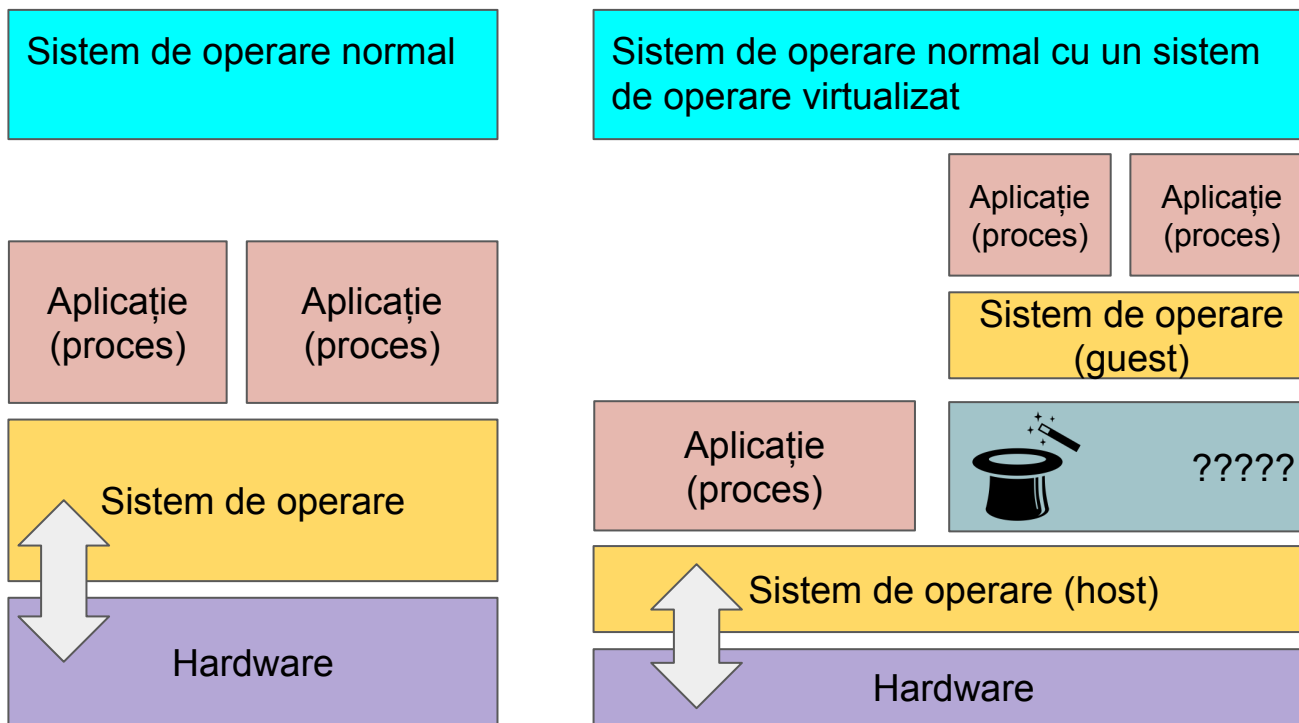


Sisteme de operare Virtualizare

Sorin Milutinovici
sorinmilu@gmail.com

Virtualizare (virtualization = v12n)

Virtualizare - posibilitatea de a rula nucleul unui sistem de operare (modificat sau nu) ca proces în interiorul unui alt sistem de operare.



Provocarea: nucleele sistemelor de operare sunt programate astfel încât să acceseze hardware-ul direct. Procesele simple NU pot accesa hardware-ul direct.

CPU: nucleul conține rutinele de întrerupere și acces la dispozitive

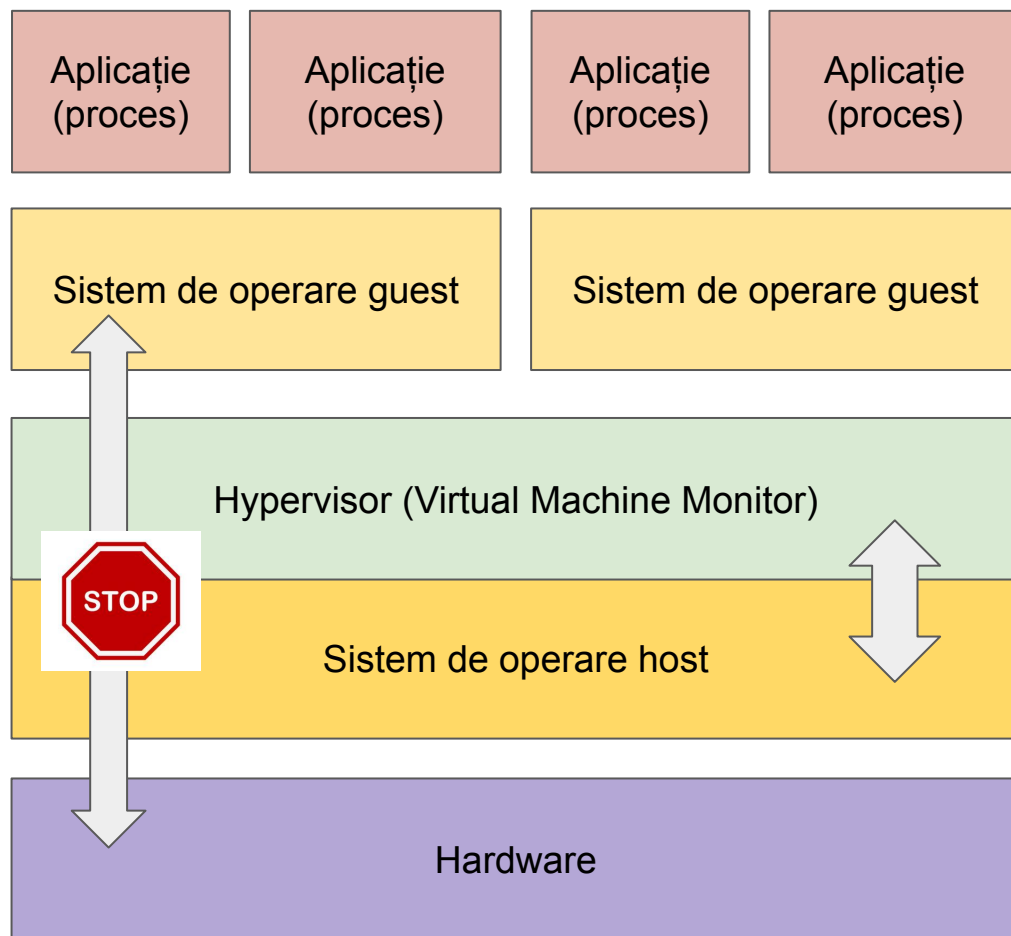
Memorie: nucleul conține funcțiile de management al memoriei virtuale

Dispozitive: nucleul comunică cu dispozitivele.

