Navidezni stroji - Xen

Seminarska naloga

Martin Perne

December 2021

1 Uvod

V seminarski nalogi bom najprej predstavil virtualizacijo in navidezne stroje na splošno. Opisal bom različne tipe virtualizacije, malo bolj podrobno bom opisal sistemsko virtualizacijo, predstavil problematiko in možne rešitve pri implementaciji le-te. Nato bom podrobneje opisal delovanje programske opreme Xen. Osredotočil se bom na implementacije glavnih in najbolj pomembnih delov za omogočanje virtualizacije, specifično paravirtualizacije.

2 Virtualizacija in navidezni stroji

Virtualizacija pomeni ustvariti nekaj navidezno. Virtualni oziroma navidezni stroj torej ni dejanski stroj, vendar teče na nekem drugem sistemu, ki pa svoje vire preslika na navidezni stroj.

Poznamo več vrst virtualizacije. Virtualiziramo lahko samo nek proces z uporabo programske opreme, kot je javanski navidezni stroj, kar omogoča prenosljivost programov. Lahko virtualiziramo strežnike, torej fizični strežnik premaknemo v navidezno okolje. En fizični strežnik lahko na tak način v resnici poganja več navideznih strežnikov. Lahko pa virtualiziramo celoten sistem in tako pridobimo sistemski navidezni stroj. O tej vrsti virtualizacije bom tudi največ govoril.

2.1 Sistemska virtualizacija

Pri virtualizaciji sistemskega navideznega stroja virtualiziramo celotno strojno opremo. Navidezni stroj se obnaša kot pravi stroj z operacijskim sistemom. Pri virtualizaciji si pomagamo s programsko/strojno opremo, ki ji rečemo nadzornik (ang. hypervisor). Le-ta ustvari in upravlja navidezne stroje. Računalniku, na katerem s pomočjo nadzornika teče en ali več navideznih strojev, rečemo gostitelj. Navideznemu stroju rečemo gost.

Nadzornike klasificiramo na dva tipa. Prvemu rečemu tip 1 ali "native" in teče neposredno na strojni opremi, na njem pa lahko teče več navideznih stro-

jev. Drugemu rečemo tip 2 oziroma gostovan nadzornik, ki teče na normalnem operacijskem sistemu na enak način kot drugi procesi.

Sistemsko virtualizacijo delimo tudi na polno virtualizacijo in paravirtualizacijo. Pri polni virtualizaciji se gostujoči operacijski sistem ne zaveda, da je virtualiziran in teče na enak način kot v nevirtualiziranem okolju. Pri paravirtualizaciji pa pridobimo na učinkovitosti, vendar je treba navidezni stroj modificirati in ve, da se izvaja virtualizirano. Gostojuči operacijski sistem komunicira in sodeluje z nadzornikom in tako pridobi na učinkovitosti.

2.2 Implementacija virtualizacije

Za učinkovito procesorsko virtualizacijo bi morali imeti idealne ukaze, katerih rezultati so vedno predvidljivi/pričakovani ne glede na stopnjo privilegija, v kateri se izvaja. V praksi temu ni tako - x86 arhitektura ima več problematičnih ukazov. Kot že prej omenjeno, to rešujemo z uporabo nadzornika. Nadzornik teče v najbolj privilegiranem načinu (obroč 0 oz. nižje, če imamo podporo strojne opreme) in nadzira vse procesorske strukture, ki niso v skupni rabi med različnimi operacijskimi sistemi. Takšna struktura je npr. tabela prekinitvenih vektorjev. Če CPE ne podpira virtualizacije, bo nadzornik moral teči na višji stopnji privilegiranosti kot operacijski sistemi - operacijski sistem bomo morali torej premakniti v bolj zunanji obroč privilegija.

2.2.1 Emulacija

Preprost način virtualizacije je emulacija CPE. Z programsko opremo, kot je Bochs, lahko emuliramo strojno opremo. Implementiran ima celoten nabor ukazov in emulira fizični CPE. To naredi tako, da ukaze, ki so mu podani, oponaša z notranjimi funkcijami. Največja slabost takšne virtualizacije je učinkovitost, saj za vsak podan ukaz emulator potrebuje več ukazov na fizičnem CPE.

