çdo sistemi Microsoft siç dihet nuk mund të nderhym dot, për arsye të disa ligjeve ekonomike.

KAPITULLI NËNTË

REFERENCAT

[1] http://en.wikipedia.org/wiki/Application_virtualization

[2] http://www.webopedia.com/TERM/A/application_virtualization.html

[3] http://en.wikipedia.org/wiki/Desktop_virtualization

[4] http://en.wikipedia.org/wiki/Network_virtualization

[5] http://searchservervirtualization.techtarget.com/definition/server-virtualization

[6] http://searchservervirtualization.techtarget.com/definition/virtual-machine

[7] http://searchdomino.techtarget.com/definition/application-clustering

[8] http://en.wikipedia.org/wiki/Paravirtualization

[9] http://searchservervirtualization.techtarget.com/definition/paravirtualization

[10] http://en.wikipedia.org/wiki/Full_virtualization

[11] http://en.wikipedia.org/wiki/Hardware_virtualization

[12] http://ssrg.nicta.com.au/publications/papers/LeVasseur_UYCCLH_08.pdf

[13] http://en.wikipedia.org/wiki/Storage_virtualization

[14] http://en.wikipedia.org/wiki/Operating_system-level_virtualization

[15] http://en.wikipedia.org/wiki/Emulator

[16] http://fengnet.com/book/solaris_admin/ch09lev1sec4.html

[17] Dalessandro, D., Wyckoff, P., (2008), Memory Management Strategies for Data Serving with RDMA.

[18] Hines, M., Gopalan, K., (2009), Post-Copy Live Migration of Virtual Machine, SIGOPS Operating System Review, vol 43, pp. 14

[19] http://xdelta.org/xdelta3.html

[20] Zaw, E.,P., Thein N., L., (2012), Improved Live VM Migration using LRU and Splay Tree Algorithm, International Journal of Computer Science and Telecommunications, Volume 3, Issue 3.

[21] http://en.wikipedia.org/wiki/Splay_tree

[22] Huang,W., Gao,Q., Liu J., Dhabaleswar K.,High Performance Virtual Machine Migration with RDMA over Modern Interconnects, (2009), pp 11-20

[23] Deng,L., Wu,S., Shi, X., (2009), Live Virtual Machine Migration with adaptive memory compression, Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International, pp 1-10.

[24] Yuan, M., Minimizing Data Migration Time, using Cross-Platform Transportable Tablespace, (2005).

[25] Das,S., Kagan, M., Crupnicoff,D., Faster and Efficient VM Migration for Improvin SLA and ROI in Cloud Infrastructure, (2009)

[26] Tanenebaum,A., (2007),Modern Operating System, Chap 3, LRU Algorithm, pp211

[27] Tuduce, Ch.,I., Gross, T., Adaptive Main Memory Compression, (2010),USENIX

[28] Raj,R., Leelipushpam, G.J, Live Virtual Machine Migration Techniques–Survey, (2012), International Journal of Engineering Research & Technology (IJERT) Volumi 1

[29] Wo, Y., Zhao, M., Performance Modeling of Virtual Machine Live Migration, (2011), CLOUD '11, IEEE 4th International Conference on Cloud Computing, p.p 492-499

[30] Deshpande,U., Wang, X., Gopalan, K., Live Gang Migration of Virtual Machines, (2005), USENIX

[31] Akoush,Sh., Sohan,R., Rice A., Hopper A.,(2010), " Predicting the Performance of Virtual Machine Migration" In proceeding of: MASCOTS 2010, 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Miami, Florida, USA.

[32] Hacking, S., Hudzia, B., Improving the Live Migration Process of Large Enterprise Applications, (2009), VTDC'09 Proceedings of the 3rd international workshop on Virtualization technologies in distributed computing, pp. 51-58.

[33] http://www.answers.com/topic/rdma

[34] http://en.wikipedia.org/wiki/Remote_direct_memory_access

[35] Tafa, I., Beqiri, E., Kajo, E., Hakik Paci, Xhuvani, A., (2011) The evaluation of Transfer Time, CPU Consumption and Memory Utilization in XEN-PV, XEN-HVM, OpenVZ, KVM-FV and KVM-PV Hypervisors using FTP and HTTP approaches, Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference IEEE, pp. 502-507

[36] Tanenbaum A., (2007) Modern Operating System, Chap 2, Process and Threads – Semaphore pp.140

[37] Nussbaum,L., Anhalt,F., Mornard,O., (2009), Linux-based virtualization for HPC clusters, Linux Symposium

[38] Heo, J., Zhu, X., Padala, P., Wang, Zh., Memory Overbooking and Dynamic Control of Xen Virtual Machines in Consolidated Environments, (2009).

[39] http://en.wikipedia.org/wiki/Bonnie%2B%2B

[40] D., Duelli M., Goll,S., (2010), Performance Comparison of Hardware Virtualization Platforms, Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part I, Pages 393-405, ISBN: 978-3-642-20756-3

[41] Tanenbaum A., (2007) Modern Operating System, Chap 3, Memory,demand paging pp.212.

[42] http://www.linux.com/news/enterprise/systems-management/327628-kvm-or-xen-choosing-a-virtualization-platform

[43] http://www.webhostingtalk.com/showthread.php?t=1000747

[44] http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=3243&context=etd

[45] http://en.wikipedia.org/wiki/Oracle_VM

[46] http://codeincodeblock.blogspot.com/2012/06/telecom-billing-management-system-in-c.html

[47] www.qumranet.com/