Cum a apărut virtualizarea?

- Sistemele de operare au apărut pentru că puterea de procesare a computerelor a crescut astfel încât nu mai era rentabil ca un computer să ruleze un singur program.
- Tehnica virtualizării a apărut în anii 1960, odată cu timesharing-ul, cu scopul de a rula mai multe programe în același timp pe marile computere de tip mainframe.
- Virtualizarea a rămas o tehnologie utilizată pe mainframe până în anii 1999 - 2000 când VmWare a lansat sistemul VmWare Workstation.
- VmWare Workstation a fost un produs cheie în dezvoltarea virtualizării, setul de instrucțiuni al procesoarelor x86 era considerat nevirtualizabil până în acel moment ([mai multe detalii](#)).
- VmWare a readus în discuție virtualizarea. Fabricanții de procesoare (Intel și AMD) au răspuns prin introducerea unor instrucțiuni speciale pentru virtualizare în anii 2005 - 2006.

Hypervisor



Primele nuclee ale sistemelor de operare se numeau "supervizoare"

Termenul "Hypervisor" a apărut ca o generalizare a termenului "supervisor"

Hypervisor - un program software, acompaniat de instrucțiuni hardware sau componente firmware care permite emularea unui computer real în spațiul software.

Hypervisor

Tipul 1

Hipervizorul de tip 1 este un sistem de operare minimal însoțit de o serie de instrumente de management al sistemelor de tip guest.

Aplicație (proces)

Aplicație (proces)

Sistem de operare guest

Hypervisor (Virtual Machine Monitor)

Hardware



Microsoft
Hyper-V



Tipul 2

Hipervizorul de tip 2 este un program software conceput astfel încât să fie instalat pe un sistem de operare preexistent.

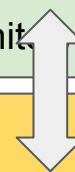
Aplicație (proces)

Aplicație (proces)

Sistem de operare guest

Hypervisor (Virtual Machine Monitor)

Sistem de operare host



Hardware



Microsoft
Hyper-V

Diferența dintre Tipul 1 și tipul 2 ține mai degrabă de modul de livrare a produsului decât de tehnologia utilizată. KVM poate fi obținut sub formă de hipervizor de tip 1 în produse ca Proxmox sau RedHat Virtualization dar și ca hipervizor tip 2 instalat ca simplu pachet într-o distribuție Linux existentă.

Utilizări ale virtualizării



Backup și mutarea aplicațiilor

Un sistem de operare care rulează într-o mașină virtuală este încapsulat în interiorul unui singur fișier. Acesta poate fi mutat de pe un hipervizor pe altul și poate fi pornit fără probleme. Aplicațiile extrem de complexe (sau greu de configurat) pot fi distribuite sub formă de imagini de mașini virtuale.



Izolarea resurselor hardware

Mai multe sisteme de operare rulând în mașină virtuală pot fi limitate în privința resurselor consumate; fiecare poate avea o cantitate fixă de memorie pe care o poate adresa, fiecare poate avea un anumit număr de procesoare, anumite dispozitive pot fi disponibile sau nu fiecărei mașini virtuale.



Medii software diferite

Mai multe sisteme de operare rulând în mașină virtuală pot fi configurate diferit, pot avea versiuni diferite de biblioteci de sistem, pot fi ele însele diferite în funcție de solicitările celor care distribuie aplicațiile necesare.



Securitate

Mai multe sisteme de operare rulând în mașină virtuală determina o izolare completă a tuturor proceselor; nici unul nu poate trece de limita sistemului de operare în care rulează. Pe același hardware, un virus care infectează un sistem guest nu se poate propaga la celelalte.

Nucleele sistemelor de operare sunt programate astfel încât să acceseze hardware-ul direct. Procesele simple NU pot accesa hardware-ul direct.

CPU: nucleul conține rutinele de întrerupere și acces la dispozitive

Memorie: nucleul conține funcțiile de management al memoriei virtuale

Dispozitive: nucleul comunică cu dispozitivele.

Sistemul de protecție al procesorului

Procesorul conține un registru special (registrul de control) în care nucleul sistemului de operare îi specifică "inelul" curent.

Anumite instrucțiuni nu se pot executa decât în "ring 0".

Ex: X86

LGDT – Load Global Descriptor Table

LLDT – Load Local Descriptor Table

LTR – Load Task Register

LIDT – Load Interrupt Descriptor Table Register

MOV (to and from control registers only) – Control Register is used to change the CPU behavior, related to interrupt control, addressing mode, paging and coprocessor control.

MOV (to and from debug registers only) – Debugging programs, and use to set debugging controls.

LMSW – Load Machine Status Word

CLTS – Clear Task Switched

INVD – Invalidate Cache

WBINVD – Write Back and Invalidate Cache

INVLPG – Invalidate TLB Entry

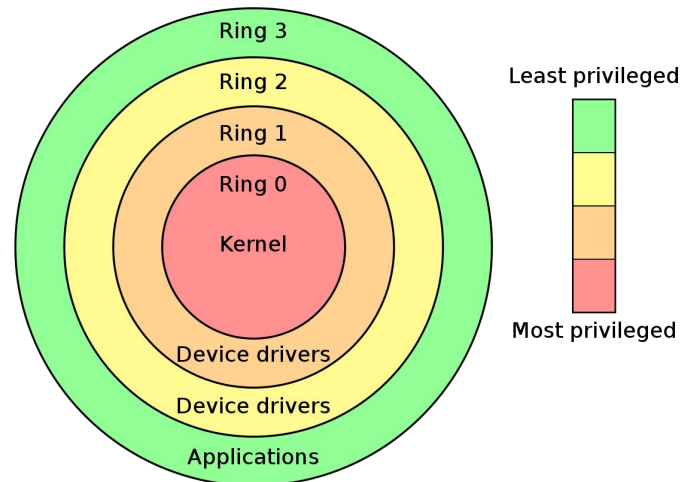
HLT – Halt; halts the CPU and causing it to wait for the next interrupt.

