Navidezni stroji

Klemen Remec

2. januar 2025

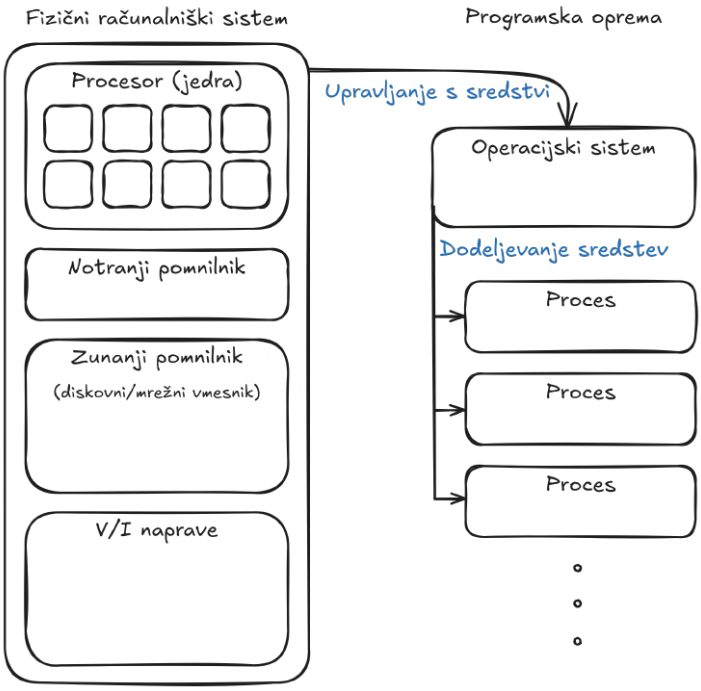
V poročilu so povzeti najpomembnejši koncepti virtualizacije, ki kot tehnika prilagodljivega in učinkovitega upravljanja strojnih virov predstavlja nepogrešljiv element sodobnih računalniških infrastruktur.

1 Uvod

Fizični računalniški sistem vsebuje strojno opremo, kot so:

- procesor, natančneje procesorska jedra
- notranji pomnilnik
- zunanji pomnilnik, dostopen preko mrežnega ali diskovnega vmesnika
- vhodno-izhodne naprave

Operacijski sistem s temi sredstvi upravlja in jih dodeljuje delujočim procesom.

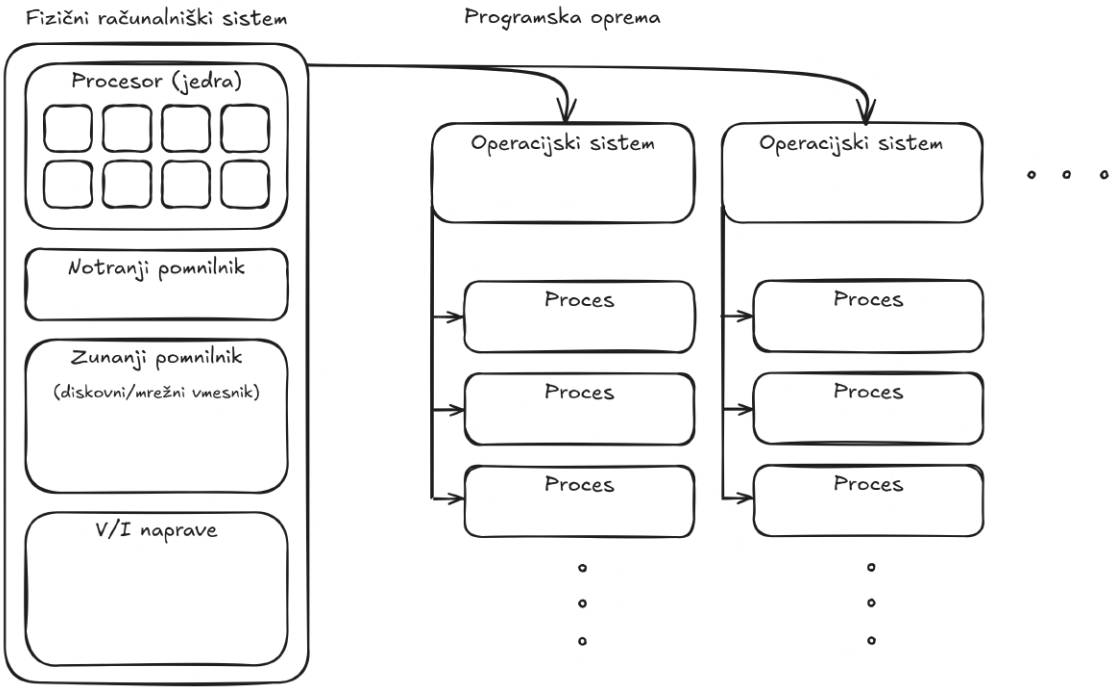


Slika 1: Delovanje računalniškega sistema

Želen rezultat je več navideznih (operacijskih) sistemov, ki dostopajo do istih fizičnih virov preko njihove virtualizacije. [4], [6]

2 Virtualizacija

Virtualizacija: tehnika abstrakcije nižjenivojskih detajlov implementacije fizične strojne opreme za preprost virtualiziran vmesnik višjenivojskim aplikacijam.



Slika 2: Delovanje virtualizacije

[4], [6]

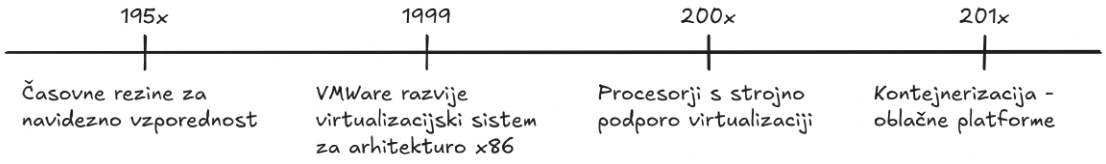
2.1 Zgodovina

Zgodovina virtualizacije sega v 60. leta 20. stoletja, ko so raziskovalci na MIT začeli razvijati koncepte za boljšo izrabo takrat primerljivo slabše strojne opreme. Ena izmed prvih idej je bila uvedba *časovnih rezin* za navidezno vzporedno izvajanje več programov. Ideje o večopravilnosti in virtualnemu pomnilniku so pomembno vplivale na nadaljni razvoj sodobne virtualizacije.

Pomemben prelom v zgodovini virtualizacije je nastopil leta 1999, ko je podjetje VMware razvilo *virtualizacijski sistem za arhitekturo x86*. Ta preboj je virtualizacijo približal širši javnosti in omogočil enostavnejše upravljanje z viri ter večjo izkoriščenost strojne opreme.

Kmalu za tem so se pojavile konkurenčne rešitve, kot so Xen - odprtokodni hipernadzornik, Microsoft je razvil Hyper-V. Proizvajalci procesorjev so začeli izdelovati čipe s *strojno podporo virtualizaciji* - Intel z VT-x in AMD z AMD-V.