2.2.2 Prevajanje

Problematične ukaze v x86 naboru ukazov lahko nadomestimo z različnimi, drugimi ukazi, ki emulirajo originalne. Prevedeni ukazi so nato združeni v bloke, ki se izvedejo skupaj na fizičnem procesorju. Na koncu vsakega bloka je ukaz, ki preda nadzor nazaj nadzorniku. Takšnemu postopku rečemo prevajanje (ang. binary translation). Uporabljajo ga npr. različni produkti VMware, ko virtualizacija ni podprta s strani strojne opreme.

2.2.3 Paravirtualizacija

Paravirtualizacija zahteva, da najprej operacijski sistem modificiramo tako, da ve, da se izvaja v virtualiziranem okolju. Potem lahko dostopa do programskega vmesnika, ki zamenja problematične ukaze. Prednost takšne virtualizacije je visoka učinkovitost, vendar pa je treba gostujoči OS modificirati. Posledica tega je, da mora gostujoči OS biti odprtokoden - težko bi bilo uporabiti tak

način virtualizacije na operacijskem sistemu Microsoft Windows. Programska oprema, ki to podpira je npr. Xen.

2.2.4 Podpora s strani strojne opreme

Moderni AMD in Intel-ovi procesorji imajo virtualizacijsko podporo že vgrajeno - dodani so določeni ukazi, ki pomagajo pri virtualizaciji. Tako se lahko katerikoli OS izvaja direktno v kontekstu navideznega stroja brez programskih rešitev, ki so bile naštete prej. S takšno podporo ima nadzornik lastno stopnjo privilegija, gostujoči OS pa se ne zaveda, da je virtualiziran. Uporablja se nadzorniška tabela, ki vsebuje različne podatke o konfiguraciji navideznega stroja in nadzira tudi dogodke, kot so dostop do procesorskih nadzornih registrov/tabel. Programska oprema, kot je Xen, VMware ali Microsoft Hyper-V podpira tudi CPE-razširjeno virtualizacijo. Na tak način je mogoča polna virtualizacija brez počasnosti čiste emulacije, ki je bila predstavljena prej.

3 Xen

Xen project je odprtokodna programska oprema. Kot projekt je sestavljen iz skupnosti ljudi, ki so pooblaščeni za spreminjanje izvorne kode, vodje skupnosti in svetovalnega odbora. Ima razne sponzorje in tudi sodeluje z skupino Linux Foundation. Ima tudi več podprojektov, kot so XCP-ng, XAPI, unikraft in več.

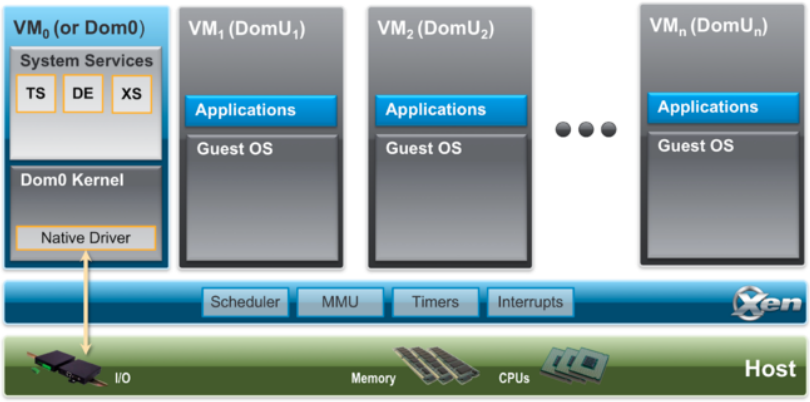
Xen project je nadzornik tipa 1 - teče torej neposredno na fizičnem stroju. Podpira več vrst virtualizacije, glavni dve sta paravirtualizacija in polna virtualizacija (Nadzornik torej ustvari celotno strojno opremo). Druge vrste so pa mešanice teh dveh pristopov, kjer ima en izmed pristopov izboljšave drugega, da se pridobi na učinkovitosti ali preprostosti implementacije. Takšne vrste so torej polna virtualizacija z gonilniki paravirtualizacije (PVHVM), paravirtualizacija na polni virtualizaciji in paravirtualizacija na vsebniku polne virtualizacije (PVH).