RDMSR – Read From Model Specific Register

WRMSR – Write to Model Specific Register

RDPMS – Read Performance Monitoring Counters

RDTSC – Time Stamp Counter



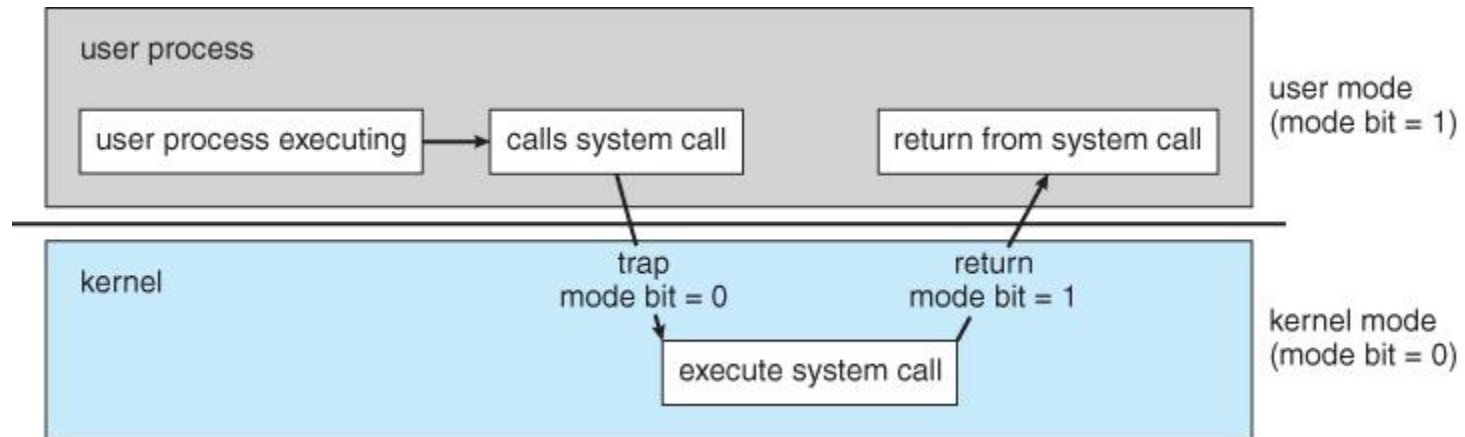
https://en.wikipedia.org/wiki/Protection_ring

Nucleul sistemului de operare "guest" va încerca să execute aceste instrucțiuni dar nu va putea.

User mode - kernel mode

Procesorul obligă la cel puțin două moduri de rulare a instrucțiunilor: user mode sau kernel mode.

Kernel mode este utilizat doar de nucleul sistemului de operare: în kernel mode pot fi rulate instrucțiuni cu acces direct la hardware. În kernel mode se poate executa orice instrucțiune a procesorului, se poate adresa direct memoria și elementele de management al acesteia etc.



În timpul rulării unui proces normal, orice eveniment extern (întrerupere) determina mutarea controlului de la procesul care rulează la nucleul sistemului de operare. Aceasta mutare însă nu implica un context switch ci doar execuția unei funcții care se găsește în memoria virtuală a procesului, în spațiul protejat

User mode - kernel mode

1

Un proces normal are nevoie de o acțiune pe care o poate executa doar nucleul și apelează un apel de sistem (syscall). Acesta are ca primă consecință inițierea unei întreruperi.

2

Întreruperea conține o serie de instrucțiuni (handler) care, în acest caz, mută procesorul în inelul 0 și trimite controlul către nucleu.

3

Nucleul determină dacă procesul are permisiunea de a executa apelul de sistem.

4

Dacă da, apelul de sistem va fi executat de către nucleu

5

După execuția apelului de sistem, nucleul va iniția mutarea procesorului în inelul 3

6

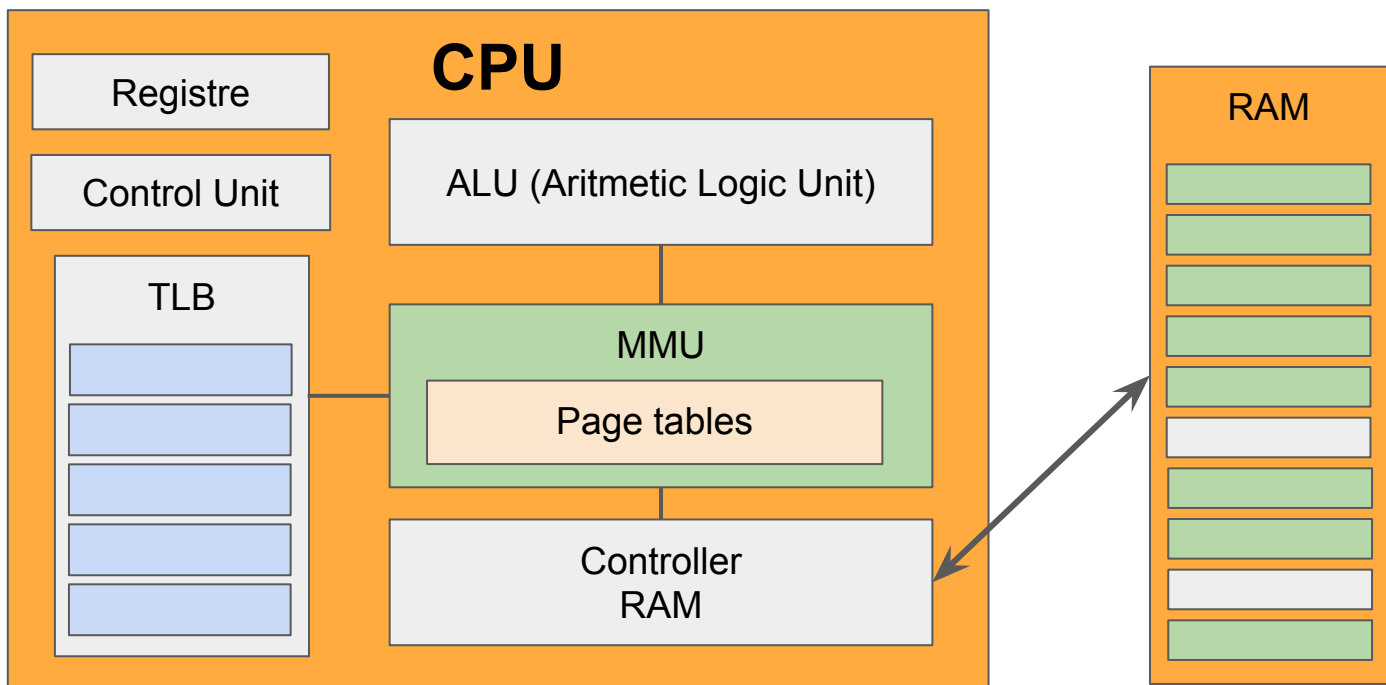
Nucleul returnează controlul către procesul din userspace.