Virtualizacija se je še naprej razvijala z vzponom tehnologij za *kontejnerizacijo* kot je Docker, ki omogoča ustvarjanje lahkih in prenosnih virtualnih okolij. V tem obdobju je virtualizacija postala ključna tudi za *oblačne platforme*, saj (v teoriji) omogoča hitro in stroškovno učinkovito širjenje IT infrastrukture. [1], [2]

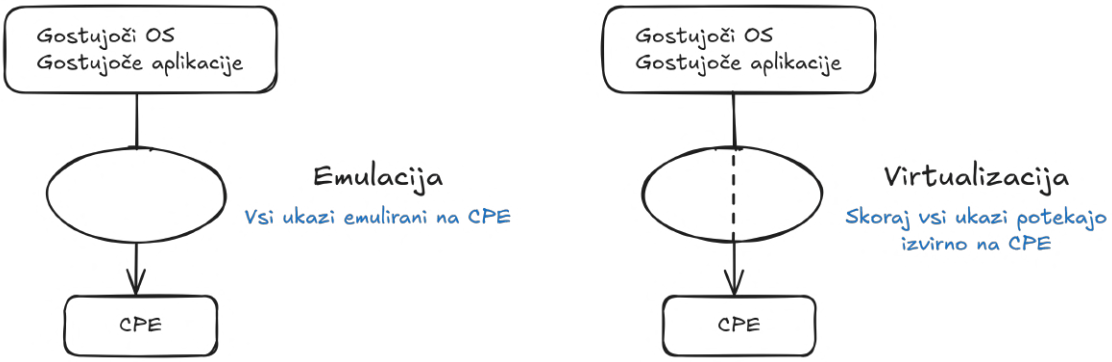


Slika 3: Zgodovina razvoja virtualizacijskih tehnologij

2.2 Emulacija in virtualizacija

Emulacija omogoča gostovanje operacijskih sistemov, zasnovanih za *drugačne arhitekture kot arhitektura gostitelja*. Emulator mora poskrbeti za popolno emulacijo ukazov aplikacij in emuliranega operacijskega sistema, tako da sproti prevaja vsak ukaz gosta v nabor ukazov arhitekture gostitelja.

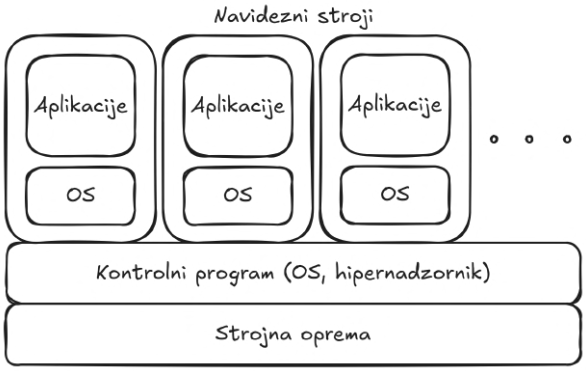
Virtualizacija poskusi tvoriti čim večjo identičnost med naborom inštrukcij, ki jih gostujoči sistem uporablja, in naborom inštrukcij, ki jih pričakuje strojna oprema.



Slika 4: Primerjava delovanja emulacije in virtualizacije

Performančne zmogljivosti virtualiziranega sistema pri izvajanju aplikacij (ki uporabljajo pretežno neprivilegirane ukaze) so veliko boljše od emulacije, primerljive z realnim sistemom. [2], [3]

3 Implementacija



Slika 5: Arhitektura splošne implementacije virtualizacije

3.1 Navidezni stroj

Navidezni stroj (angl. virtual machine, VM): gostujoči (operacijski) sistem, nameščen na programski opremi, ki virtualizira fizično strojno opremo gostiteljskega sistema.

Trajno stanje navideznega stroja je shranjeno v obliki datotek na fizičnem disku:

- navidezni disk predstavlja abstrakcijo fizičnega diska
- konfiguracijska datoteka zagona navideznega stroja
- BIOS datoteka
- posnetki (angl. snapshot)

Kot pri fizičnem računalniku se ob izklopu gostiteljskega (fizičnega) sistema ohrani le stanje na disku, vsebina pomnilnika (aktivno stanje sistema) pa se izgubi. [6]

3.2 Hipernadzornik

Hipernadzornik (angl. hypervisor, virtual machine monitor): programska oprema, ki na gostiteljskem strežniku ustvari virtualno okolje, v katerem nadzira:

- kreiranje navideznih strojev
- nadzor izvajanja navideznih strojev, kar dela na principu, podobnemu delitvi procesorskega časa na časovne rezine med procese
- *upravljanje in dodeljevanje virtualiziranih virov strojne opreme*, kot so procesorska jedra, notranji in zunanji pomnilnik, vhodno-izhodne naprave

Čeprav gostujoči sistem upravlja le s strani hipernadzornika dodeljenim deležem strojnih virov gostiteljskega strežnika, *pričakuje delovanje v okolju z neomejenim dostopom do namenske strojne opreme.* A navidezni stroji istega gostitelja si vire med seboj delijo, mednje jih razporeja hipernadzornik.

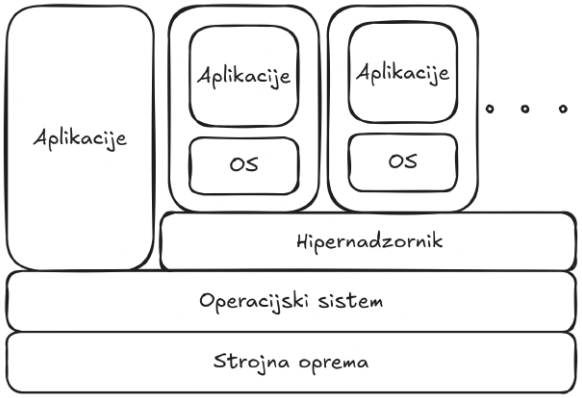
Iz vidika varnosti hipernadzornik razdeli vire tako, da so *viri določenega navideznega stroja izolirani od drugih navideznih strojev*, čeprav navidezni stroji delujejo na isti strojni opremi.

Poznamo več vrst virtualizacij oz. delovanja hipernadzornikov, v nadaljevanju je opisana najpogotejša delitev. [3], [4], [7]

3.2.1 Polna virtualizacija, hipernadzornik tipa 2

Med hipernadzornikom in strojno opremo je nameščen še gostiteljev operacijski sistem, zato *hipernadzornik deluje kot aplikacija znotraj gostiteljevega operacijskega sistema* (njegovo delovanje je odvisno od njega).

Navidezni stroji se ne zavedajo, da tečejo v virtualnem okolju, zato za delovanje ni potrebnih sprememb na gostujočih operacijskih sistemih.



Slika 6: Arhitektura sistema polne virtualizacije

Prednost polne virtualizacije je *popolna abstrakcija strojne opreme*, gostujoči operacijski sistem je tako izoliran od hipernadzornika, gostiteljevega sistema in drugih navideznih strojev.

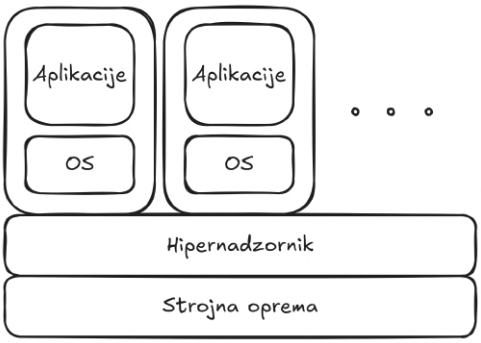
A univerzalna združljivost prinese *dodatne upočasnitve delovanja* v primerjavi z delovanjem na fizičnem sistemu, zato je tak način virtualizacije uporaben za sisteme, ki ne zahtevajo visoke zmogljivosti.