3.1 Arhitektura

Xen project nadzornik je programska plast, ki se izvaja na strojni opremi in upravlja z CPE, spominom in prekinitvami, ne upravlja pa z vhodno-izhodnimi napravami. Omembne vredna je njena velikost, nadzornik za x86 arhitekturo namreč obsega manj kot tristo tisoč vrstic kode.

Xen reče navideznim strojem, ki tečejo, domene. Eni instanci navideznega stroja se torej reče domena (ang. domain), lahko tudi gost (ang. guest). Domene se deli na dva tipa: priviligirane in nepriviligirane. Nepriviligirane domene nimajo dostopa do strojne opreme. Nepriviligirani domeni na splošno se reče Domena U oz. DomU na kratko, posamezni domeni pa DomU_i. Priviligirana je samo ena domena, tej se reče Domena 0 oz. Dom0 na kratko - to je v bistvu posebna vrsta navideznega stroja in se tudi prva zažene. Je sicer gostujoči OS, vendar ima poseben dostop do strojne opreme. Nadzornik brez Domene 0 ni

uporaben. Mehanizme torej implementira nadzornik, vendar samo upravljanje prepusti Domeni 0.



Slika 1: Poenostavljen diagram Xen arhitekture, vir: https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview

Domena 0 upravlja s sistemskimi storitvami in je vir gonilnikov fizičnih in navideznih naprav. Domenami U omogoča dostop do strojne opreme preko BackendDriverja in FrontendDriverja. Uporabniku z orodji, imenovanimi *toolstack* omogoča ustvarjanje, konfiguriranje in brisanje nepriviligiranih navideznih strojev. To mu je omogočeno preko ukazne vrstice ali grafičnega vmesnika, lahko pa tudi iz storitve v oblaku - odvisno od toolstacka, saj obstaja več različnih.

3.2 Nadzornik

Nadzornik ima nadzor nad stvarmi, kot je razvrščevalnik (scheduler), upravljanjem s pomnilnikom (memory management), časovniki, prekinitvami (interrupts).

Za komunikacijo med nadzornikom in gostom Xen uporablja t.i. nadzorne klice (ang. hypercall). Nadzorni klic je nadzorniku to, kar je sistemski klic jedru. Je torej programska past iz domene (torej navideznega stroja) v nadzornik. Domene uporabljajo nadzorne klice za zahteve privilegiranih operacij, kot so posodabljanje ostranjevalne tabele (pagetable). Gost lahko izvede nadzorni klic s klicem naslova v tabeli, imenovani *stran nadzornih klicev* (ang. *hypercall page*). To je pomnilniška stran, ki je preslikana v naslovni prostor gosta, ko je sistem prvič zagnan.

3.2.1 Razvrščevalnik

Xen ima več vrst razvrščevalnikov: Credit, Credit2 in (poskusno, v razvoju) RDTS-Based scheduler. Credit2 in RDTS je potrebno z uporabnikove strani eksplicitno aktivirati.

Credit je obtežen, sorazmeren in pravičen razvrščevalnik za navidezne CPE. Vsakemu navideznemu stroju je dodeljena teža in maksimalna vrednost max. Če je $max == 0$, je navidezni stroj dan v način ohranjanja dela, če pa je $max > 0$, to pomeni, da navidezni stroj ne bo tekel dalje od določenega časa - tudi če je sistem nedejaven.
Algoritem je naslednji:

- Vsaka fizična CPE ima prioritetno vrsto navideznih CPE - vCPE. Če je vCPE že izkoristila svoj čas, ima podano prioriteto *OVER*, sicer ima prioriteto *UNDER*. Pri vstavljanju v vrsto je vCPE dodana na konec vrste vseh drugih vCPE z isto prioriteto.

- Po končanem izvajanju vCPE porabi *kredite*. Na določen čas nit preračuna koliko kreditov ima vsak navidezni stroj in doda nove. Če ima vCPE manj kot 0 kreditov, ima prioriteto OVER, sicer ima prioriteto UNDER.

- Ko neka vCPE preda nadzor, je naslednja vCPE že izbrana iz prioritetne vrste.