Accesul la memorie

Memoria fiecărui proces este izolată cu ajutorul unui component numit Memory Management Unit care menține relația dintre adresele reale din RAM și adresele virtuale ale fiecărui proces. Când un proces accesează o adresă de memorie, aceasta este "tradusă" din adresa spațiului procesului respectiv în adresa reală din RAM. Acest proces are un cache numit TLB (translation Lookaside Buffer). TLB este golit la fiecare context switch

Translation lookaside buffer (TLB) este un sistem de cache pentru adresele de memorie care au fost accesate recent. TLB poate ține doar o parte dintre perechile de adrese virtuale/reale și permisiunile acestora.

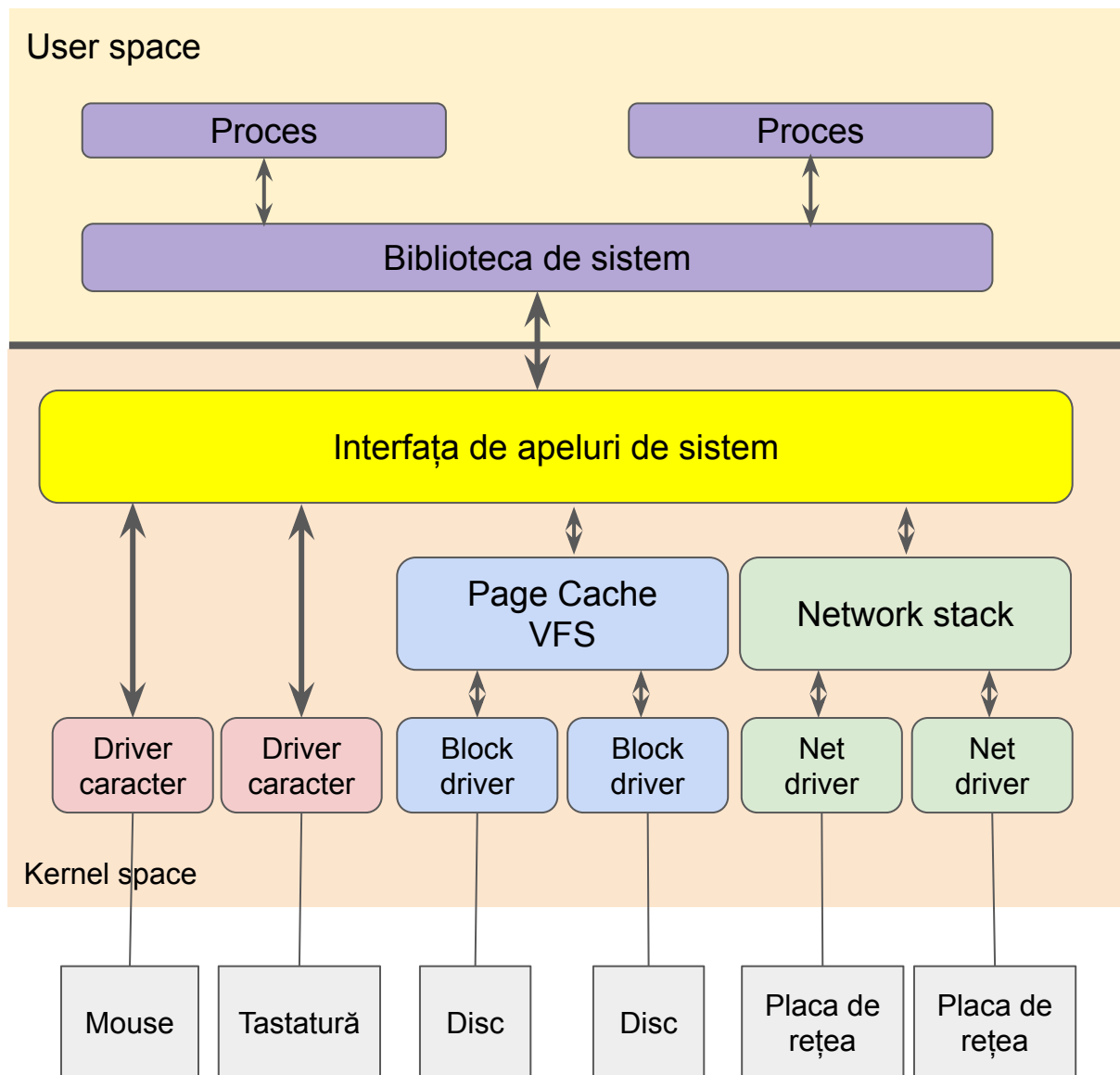
Tabelele de pagini ale tuturor proceselor sunt ținute în RAM și modificate la fiecare context switch în MMU



Este nevoie de un sistem care să permită izolarea paginilor de memorie pentru procesele sistemului de operare "guest".

Hipervizorul trebuie să mențină propria listă de mapare a paginilor (shadow paging).