Primeri: Oracle VM Virtual Box, VMware Player, VMware Workstation, Microsoft Virtual PC, QUEMU, Parallels za MacOS, ... [4], [5]

3.2.2 Paravirtualizacija, hipernadzornik tipa 1

Hipernadzornik deluje neposredno na strojni opremi gostitelja.

Navidezni stroji se zavedajo, da tečejo v virtualnem okolju, za kar potrebujejo posebne gonilnike. *Gostujoči operacijski sistem ima prilagojeno jedro*, se zaveda, da teče virtualizirano, in sodeluje s hipernadzornikom. Pri tem je treba omeniti, da gostovih aplikacij ni treba modificirati.



Slika 7: Arhitektura sistema paravirtualizacije

S tem pristopom se izognemo stroškom polne virtualizacije in dosežemo veliko večjo zmogljivost sistema, a smo zaradi tega omejeni le na primerno modificirane operacijske sisteme.

Kasneje opisano delovanje paravirtualizacije se nanaša na delo s hipernadzornikom Xen, ki ga lahko uporabimo v npr. XenoLinux modificiranem operacijskem sistemu z 1-2% spremenjene kode jedra operacijskega sistema.

Primeri: Citrix Xen Server, VMware vSphere in ESXi, Microsoft Hyper-V [4], [5]

3.2.3 Kontejnerizacija

Kontejnerizacija deluje brez hipernadzornika.

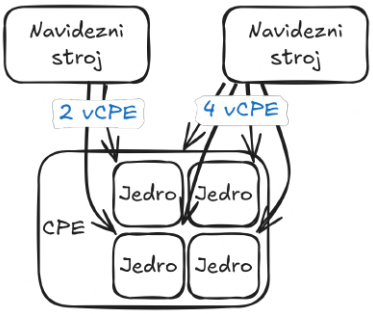
Jedro operacijskega sistema, ki je nameščeno neposredno na strojno opremo, razdeli uporabniški prostor na več izoliranih kontekstov, v katerih delujejo posamezne izolirane aplikacije, ki lahko uporabljajo običajne sistemske klice (ni potrebe po prilagajanju aplikacije za delovanje v takem okolju).

Primer: Docker [4], [5]

4 Virtualizacija CPE

4.1 Polna virtualizacija CPE

Moderni fizični CPE ima več CPE jeder, ki jim v kontekstu virtualizacije prečemo tudi *vCPE*. Vsakemu navideznemu stroju hipernadzornik dodeli določeno število vCPE, s čimer navideznemu stroju da možnost uporabe največ tolikšnega števila vCPE.



Slika 8: Deljenje vCPE več navideznih strojev

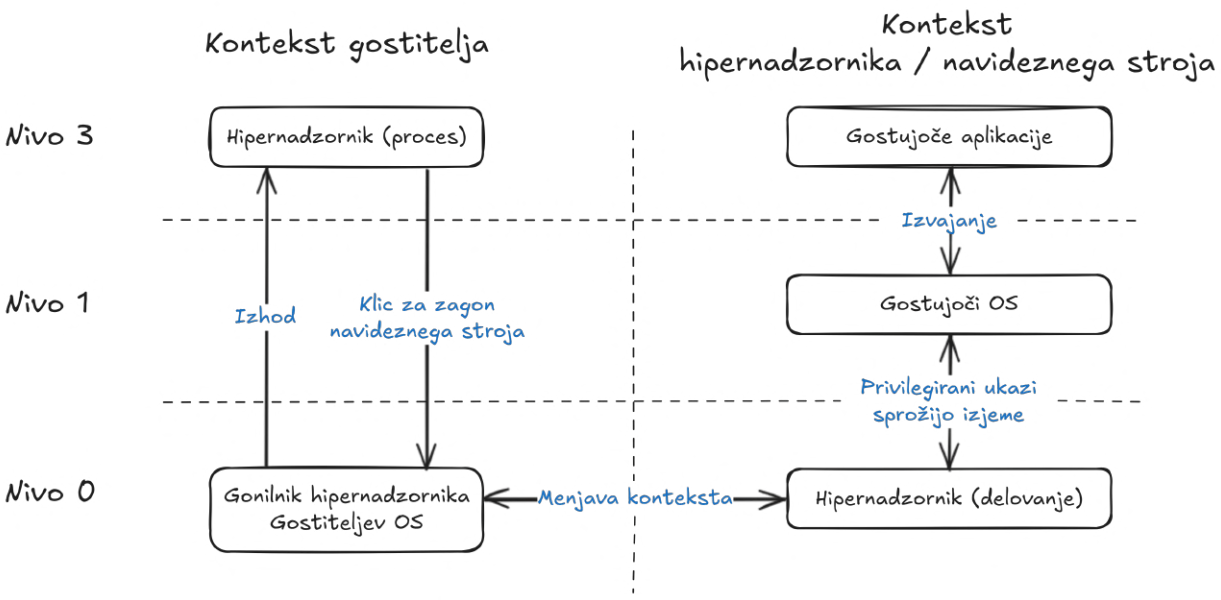
Hipernadzornik ne zagotavlja niti razpoložljivosti takega števila vCPE niti, da bo dodeljene vCPE uporabljal le ta navidezni stroj (lahko si jih deli več navideznih strojev). Posledica tega je *časovno deljenje vCPE*, ki povzroči čakanje enega navideznega stroja, da se razpoložljiva sredstva sprostijo. [6]

4.1.1 Izvajanje ukazov

Ker nabor ukazov x86 ni bil zasnovan z virtualizacijo v mislih, je virtualizacija ukazov otežena.

Arhitektura x86 (Intel-ova različica) pozna *4 nivoje privilegiranosti ukazov* (nivoje od 0 do 3), pri čemer je nivo 0 najbolj privilegiran (izvrševanje ukazov jedra), nivo 3 pa najmanj privilegiran (izvrševanje uporabniških ukazov).

Tabele strani pomnilnika, registri, PC števec in druge kontrolne strukture predstavljajp *kontekst navideznega stroja*, ki je shranjen v pomnilniku in dostopen pa tako iz gostovega kot tudi gostiteljevega naslovnega prostora.



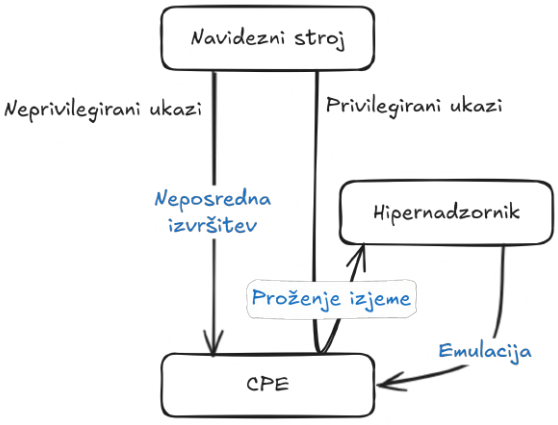
Slika 9: Splošen potek izvedbe ukaza z menjavami konteksta

[3], [7]

4.1.2 Ujemi in emuliraj

Neprivilegirani ukazi (nivoja 3) *se izvajajo neposredno na CPE*, kar pomaga pri učinkovitosti delovanja.