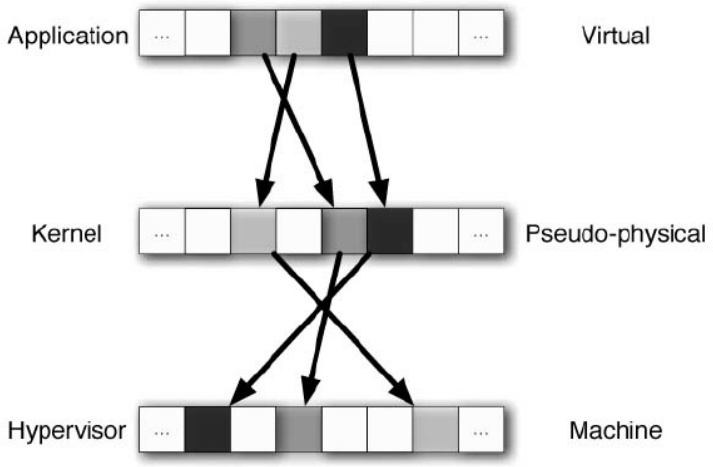
Tipično Credit razvrščevalnik uporablja $30ms$ dolge časovne rezine za alociranje CPE. Vsaka vCPE ima 30ms za izvajanje, nato mora predati nadzor. Prav tako so vsakih 30ms preračunani krediti.

Credit2 je bil razvit z namenom višje skalabilnosti. Na tak način bi bil bolj primeren za uporabo pri strežnikih. Je nadgradnja Credit razvrščevalnika - je bolj pošten in ima manjše zamike.

RDTS-Based scheduler stoji za Real-Time-Deferrable-Server-Based CPU scheduler. Dodan je bil za zagotovljeno CPE izvajanje na gostiteljih, ki uporabljajo simetrično večprocesiranje (Symmetric multiprocessing - SMP).

3.2.2 Upravljanje s pomnilnikom

Pomnilniku, za katerega navidezni stroj misli, da je fizični, rečemo psevdo-fizični pomnilnik. Za ta pomnilnik se tudi paravirtualizirani stroj ne zaveda, da je virtualiziran. Slika 1 prikazuje različne vrste pomnilnika v Xen-u. Na vrhu je (popolnoma) navidezni pomnilnik, ki ga uporabljajo aplikacije gostujočega OS. To se preslika v psevdo-fizični pomnilnik, za katerega gost misli, da je fizični pomnilnik. Iz psevdo-fizičnega pomnilnika se nato naslovi preslikajo v pravi fizični oz. strojni pomnilnik.



Slika 2: Pomnilnik v Xen-u, vir: David Chisnall - The Definitive Guide to Xen Hypervisor - Prentice Hall (2007), str. 80 (Figure 5.1)

Pri virtualizaciji je ponavadi spomin implementiran tako, da nadzornik doda dodatno stopnjo abstrakcije med psevdo-fizičnim pomnilnikom in strojnim pomnilnikom. To se ponavadi implementira z preslikavo fizično-v-strojno (ang. physical-to-machine mapping). To bi upravljal nadzornik in bi bilo skrito od gostujočega OS z uporabo tehnik, kot so senčne preslikovalne tabele. Senčna preslikovalna tabela bi bila torej takšna, kot bi gostujoči OS mislil, da je njegova prava preslikovalna tabela, nadzornik pa bi vodil lastno preslikovalno tabelo, ki slika iz navideznih naslovov v fizične.

V Xen-u pa se paravirtualizirani gost zaveda preslikave fizično-v-strojno in je spremenjen na tak način, da namesto tega, da piše vnose v preslikovalno tabelo, ki bi se iz navideznega (aplikacijskega) pomnilniškega prostora nato preslikala v psevdo-fizični pomnilniški prostor rajši navidezne naslove preslika neposredno v strojni pomnilniški prostor. To je v Xen-u znano kot neposredno ostranjevanje (ang. direct paging).