Interacțiunea cu dispozitivele periferice

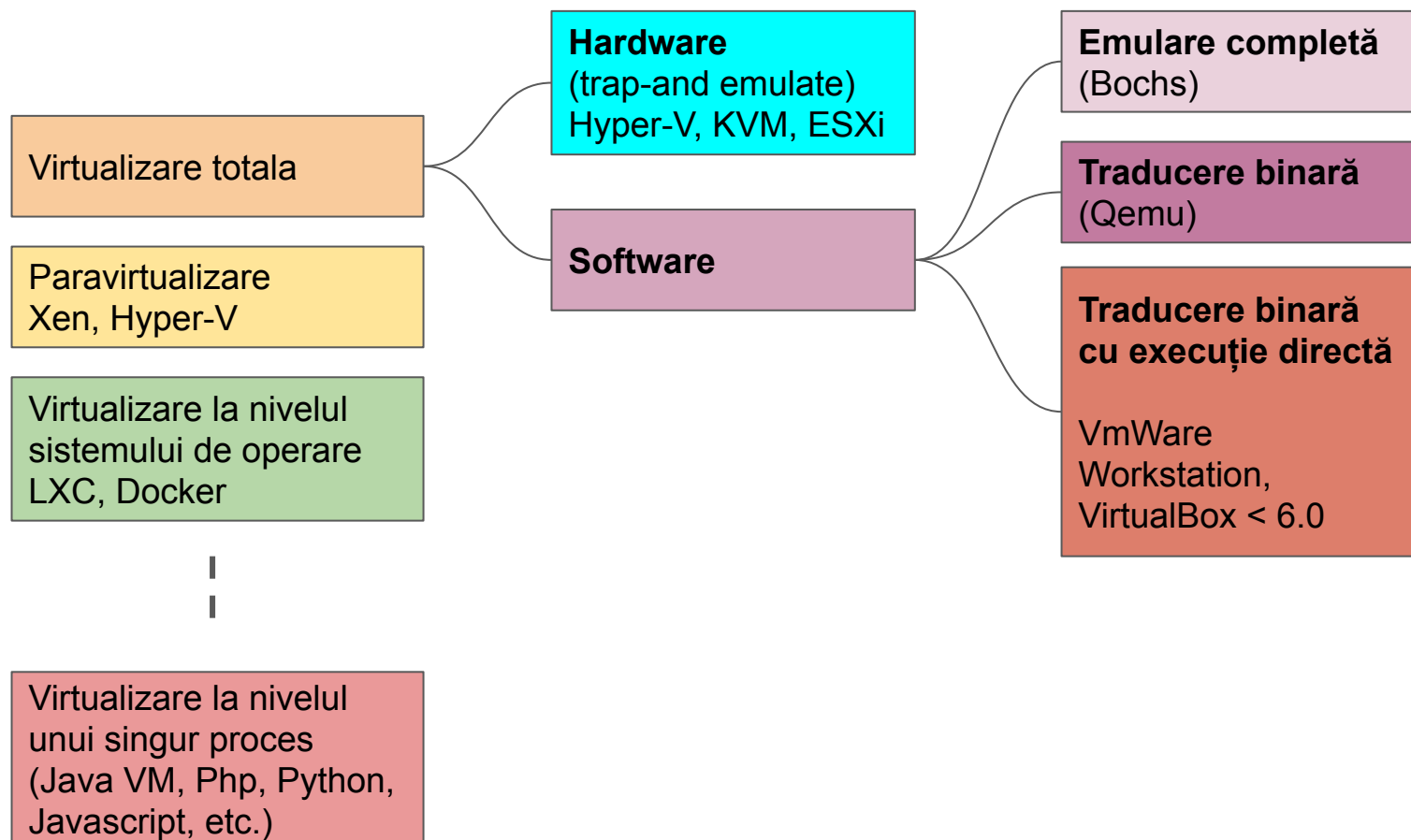


Accesul la dispozitive se face prin intermediul apelurilor de sistem. Majoritatea apelurilor de sistem solicita procesorului instrucțiuni care nu pot rula decât în inelul 0.

Apelurile I/O trebuiesc catalogate și corect rutate către procesele sistemului de operare guest.

Nivele de virtualizare

Virtualizarea unui computer presupune transformarea hardware-ului acestuia în software. În final, virtualizarea conduce la izolarea unor procese și a mediului lor de operare, într-o serie de zone diferite care se pot găsi împreună pe același computer.



Principalele nivele de virtualizare

Virtualizare totală

Virtualizarea totală înseamnă că platforma de virtualizare este capabilă să ruleze sistemele de operare guest complet nemodificate.

Nemodificate înseamnă că nucleul sistemului de operare nu este modificat. În multe situații este nevoie de instalarea unor drivere pentru dispozitivele exportate de hypervisor pentru obținerea unor performanțe crescute sau caracteristici îmbunătățite.

Paravirtualizare

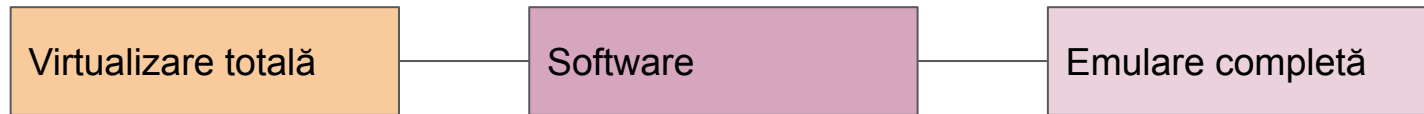
Se acceptă mici modificări ale sistemului de operare guest, în condițiile în care aplicațiile care rulează în acesta rămân nemodificate. Modificările afectează în general apelurile de sistem care vor fi modificate în apeluri către hypervisor. Sistemul guest rulează pe baza unui nucleu modificat.

Virtualizare la nivelul sistemului de operare

Virtualizarea la nivelul sistemului de operare reprezintă o izolare cât mai bună a anumitor seturi de procese față de alte seturi de procese, toate rulând în același sistem de operare, comunicând cu același nucleu. Aceasta metodă se numește și metoda containerelor.

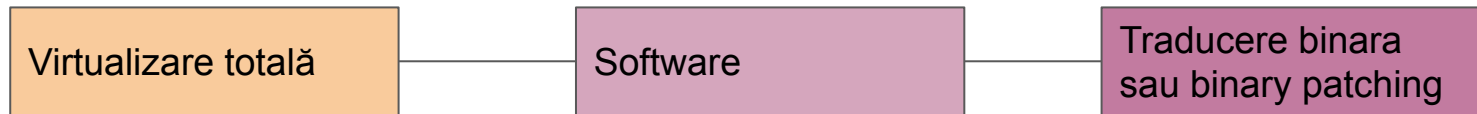
Virtualizare totală

Virtualizare totală - CPU - Emulare



- Emularea presupune captarea de către hipervizor a **tuturor instrucțiunilor** pe care sistemul de operare guest le trimite către procesor și **asocierea fiecăreia dintre ele** cu o funcție software.
- Instrucțiunile sunt captate și executate una câte una. Procesul este simplu de implementat dar foarte lent.
- Nu tot timpul o singură instrucțiune din arhitectura sistemului de operare guest corespunde unei singure instrucțiuni în arhitectura sistemul host.
- Emulatoarele emulează de obicei un set de instrucțiuni. De aceea, sistemele de operare care pot fi instalate în interiorul unui emulator sunt acelea pentru care emulatorul este pregătit - cele pentru care poate traduce complet instrucțiunile.

Virtualizare totala - CPU - traducere binara



➤ Captează instrucțiunile înainte de a fi executate.