Privilegirani ukazi (sistemski klici, prekinitve, ...) *na CPE prožijo izjeme*, ki jih obravnavamo s skokom na pravi PSP. Ker pa gostujoči operacijski sistem ne deluje v privilegiranem načinu, jih mora *obravnavati hipernadzornik*, tako da prej neuspešno izveden ukaz sam tolmači, uredi in ponovno izvede na CPE.



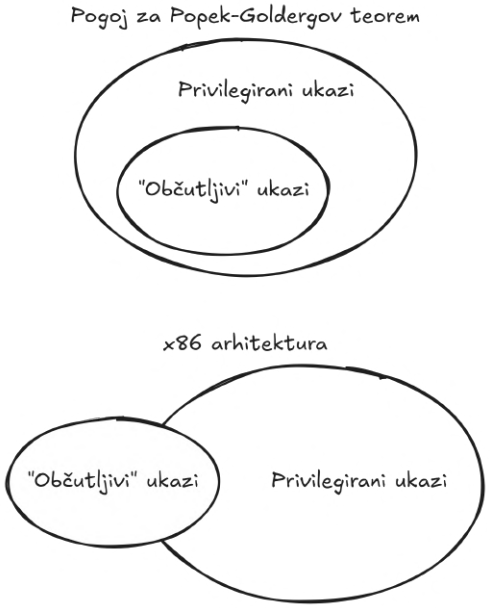
Slika 10: Tehnika izvajanja privilegiranih ukazov "ujemi in emuliraj"

Primer problematike takega pristopa je ukaz `popf`, ki naloži zastavice iz sklada v namenski register `%eflags`.

- Če ukaz izvede jedro operacijskega sistema v privilegiranem načinu, bo v register naložil vse zastavice, vključno s sistemskimi, kot je prekinitvena zastavica (angl. Interrupt Flag), ki hrani stanje omogočenosti prekinitev.
- Če ukaz izvede program v uporabniškem načinu, se ukaz tudi izvede, le da sistemske zastavice pusti nedotaknjene, napake pa ne proži.

V virtualiziranem sistemu v tem primeru ne bi dobili pričakovanih rezultatov, hipernadzornik pa zaradi nesprožitve izjeme ne dobi nobene informacije, da je šlo kaj narobe in da bi moral posredovati. [3]

Popek-Goldbergov teorem govori o pogoju za učinkovito virtualizacijo: vsi občutljivi ukazi (razkrivajo / spreminjajo privilegirano stanje sistema) morajo biti podmnožica privilegiranih ukazov (v uporabniškem načinu prožijo izjemo).



Slika 11: Ukazi Popek-Goldbergovega teorema in x86 arhitekture

Če občutljivi ukazi niso privilegirani, hipernadzornik teh ukazov ne more prestreči, kar onemogoča pravilno izvajanje navideznih strojev. Možni rešitvi:

- v paravirtualizaciji gostujoči operacijski sistem namesto privilegiranih ukazov izvaja hiperklice, ki jih hipernadzornik vedno lahko ujame. Je pa potrebno ponovno napisati operacijski sistem, kar ni vedno možno.
- v polni virtualizaciji se ukazi gostujočega operacijskega sistema nadzorujejo, pri čemer ob privilegiranih ukazih prožimo prekinitev na hipernadzorniku. To prinese dodatno obremenitev.

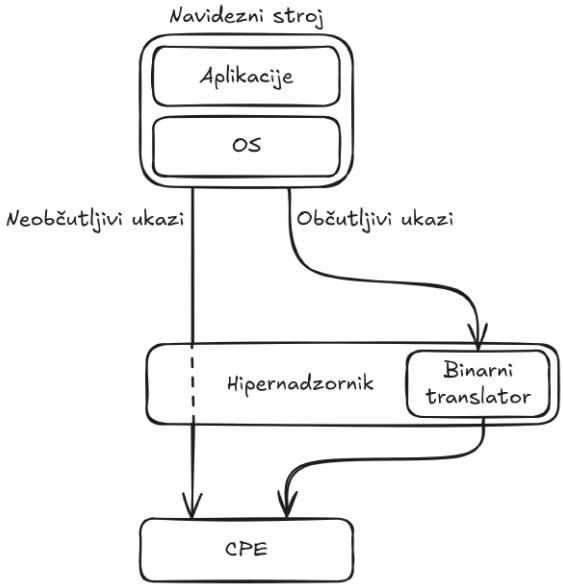
[7]

Optimizacija: binarna/dinamična translacija

4.1.3 Binarna/dinamična translacija

Binarna/dinamična translacija predstavlja programsko tehniko virtualizacije, s katero je *hipernadzornik sposoben prestrezanja za virtualizacijo občutljivih ukazov* ter *ukazov, ki dostopajo do konteksta brez zanašanja na izjeme.*

Binarni prevajalnik deluje dinamično - tik pred izvršitvijo se občutljiv vhodni ukaz prevede v zaporedje ukazov, s katerimi hipernadzornik emulira izvajanje privilegiranega ukaza v navideznem stroju in uveljavi spremembe na navidezni strojni opremi.



Slika 12: Tehnika izvajanja privilegiranih ukazov binarna translacija

Možne optimizacije:

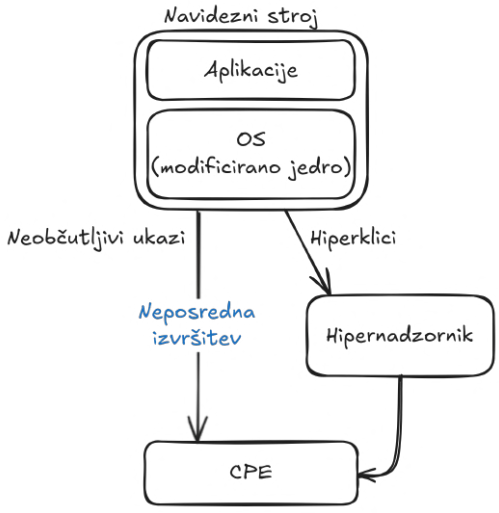
- bločno prevajanje ukazov do naslednjega skočnega ukaza,
- prevedene ukaze se hrani v posebnem medpomnilniku (angl. Translation Cache), ki se uporabi za medpomnjenje prevodov pogosto uporabljenih občutljivih ukazov.

Oba principa polne virtualizacije prinašata dodatno procesiranje v primerjavi s paravirtualizacijo / strojno podprto virtualizacijo. [3], [7]

4.2 Paravirtualizacija CPE

Enako kot pri polni virtualizaciji se tudi pri paravirtualizaciji srečujemo s problemom privilegiranih (na virtualizacijo občutljivih) ukazov.

Gostujoči operacijski sistem je modificiran, da ne izvaja privilegiranih klicev, temveč izvaja *hiperklice* (angl. hypercall), ki delujejo podobno kot sistemski klici - hipernadzornik, ki teče na nivoju 0, ustavi gostujoči operacijski sistem, hiperklic validira in izvrši.



Slika 13: Uporaba hiperklicev v paravirtualizaciji

Za pravilno izvajanje izjem skrbi hipernadzornik preko tabele rutin obdelave izjem (angl. exception handlers). Ob izjemi hipernadzornik identificira izjemo in nadzor prepusti ustrezni rutini.