Zaradi neposrednega ostranjevanja potrebujemo dve tabeli: gostujoči OS mora poznati preslikavo med psevdo-fizičnimi naslovi in strojnimi naslovi, to se reši z uporabo t.i. P2M tabele (physical-to-machine table). Potrebuje pa tudi M2P tabelo (machine-to-physical table) za preslikovanje iz strojnih naslovov v psevdofizične naslove. Obe tabeli vsebujeta številke okvirjev strani, indeksirane z psevdo-fizičnimi ali strojnimi okvirji. P2M tabela je upravljana popolnoma s strani gosta - njena velikost je tudi odvisna od naslovnega prostora gosta. Velikost M2P tabele pa je odvisna od gostitelja - lahko je torej precej velika, težavno je pa tudi to, da je lahko velik del te tabele za enega gosta neuporabna, saj veliko strojnih strani ne bo pripadalo temu gostu. Zato Xen poda bralno verzijo M2P tabele gostu in omogoča tudi posodabljanje le-te z nadzornim klicom.

Vendar pa gostu Xen ne zaupa preveč - vsako spremembo ostranjevalne oz. preslikovalne tabele rabi Xen odobriti. To doseže tako, da zahteva, da je preslikovalna tabela vsakič posodobljena z uporabo nadzornega klica. Definira tudi štiri različne tipe strani - vsaka stran ima lahko sam en tip. Tipi so naslednji:

- Brez - stran nima posebne uporabe
- L_N stran preslikovalne tabele - strani, ki so uporabljene kot tabela strani na stopnji N. Tipi so različni za vsake od 3 (32bit) oz. 4 (64bit) stopenj na gostih.
- Stran deskriptorja segmenta (ang. segment descriptor page) - uporabljena kot del globalnih ali lokalnih deskriptorskih tabel
- Zapisljiva (ang. writable)

Piše se lahko samo v zapisljive tabele. Tako se nadzornik prepriča, da gostujoči OS ne piše na prostore, kamor ne bi smel (varnostna grožnja). Če ima tabela tip L_3, mora biti referencirana s strani tabele tipa L_4. To se rekurzivno ponavlja, dokler ne pridemo do tabel s tipi "Zapisljiva". S takšnimi tipi in zahtevami se Xen prepriča, da je množico preslikovalnih tabel varno naložiti v bazni register preslikovalnih tabel. Xen mora rekurzivno preveriti preslikovalne tabele, ko so naložene v register, ob vsakem preklopu konteksta (ang. context switch) - to je zahtevna/neučinkovita operacija. Zato lahko določeno stran "zataknemo" (ang. pin) na določen tip. Na tip tabele torej dodamo referenco, Xen pa validira tabelo že vnaprej. Ko je nova tabela naložena, ima referencirana tabela že primerni tip (to je npr. L4 za 64bitne sisteme).

Ker gost ne more direktno pisati v preslikovalne tabele, Xen ponuja določene storitve, s katerimi lahko gost posodobi vnos v preslikovalni tabeli - to so različni nadzorni klici, kot so *mmu_update* in *update_va_mapping*.

Tudi druge privilegirane operacije, kot je splakovanje TLB, mora opraviti nadzornik. To se doseže z nadzornim klicem *mmuext_op*.

Ker lahko veliko število nadzornih klicev pri določenih operacijah upravljanja s spominom vpliva na učinkovitost, Xen omogoča tudi funkcionalnost *multicall*. Tako se lahko več nadzornih klicev izvede skupaj.

Gosti, ki niso paravirtualizirani in se ne zavedajo virtualizacije, seveda upravljajo s pomnilnikom na natančno enak način kot sicer (iz njihovega pogleda). To je sicer manj učinkovito kot pristop paravirtualizacije.

3.2.3 Dogodki in prekinitve

Dogodek v Xen-u je ekvivalenten strojni prekinitvi. Pošiljajo se s pomočjo dogodkovnega kanala. Dogodek je v resnici en bit - pripeti se, ko se ta bit spremeni iz 0 v 1. Xen gostu pošlje obvestilo s klicom navzgor, s tem prispe dogodek. Naslednja obvestila so zamaskirana, dokler bit ni spet nastavljen na 0.

Dogodke se lahko zamaskira s postavitvijo zastavice - to je ekvivalentno onemogočanju prekinitev. Uporablja se za atomičnost določenih operacij v gostujočem OS.