- O mare parte din instrucțiuni pot fi executate direct (ex: calculele aritmetice)
- Cele care nu pot fi executate direct sunt înlocuite cu un alt set de instrucțiuni
- Traducerea se face la nivelul blocurilor de instrucțiuni, nu una cate una

➤ X86 aduce o serie de dificultăți în plus.

- Instrucțiunile și datele sunt amestecate, traducătorul trebuie să știe să le separe
- Segmentele de cod pot avea lungime variabila.

➤ În mare măsură procedeul de traducere binară seamănă cu compilatorul JIT folosit în Java

Virtualizare totala - CPU - trap-and-emulate



Sistemul de operare guest rulează ca un proces normal, alături de celelalte procese

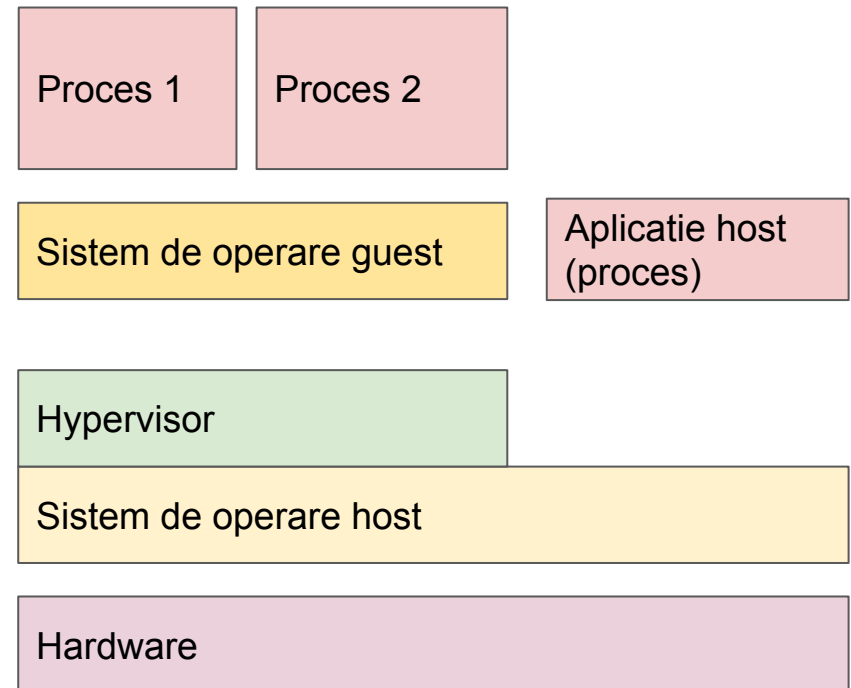
➤ La un moment dat, sistemul de operare guest va dori să execute o operație în kernel mode ex: să scrie ceva în memoria unui dispozitiv.

➤ Pentru aceasta, nucleul sistemului guest va executa un apel de sistem

➤ Când nucleul sistemului guest încearcă aceasta, procesorul va vedea că apelul de sistem vine de la o aplicație care rulează în mod utilizator, va genera o eroare și va preda controlul sistemului host.

➤ Sistemul host va prelua controlul și îl va ceda hipervisorului care va inspecta apelul de sistem, îl va executa și va trimite către guest răspunsul așteptat.

➤ Acest mod de execuție se numește "trap and emulate". Acest mod permite ca sistemul guest să utilizeze procesorul în același mod în care l-ar utiliza dacă ar fi instalat direct.



Virtualizare totala - CPU - trap-and-emulate

- Pentru ca o arhitectură să fie “virtualizabilă” prin metoda trap-and-emulate trebuie să fie proiectată astfel încât **toate** instrucțiunile sensibile să fie executate în mod privilegiat. Astfel ele pot fi captate și emulate de hypervisor.
- Arhitectura x86 conținea inițial 18 instrucțiuni considerate sensibile dar neprivilegiate (ex. acces la registrele procesorului). De aceea, virtualizarea sistemelor de operare compilate pentru arhitecturi x86 (ex: Pentium) nu puteau folosi sistemul “trap-and-emulate” - hipervizorul trebuia să capteze toate instrucțiunile sistemului guest și să le verifice.
- În anii 2005 - 2006 atât AMD și Intel au început să vândă procesoare care conțineau un set extins de instrucțiuni care permitea rezolvarea acestor probleme.