Izjemi, ki se pojavljata dovolj pogosto, da vplivata na zmogljivosti sistema:

- sistemski klici: hipernadzornik omogoča registracijo posebne rutine za obdelavo, pri kateri preusmeritev preko hipernadzornika ni potrebna
- napake strani: pospešitev ni možna, saj iz registra, ki vsebuje naslov napake strani, lahko bere le ukaz, ki se izvršuje na nivoju 0. Za pridobitev naslova mora tako poskrbeti hipernadzornik (in ga dati na sklad v pomnilnik, iz kjer ga rutina prebere).

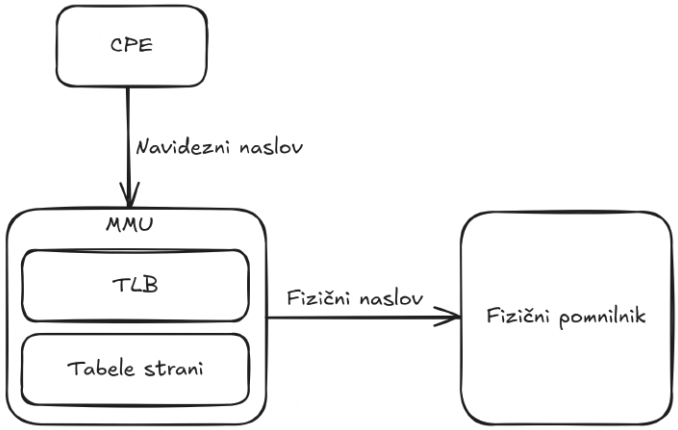
[3], [7]

5 Virtualizacija pomnilnika

5.1 Polna virtualizacija pomnilnika

Navidezni pomnilnik predstavlja mehanizem za *preslikavo več navideznih naslovnih prostorov v fizični naslovni prostor*, pri čemer ima ponavadi vsak proces svoj navidezni naslovni prostor. Arhitektura x86 je navidezni pomnilnik dobila s procesorji 80386, ki so dobili za to namenjen MMU (angl. Memory Management Unit). Za preslikavo arhitektura uporablja hierarhično drevo tabel strani, shranjene v pomnilniku.

Enota MMU uporablja dve strukturi: sprehajalca po tabelah strani (angl. table walker) in medpomnilnik TLB (angl. Translation Lookaside Buffer). Ko nek ukaz dostopa do nekega navideznega naslova, mora sprehajalec *preiskati tabele strani, da dobi ustrezen fizični naslov*, najdeno *preslikavo pa shrani v TLB za pospešitev* iskanja te preslikave v prihodnosti. [3]



Slika 14: Preslikava navideznih v fizične pomnilniške naslove

Vsakemu navideznemu stroju hipernadzornik dodeli del fizičnega pomnilnika, s čimer da navideznemu stroju le možnost uporabe največ toliko pomnilnika.

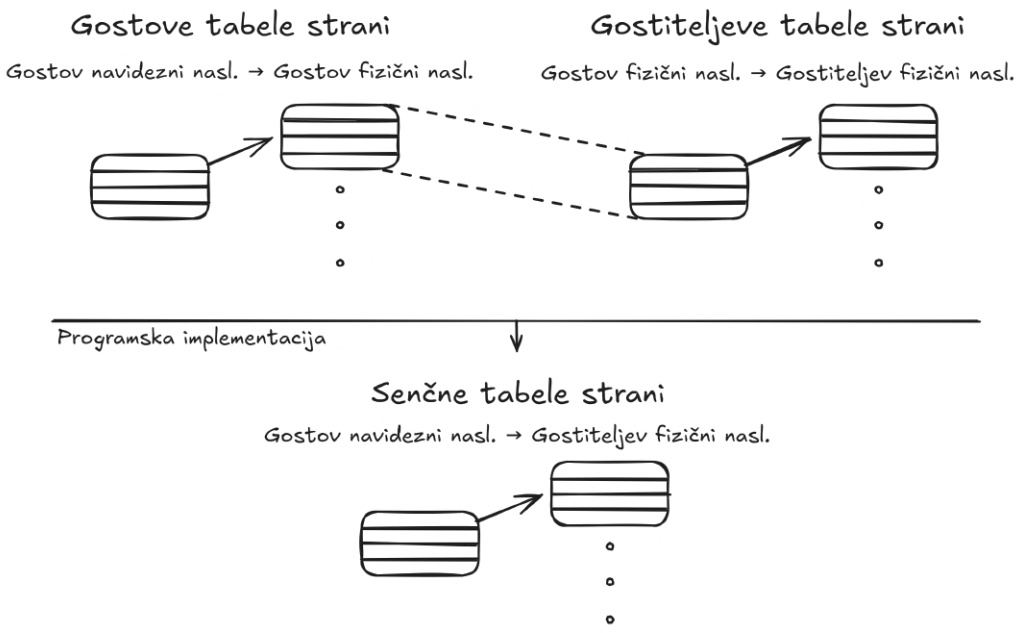
Ker *gostujoči operacijski sistem pričakuje poln naslovni prostor* (npr. 32b gost lahko uporablja 4 GB naslovni prostor), mora hipernadzornik poskrbeti za virtualizacijo navideznega pomnilnika, pri čemer mora poskrbeti, da:

- aplikacije gosta ne dostopajo do pomnilnika, ki jim ne pripada (do gostiteljevega ali hipernadzornikovega pomnilnika ter pomnilnika drugih gostov)
- je gostov navidezni prostor ustrezno preslikan

Gostujoči operacijski sistem hrani preslikovalno tabelo strani: GVA (angl. Guest Virtual Address) → GPA (ang. Guest Physical Address). Hipernadzornik oz. gostiteljev operacijski sistem hrani preslikovalno tabelo strani GPA (angl. Guest Physical Address) → HPA (angl. Host Physical Address)

Za pohitritev dostopa do pomnilnika hipernadzornik uporablja strojni TLB in MMU uporablja dodatne *senčne tabele strani* (angl. shadow page tables). Za vsako tabelo strani, ki jo uporablja navidezni stroj, mora hipernadzornik zgraditi senčno tabelo, v kateri hrani združeno preslikavo iz gostovega navideznega v gostiteljev

fizični prostor (neposredno GVA → HPA). Tako lahko hipernadzornik strojni MMU usmeri neposredno na prave naslove.



Slika 15: Uporaba senčnih tabel strani

Vseeno pa je taka rešitev *počasnejša zaradi potrebnih pregledov in vzdrževanja senčnih tabel* - sploh pri več procesih - kontekstih. Senčne tabele lahko uničimo in ponovno ustvarimo ob vsaki menjavi konteksta (dodatna obremenitev med menjavanjem konteksta), ali pa hranimo več senčnih tabel vseh kontekstov, ki jih moramo ob spremembah popravljati (dodatna obremenitev med spremembami pomnilnika)

Ko se aplikacija v navideznem stroju zapre, njegov operacijski sistem strani virtualnega pomnilnika, namenjene temu procesu, označi kot proste. Ker ne ve, da teče v virtualnem okolju, o tem ne obvesti gostitelja. Zato mora gostitelj na vsake toliko preveriti stanje pomnilnika navideznega stroja in ustrezno sprostiti žasedene-štrani.