Z uporabo dogodkovnih kanalov lahko pošiljamo štiri vrste dogodkov:

- Meddomenska obvestila
- Virtualne zahteve za prekinitve (VIRQ) - ponavadi uporabljene za časovnike.
- Medprocesorske prekinitve (IPI)
- Fizične zahteve za prekinitve (PIRQ)

Dogodki sami so shranjeni v bitnem polju med gostom in nadzornikom. Dogodkovni kanali so implementirani kot 2 ali 3 nivojski za hitrejše iskanje. Pri dvonivojskem je prvi nivo niz bitov, ki vsebuje čakalne dogodkovne bite. Drugi nivo je niz bitov čakalnih dogodkov.

3.2.4 Skupni pomnilnik

Xen za komunikacijo in prenos podatkov med domenami omogoča mehanizem skupnega pomnilnika.

Xen dostop nadzira z *dodelitveno tabelo* (ang. *grant table*). Vsaka domena ima lastno dodelitveno tabelo, Xen jo lahko prebere in na tak način ve, kakšna dovoljenja imajo druge domene nad stranmi domene, katere dodelitveno tabelo bere.

Vnosi v dodelitveni tabelah so identificirani z dodelitvenimi referencami. To je število, ki je nato indeksirano v dodelitveno tabelo. Na tak način lahko druga domena izvaja operacije nad domeno, ki ji je dodelila dovoljenja. Referenca enkapsulira tudi podrobnosti skupne strani v pomnilniku, na tak način domena ne potrebuje znanja o strojnem naslovu strani, ki jo da v skupno rabo. To omogoča skupni pomnilnik tudi z domenami, ki imajo popolnoma virtualizirani pomnilnik.

S skupnim pomnilnikom je mogoče upravljati preko nadzornega klica, imenovanega grant_table_op. Vzame tri argumente: tip operacije (ukaz), tabelo struktur, ki vsebujejo operacije in število operacij v tabeli. Operaciji sta dve: Preslikovanje, kjer ustvarimo skupni pomnilnik, ali pa prenos podatkov, kjer domena, ki kliče nadzorni klic, pošilja podatke drugi.

Z dodelitveno tabelo lahko upravljamo direktno. Vnos, ki ga dodajamo, mora imeti definirane naslednje vrednosti:

- identifikator domene
- zastavice, kot je npr. tip pravice, ki jo dodajamo.
- okvir, do katerega dodajamo dostop ali okvir, na katerem sprejmemo prenos

Večina domen lahko upravlja le z lastno dodelitveno tabelo, Domena 0 pa lahko tudi z drugimi.

3.3 Domena 0

Domena 0 je nekakšen nadzornik. Upravlja s sistemskimi storitvami. To so storitve, kot so XenBus, XenStore, Toolstack in Device Emulation. Ima tudi nadzor nad gonilniki in dostopom do naprav. Domene U morajo namreč do naprav priti preko Domene 0.

3.3.1 XenBus in XenStore

XenBus je abstrakcija vodila, ki paravirtualiziranim gonilnikom omogoča komunikacijo med domenami. Uporablja se za bolj za spremembe konfiguracije kot prenos podatkov. Podatki se prenašajo preko meddomenskega kanala, ki je sestavljen iz skupnih strani v spominu in dogodkovnega kanala. Ob inicializaciji se mora vsak gonilnik naprave registrirati pri XenBusu.