AMD-V

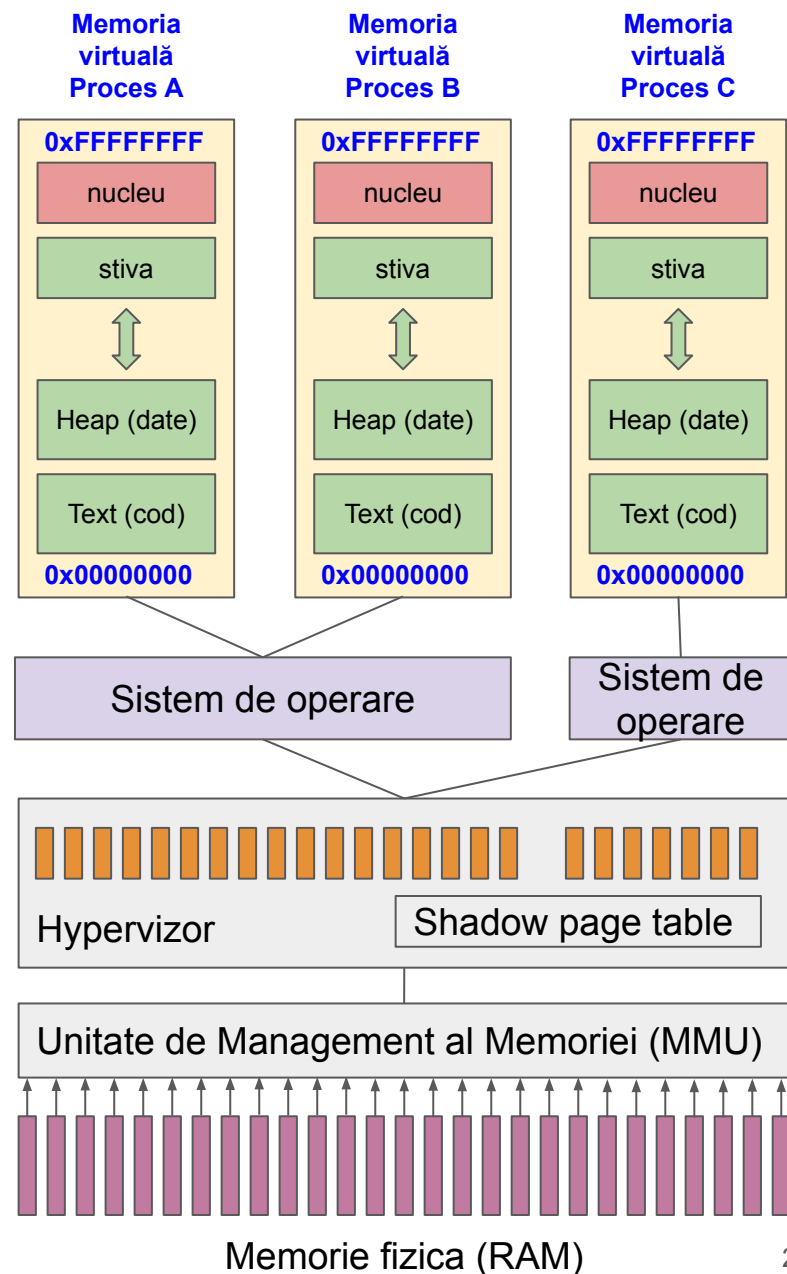
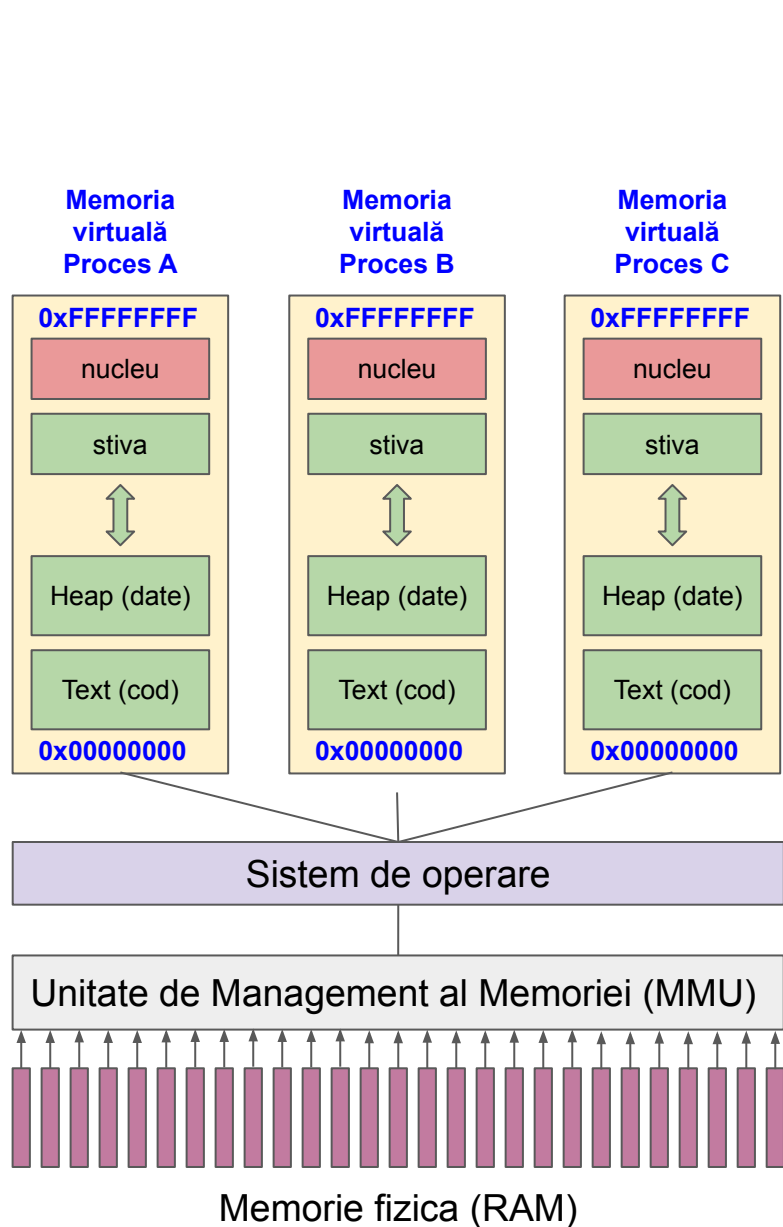
<http://developer.amd.com/wordpress/media/2012/10/NPT-WP-1%201-final-TM.pdf>

Intel VT-x

<https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/virtualization-tech-converged-application-platforms-paper.pdf>

Noile instrucțiuni permit execuția în inelul 1 (VMX Root Mode) - acolo vor rula nucleele sistemelor de operare guest. Astfel, arhitectura X86 devine complet virtualizabilă prin metoda trap-and-emulate.