Hipernadzornik ne zagotavlja razpoložljivosti določene količine pomnilnika, zaradi česar je možen "over-subscription"

Možna rešitev tega problema je rezervacija pomnilnika - navideznemu stroju se zagotovi del pomnilnika, do katerega ne more dostopati noben drug navidezni stroj. Tak pristop je načeloma neprimeren, saj drugi navidezni stroji ne morejo uporabiti strani v rezervaciji, četudi jih dodeljen navidezni stroj ne uporablja. Zato raje *pomnilnika ne rezerviramo vnaprej*, če navidezni stroji ne uporabljajo (strani) fizičnega pomnilnika, dokler jih ne rabijo.

Druge možne rešitve delovanja s premalo prostega fizičnega pomnilnika:

- hipernadzornik naredi *švapštrani na zunanji pomnilnik*, zgrešitve strani pa jih naložijo nazaj v pomnilnik. Nalaganje strani iz zunanjega pomnilnika je veliko počasnejše.
- *deljenje enakih strani* (enaka vsebina, npr. koda operacijskega sistema, prazne strani). Za tak pristop je potrebno redno primerjanje podobnosti strani (pogosto implementirano s hashingom).

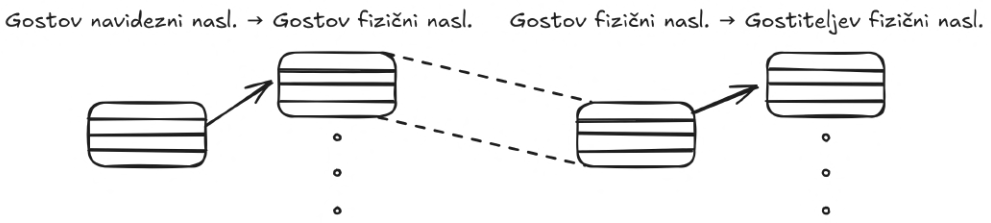
[3], [6], [7]

5.2 Paravirtualizacija pomnilnika

Xen uporablja zgornjih 64 MB naslovnega prostora, ki si ga sicer, podobno kot pri polni virtualizaciji, deli z gostom. S tem preprečimo zmanjšanje učinkovitosti pri prehodu iz jedra v hipernadzornik in nazaj.

Gostujoči operacijski sistem skrbi sam za svoje tabele strani (hrani preslikave GVA → HPA), pri čemer po kreiranju nove tabele strani o tem obvesti hipernadzornika. Od takrat naprej ima le dostop za branje, vsa *nadaljnja ažuriranja validira in opravi hipernadzornik*.

Extended page tables (EPT) predstavlja princip delovanja strojnega MMU, ki se zaveda virtualizacije in obdeluje kazalce na dve tabeli strani.



Slika 16: Uporaba EPT

Vsak dostop do gostove tabele zahteva pregled gostiteljeve tabele ($O(n^2)$) Čeprav uporaba EPT pomeni višjo ceno pri zgrešitvah pri gostih (zaradi dvostopenjske translacije naslova), v splošnem pomeni boljše performančne zmogljivosti v primerjavi z uporabo programske MMU. [3], [7]

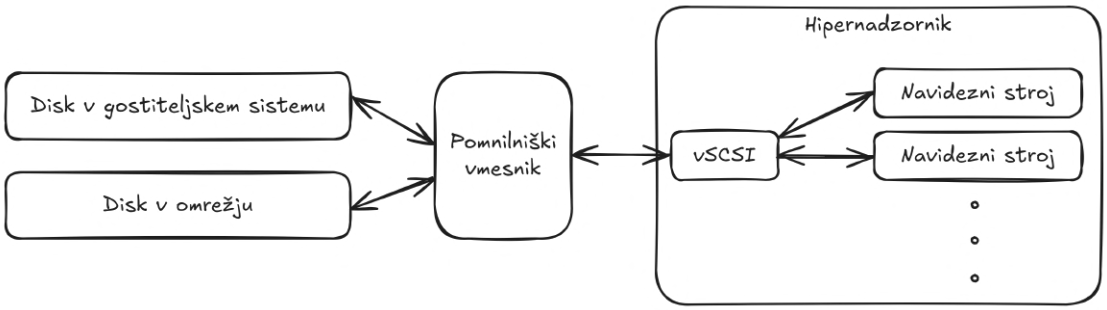
6 Virtualizacija diskovja (zunanjega pomnilnika)

Virtualizira se tudi zunanji pomnilnik, ki je lahko pristoten:

- neposredno *v gostiteljskem sistemu*, do njih hipernadzornik lahko dostopa neposredno
- *v omrežju*, do njih hipernadzornik dostopa preko virtualiziranega mrežnega vmesnika

[8]

vSCSI (angl. virtual Small Computer System Interface) predstavlja del hipernadzornika, ki navideznemu stroju posreduje diskovne operacije (npr. pisanje na disk, branje iz diska, ...) preko t.i. SCSI ukazov na fizične naprave.



Slika 17: Arhitektura virtualizacije diskovja

Ob menjavah lokacij virtualnega diska hipernadzornik kot vmesni člen poskrbi za preklop delovanja brez časa zastoja (angl. downtime). [6]

7 Virtualizacija omrežij

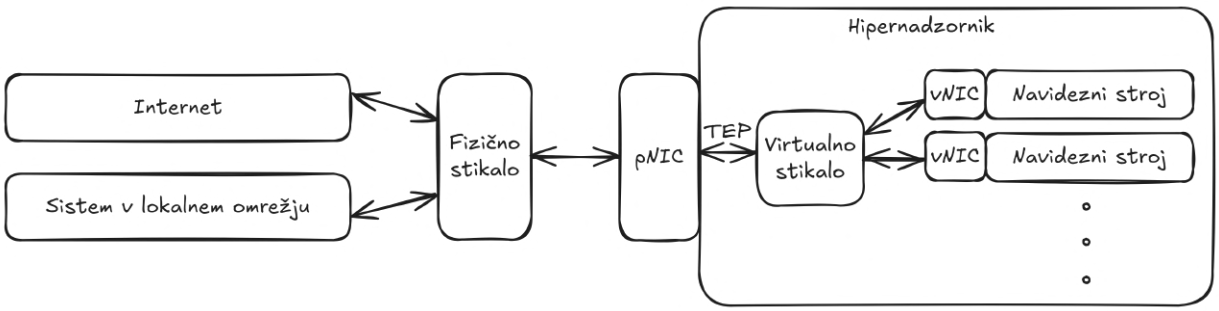
vNIC (angl. virtual Network Interface Card) omogoči, da navidezni stroj vidi gonilnike in NIC kot fizične.

Hipernadzornik omogoča

- medsebojno lokalno komunikacijo med navideznimi stroji preko *virtualnega stikala* - mrežni paketi ne zapustijo virtualnega omrežja
- komunikacijo navzven preko *pNIC*, ki je povezan na *fizično stikalo* - povezan na zunanji usmerjevalnik, druge fizične sisteme, ...