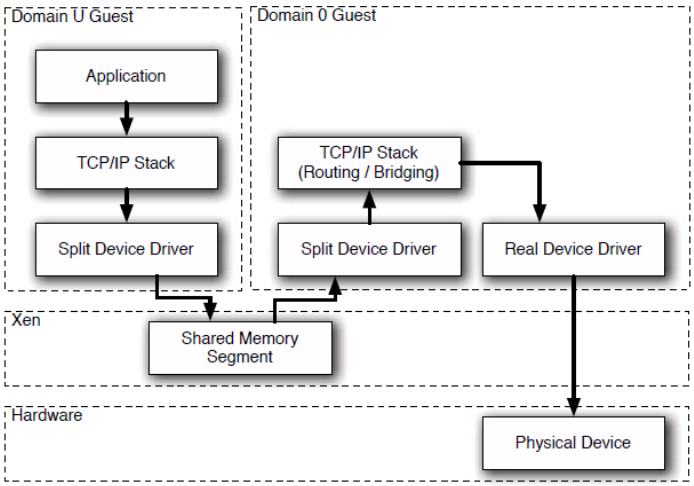
XenStore je prostor za shranjevanje informacij med domenami. Struktura spominja na datotečni sistem. Ni namenjen za prenos velike količine podatkov, vendar za stvari, kot so informacije o konfiguraciji in informacije o statusu. Upravlja ga proces v ozadju (daemon). Xen podpira dve vrsti tega procesa: cxenstored in oxenstored. Večinoma se uporablja le cxenstored, vendar ima oxenstored določene optimizacije za veliko število gostov. Dostopen je preko vtičnika v Domeni 0, jedrnega API-ja ali "input-output control" vmesnika preko XenBus-a.

3.3.2 Gonilniki

Za dostop do naprav ima Xen implementirane t.i. razdeljene gonilnike naprav (ang. split device drivers).

BackendDriver in FrontendDriver sta dva gonilnika, preko katerih Domene U lahko pošljejo zahteve za dostop do strojne opreme. FrontendDriver gostujočemu OS zgleda kot navaden disk ali omrežni vmesnik. Pomembno je vedeti, da je ta gonilnik nepriviligiran. Preko FrontendDriverja torej gostujoči OS pošlje zahtevo, ta se s pomočjo XenBus, XenStore in skupnih strani v spominu prenese do BackendDriverja, ki se nahaja v Domeni 0. BackendDriver ima več nalog: multipleksira vhode (tako omogoča kominiciranje z več FrontendDriverji naenkrat) in komunicira z gonilnikom naprav, (Device driver), ta pa potem komunicira z fizičnimi vhodno-izhodnimi napravami strojne opreme.

XenStore in XenBus torej ne uporabljamo za prenos podatkov, vendar rajši za oglaševanje podatkov v skupnem pomnilniku. Primer (poenostavljen): Po omrežju pošiljamo paket. Najprej ta paket enkapsuliramo z pomočjo TCP/IP protokolov in shranimo v skupen pomnilnik s pomočjo gonilnika BackendDriver. Ta pomnilniški prostor je oglaševan s pomočjo XenStore in XenBus. Gonilnik FrontendDriver v Domeni 0 potem ta paket pošlje po fizični napravi z uporabo resničnega gonilnika.



Slika 3: Razdeljeni gonilniki, vir: David Chisnall - The Definitive Guide to Xen Hypervisor - Prentice Hall (2007), str. 20 (Figure 1.5)

3.4 Pomnilniške strani z informacijami

Ob zagonu operacijskega sistema le-ta ponavadi najprej pregleda in pridobi informacije o strojni opremi. Primer tega je poizvedba o razpoložljivem pomnilniku ali pa pregled periferij.

Ker v Xen-u gostujoči OS nima dostop do resnične strojne opreme, moramo te podatke podati na nek drug način. To je z uporabo dveh vrst strani v pomnilniku. Prva se zapiše v gostov naslovni prostor ob zagonu, drugo pa mora eksplicitno zapisati gost.

3.4.1 Začetna stran z informacijami

Ob zagonu se torej v naslovni prostor zapiše stran z informacijami o sistemu (ang. start info page). Najprej jedro preveri, če res teče v pravilni verziji Xen-a. Če je to upešno, prebere, koliko pomnilnika ima na voljo in število CPE. Ostala polja v strani podajo informacije o drugih straneh v spominu. Te strani mora gost zapisati v svoj naslovni prostor z uporabo nadzornih klicev.

Začetna stran vsebuje tudi druge vrednosti. To so:

- določene neobvezne nastavitve domene
- naslov skupne strani, uporabljene za komunikacijo z XenStore in dogodkovni kanal, uporabljen za XenStore.
- odmik in velikost strukture, ki definira konzolo za domeno 0 ali stran v spominu in dogodkovni kanal, ki definira konzolo za domene U

Naslednja tri polja definirajo upravljanje s spominom in ostranjevanje, naslednja dva polja pa opišejo nalaganje modulov. Vsa polja za tem so uporabljena za argumente ukazne vrstice.