[6]

TEP (Tunelling EndPoint) predstavlja točko stičišč virtualiziranega in fizičnega nivoja, kjer poteka enkapsulacija sporočil za posredovanje preko fizične insfrastrukture do virtualiziranega cilja. [9]



Slika 18: Arhitektura virtualizacije omrežij

Komunikacija poteka neposredno med gonilnikom in gostujočim operacijskim sistemom. Gostitelj omrežnih paketov ne vidi, saj se uporabi gostov MAC naslov, paketi pa se preko DMA zapišejo neposredno v gostov pomnilniški prostor. [6]

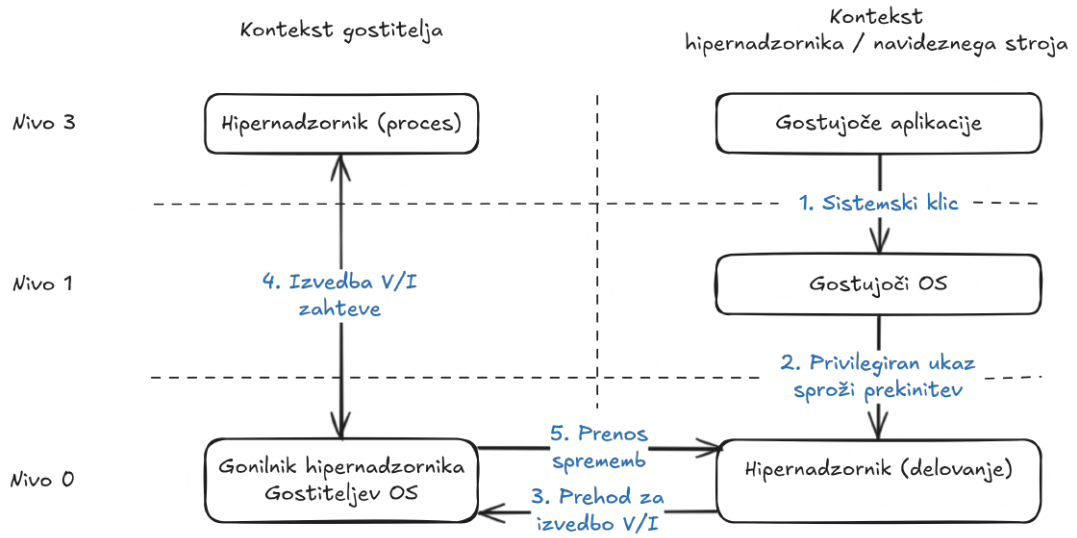
8 Virtualizacija V/I naprav

8.1 Polna virtualizacija V/I naprav

Možna pristopa:

- *Emulacija*: hipernadzornik prestreže ukaze in jih emulira
- *Device passthrough / Direct IO*: navideznemu stroju omogoči dostop do V/I naprav - boljša zmogljivost

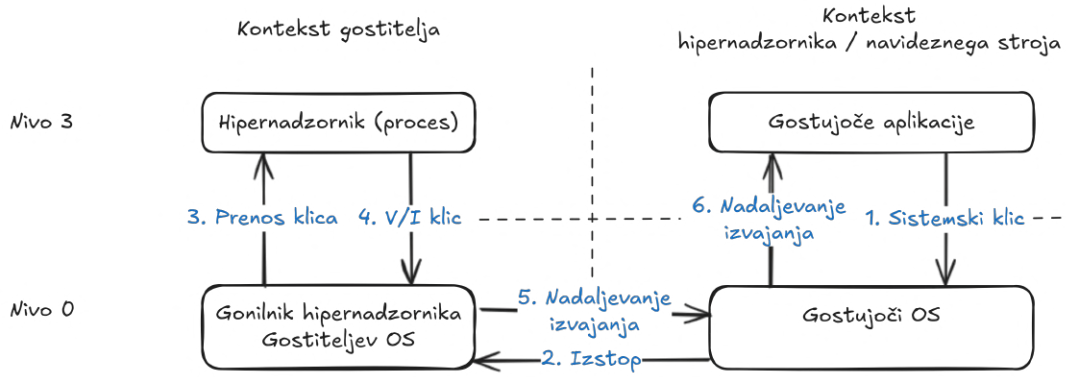
Registri V/I naprav so dostopni kot naslovljiv pomnilniški prostor, z napravo upravljamo preko branja/pisanja na ta naslovni prostor. Naprava ob dogodkih proži prekinitve (namesto pollinga). Moderne V/I naprave preko DMA (Direct Memory Access) podatke najprej prenesejo iz spomina naprave v RAM, pri čemer gonilnik naprave določi fizične naslove za hrambo takih medpomnilnikov. [6], [7]



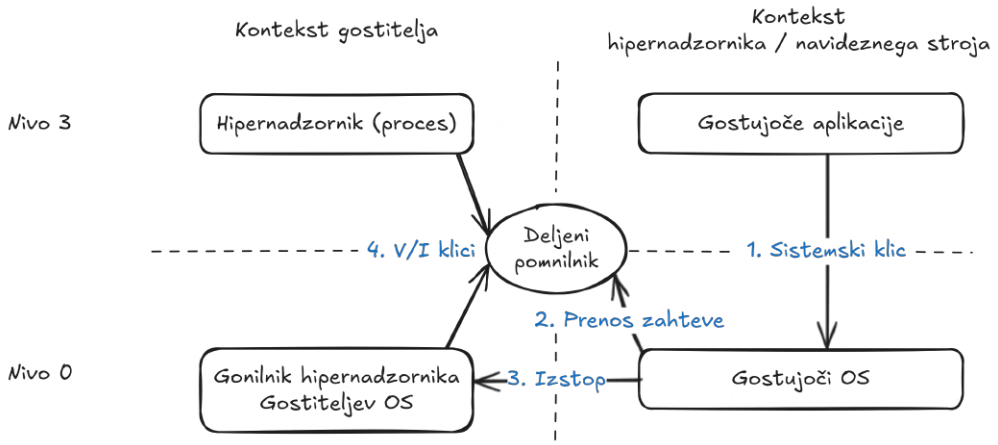
Slika 19: Polna virtualizacija upravljanja V/I naprav

8.2 Paravirtualizacija V/I naprav

Pri paravirtualizaciji V/I naprav ima gost na voljo preproste vmesnike za prenos V/I podatkov, ki delujejo na skupnem/deljenem spomninu. Prenosi potekajo preko hipernadzornika, kar prinese boljšo zmogljivost v primerjavi z emulacijo naprav. [7]



Slika 20: Paravirtualizacija upravljanja V/I naprav



Slika 21: Optimizacija paravirtualizacije upravljanja V/I naprav z deljenim pomnilnikom

9 Strojno podprta virtualizacija

Področje strojne podpore virtualizacije arhitekture x86 ni standardizirano - leta 2005 sta Intel in AMD predstavila vsak svojo rešitev (obe delujeta na način "ujemi in emuliraj", brez uporabe programskih rešitev, kot je binarna translacija). Razširitvi (Intel) VT-x in AMD-V razen nekaj razlik v podrobnostih izvedbe delujeta podobno. [7]

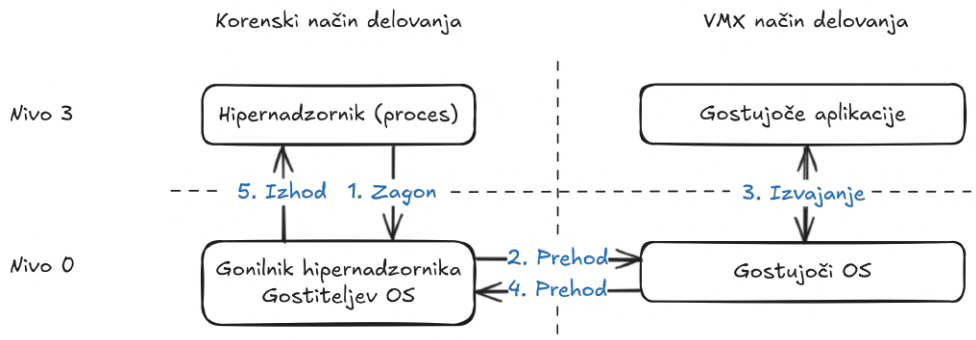
9.1 Prva generacija strojne podpore virtualizacije

VT-x in AMD-V štejemo v prvo generacijo strojne podpore virtualizaciji.