3.4.2 Stran s skupnimi informacijami

Stran s skupnimi informacijami vsebuje podatke, ki so dinamično posodobljeni med izvajanjem navideznega stroja. Ob zagonu jih je v svoj naslovni prostor preslikal gostujoči OS. Vsebuje podatke o dogodkih, kot je informacija, da dogodek čaka na dostavo do vCPE. Pove tudi, kateri dogodkovni kanali točno imajo čakajoči dogodek.

Vsebuje tudi podatke o navideznem CPE in določene spremenljivke, odvisne od arhitekture, pomembne pa so tudi spremenljivke, ki so uporabljene za vodenje časa. V tem primeru je treba voditi podatke o realnem in o navideznem času. Vodenje navideznega časa je enostavno: ko gost teče, vsakih 10ms dobi obvestilo o trenutnem času. Vodenje realnega časa je težje - uporabi se začetni čas sistema, trenutni čas sistema in število ciklov, ki se je odvilo od določenega časa v preteklosti.

3.5 Xen in polna virtualizacija

Pri polni virtualizaciji (ang. Hardware virtual machine, HVM) Xen potrebuje pomoč CPE - ta mora torej podpirati virtualizacijo. Xen uporablja orodje QEMU za emulacijo strojne opreme. Emulacija je v bistvu sistemska storitev Domene 0. Za razliko od paravirtualizacije lahko uporabimo kot gostujoči OS tudi Microsoft Windows, ker ne potrebujemo dostopa do izvorne kode jedra.

Če želi gost, ki je virtualiziran na ta način, dostopati do lastnosti, ki so specifične za Xen, uporabi ukaz CPUID in dostopa do specifičnega registra, da lahko dostopa do strani nadzornih klicev. Na ta način izvaja nadzorne klice na podoben način kot paravirtualiziran gost - razlika je v tem, da ne more uporabiti prekinitev za izvedbo nadzornega klica, vendar rabi uporabiti ukaz VMEXIT.

Slabost je nepotrebna zapletenost implementacije celotnega sistema in nižja učinkovitost v primerjavi z paravirtualizacijo.

3.6 Hibridi

Imamo več kot samo polno virtualizacijo in paravirtualizacijo. V zadnjih letih se je virtualizacija namreč razvila v nek spekter. Želimo si prednosti polne virtualizacije in prednosti paravirtualizacije brez slabih lastnostih obeh.

3.6.1 PVHVM

PVHVM dobimo, ko na polni virtualizaciji (HVM) uporabimo v precejšnji meri tudi lastnosti paravirtualizacije za višjo učinkovitost in nižjo zapletenost implementacije. V tem primeru so paravirtualizirani disk, omrežje, prekinitve in časovniki, vendar se še vedno zažene, kot bi se normalni polno virtualizirani stroj itd.

3.6.2 PVH

PVH dobimo, ko na paravirtualizaciji dodamo določene razširitve polne virtualizacije. Tako se znebimo paravirtualiziranega MMU, ki ustvarja težave pri začetnem zagonu jedra. Rajši uporabljamo razširitve, ki so nam omogočene z pomočjo strojne opreme - virtualizacija je še bolj hitra in imamo veliko pomoč strojne opreme.

4 Literatura

David Chisnall - The Definitive Guide to Xen Hypervisor - Prentice Hall (2007)
https://wiki.xenproject.org/wiki/Main_Page
https://github.com/xen-project/
https://xenproject.org/
https://en.wikipedia.org/wiki/Virtualization
https://en.wikipedia.org/wiki/Hardware_virtualization
https://en.wikipedia.org/wiki/Hardware-assisted_virtualization
https://en.wikipedia.org/wiki/OS-level_virtualization
https://en.wikipedia.org/wiki/X86_virtualization
https://en.wikipedia.org/wiki/Hypervisor
https://en.wikipedia.org/wiki/Xen
https://www.usenix.org/system/files/login/articles/105498-Revelle.pdf
https://www.vmware.com/pdf/virtualization.pdf
https://doc.opensuse.org/documentation/leap/archive/42.2/virtualization/html/book.virt/cha.xen.basics.htm