V strojno-podprti virtualizaciji se CPE na zahtevo hipernadzornika (npr. KVM) prestavi v *VMX* (Virtual Machine eXtensions) način delovanja. Upravljanje:

- `vmlaunch`, `vmresume` - vhod v VMX način delovanja (kliče KVM)
- `vmexit` - izhod iz VMX načina delovanja (kliče gostujoči operacijski sistem)

Ob vhodu se spremeni kontekst iz gostiteljskega na gostujoči operacijski sistem, ob izhodu obratno.



Slika 22: Strojno podprta virtualizacija

Nadzorna struktura v pomnilniku hrani kontekste vCPE posameznega navideznega stroja / gostiteljskega sistema, in njegovo nadzorno stanje (o VMX vstopu/izvajanju/izstopu):

- VMCB (Virtual Machine Control Block) pri AMD-V
- VMCS (Virtual Machine Control Structure) pri VT-x

Struktura je dostopna v kateremkoli kontekstu preko posebnih ukazov.

Razširitvi dodajata tudi nov, manj privilegiran, *gostujoči način delovanja*, v katerem izvaja gost svoje ukaze:

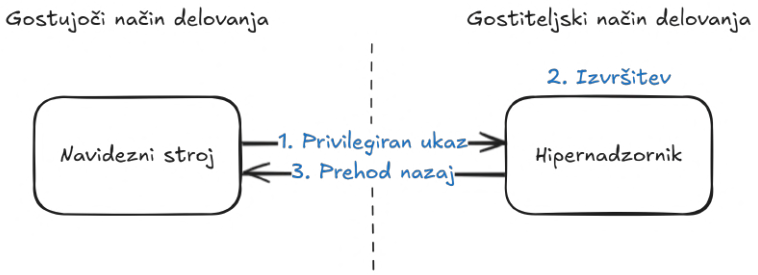
- Guest Mode pri AMD-V
- VMX Non-Root Operation pri VT-x

Na drugi strani hipernadzornik deluje v *gostiteljskem načinu delovanja*

- Host Mode pri AMD-V
- VMX Root Operation pri VT-x

Hipernadzornik z izvršitvijo posebnega ukaza prepusti nadzor gostu, ki teče v gostujočem načinu - takrat shrani trenutno stanje CPE in naloži gostovo stanje iz VMCB/VMCS. Ko gost izvede privilegiran ukaz (ali če pride do izjeme), sproži prehod v gostiteljski način, pri čemer se trenutno stanje gosta shrani v VMCB/VMCS. Hipernadzornik na podlagi teh podatkov ustrezno poskrbi za pravilno izvršitev, nato pa spet sproži prehod v gostov način.

Prehajanje iz gostujočega v gostiteljski način in obratno potrebuje določeno število ciklov procesorja (vsaka kasnejša generacija CPE je to število zmanjševala), kar lahko pri preprostejših operacijah prinese sorazmerno nižjo zmogljivost.



Slika 23: Prehajanje med načinoma delovanja

Prva generacija ni imela podpore za virtualizacijo MMU (npr. za uporabo senčnih strani), zato je moral za *programski virtualni MMU poskrbeti hipernadzornik*, kar je pomenilo še večje število prehodov iz gostujočega v gostiteljski način in obratno. Prva generacija strojne podpore v določenih primerih zato ni dosegala performančnih zmogljivosti polne virtualizacije z uporabo binarne translacije. [7]

9.2 Druga generacija strojne podpore virtualizacije

Druga generacija strojne podpore virtualizaciji je dodala podporo *strojni virtualizaciji MMU*. Tako Intel in AMD sta ponovno predstavila vsak svojo razširitev, ki delujeta na podoben način:

- Intel EPT (Extended Page Tables)
- AMD RVI (Rapid Virtualization Indexing) / NPT (Nested Page Tables)

Dodatna sprememba so oznake na vnosih TLB-ja, ki označujejo kateremu vCPU pripada posamezen vnos. To pomeni, da se *TLB pri prehodu med načini delovanja ne prazni* več, kar dodatno zmanjša upočasnitve. [7]

10 Zaključek

Uporaba virtualizacije prinaša veliko prednosti v primerjavi s tradnicionalnimi sistemi.

Z virtualizacijo lahko na istem strežniku zaženemo več virtualnih sistemov, ki gostijo različne aplikacije, dokler ne dosežemo polne zmogljivosti strojne opreme, kar *zmanjša stroške strojne opreme in energije*.

Datotečna predstavitev virtualnih sistemov omogoča hitro in enostavno premikanje virtualnih strežnikov med gostiteljskimi strežniki, in ustvarjanje varnostnih kopij, kar posledično lahko *skrajša čas izpada storitev in zmanjša tveganje izgube podatkov*.

A ni vse tako črno-belo, saj virtualizacija prinaša tudi določene omejitve.

Pri tradicionalnih strežnikih je administracija osredotočena le na fizični strežnik, medtem ko virtualizacija zahteva vzdrževanje tako gostujočih strežnikov kot gostiteljskega sistema, kar lahko *oteži in poveča stroške upravljanja*.

Ob odpovedi gostiteljskega sistema odpovejo tudi vsi virtualni strežniki, ki tečejo na njem, kar ob neustrezni nastavitvi povečuje *tveganje za večje izpade*. [1]

Literatura

[1] N. Pelko, »Evaluacija platforme za virtualizacijo,« sl, 2013.
[2] B. Bojkovski, »Postavitev ESX 3.5 strežniške infrastrukture za doseganje virtualnega okolja,« sl, 2009.
[3] M. Đukic, »Primerjava zmogljivosti virtualizacijskih tehnologij in tehnologij vsebnikov,« sl, 2016.
[4] M. Pirnat, »Vzorčni primeri konfiguriranja virtualiziranih sistemov v poslovnih okoljih,« sl, 2022.
[5] J. Zor, »Virtualizacija strežniške infrastrukture,« sl, 2018.
[6] One Hour Crash Course, *Introduction to Virtualization*, sl-SI, 2023. spletni naslov: `http://www.youtube.com/playlist?list=PLf7QeYWBebsdlJguAzRR8I1bUIxCHujNo` (pridobljeno 2. 1. 2025).
[7] M. Vutukuru, *CS 695: Virtualization and Cloud Computing*, 2021. spletni naslov: `https://www.cse.iitb.ac.in/~cs695/` (pridobljeno 2. 1. 2025).
[8] IBM Technology, *What is a Hypervisor?* 2021. spletni naslov: `https://www.youtube.com/watch?v=LMAEbB2a50M` (pridobljeno 2. 1. 2025).
[9] IBM Technology, *Virtual Networking Explained*, 2019. spletni naslov: `https://www.youtube.com/watch?v=u0TgGIn2LIM` (pridobljeno 2. 1. 2025).