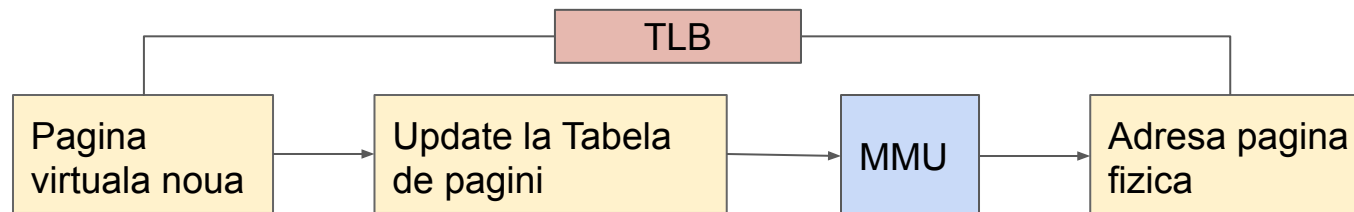
Virtualizarea totală - memorie



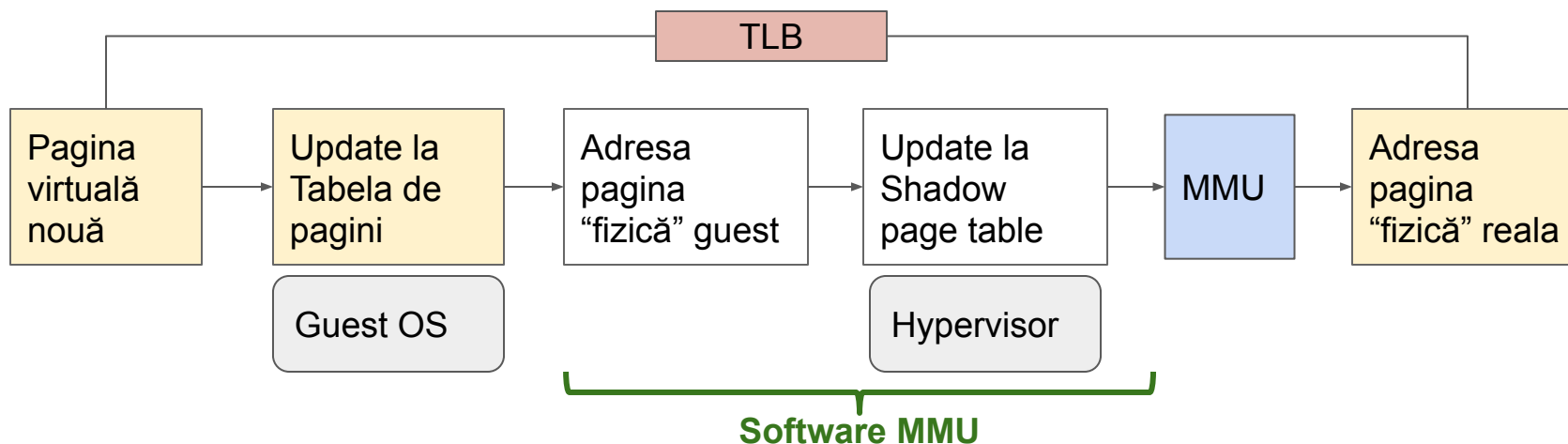
Virtualizarea totală - memorie

Procesul X încearcă să aloce o nouă pagină de memorie.

➤ Sistem de operare instalat direct



➤ Sistem de operare instalat în mașina virtuală

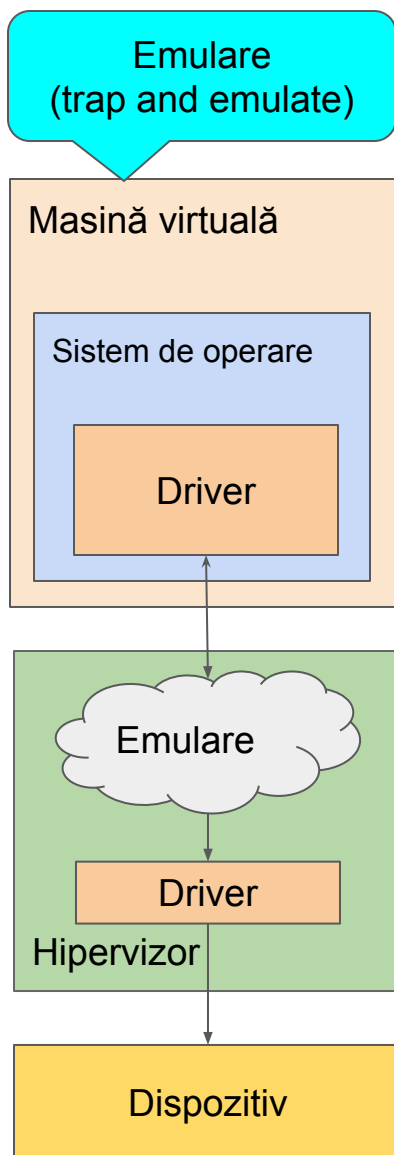


Update la tabela de pagini în Guest OS este o instrucțiune privilegiată, care va determina întreruperea procesului și cedarea controlului către hypervisor. Hypervisorul va modifica propriul tabel de pagini și apoi va introduce maparea între pagina virtuală a procesului care rulează pe guest și pagina reală în TLB.

Virtualizarea totală - memorie

- Modificările făcute în “shadow page tables” ridică probleme de performanță. Mai mult, performanța scade foarte repede cu creșterea încărcării mașinii virtuale - încărcarea mai mare implică de obicei mai multe modificări în tabelele de pagini ale sistemelor guest, ceea ce conduce la mai multe modificări în tabelele shadow.
- Hipervizorul devine extrem de complicat.
- Orice tranziție între o mașină virtuală și alta determina golirea completă a TLB-ului (ca orice mutare de la un proces la altul), inclusiv intrările din TLB care aparțin hipervizorului.
- Intel Nehalem (2008) a introdus două îmbunătățiri:
- **Virtual Procesor Identifier** (intrările din TLB pot fi marcate cu un procesor virtual și astfel TLB-ul nu trebuie golit complet)
- **Extended Page Table** - suport hardware pentru maparea adreselor din sistemele de operare guest direct către adrese reale.
- AMD Opteron seria Barcelona (2007): **Nested Page Tables** (echivalent cu Extended Page Table) și **Address Space ID's** pentru partiționarea TLB-ului, echivalent cu Virtual Procesor Identifier.

Virtualizarea totală - dispozitive



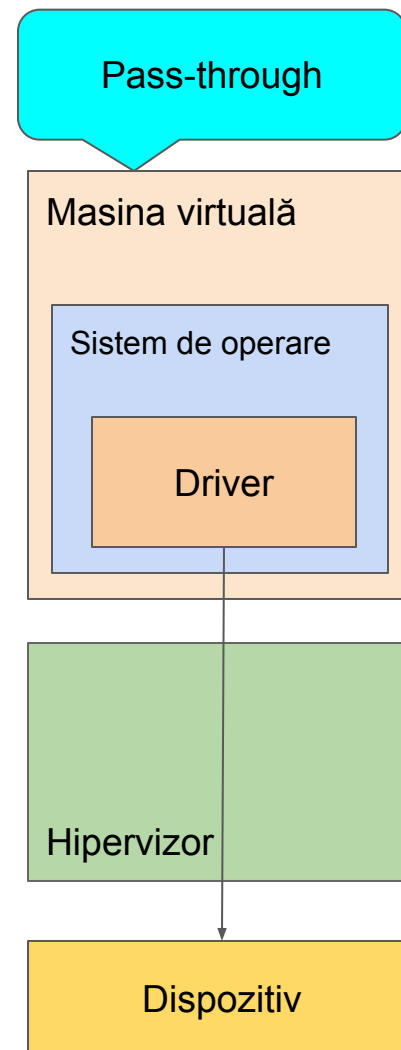
Passthrough presupune expunerea completă a unui dispozitiv către un anumit sistem de operare guest.

Utilizarea dispozitivului este exclusivă iar performanța este aproape de cea nativă.

Încep să apară tehnologii pentru a îmbunătăți accesul direct: IO/MMU, VT-d

Sistemul Guest încearcă să acceseze un anumit dispozitiv.

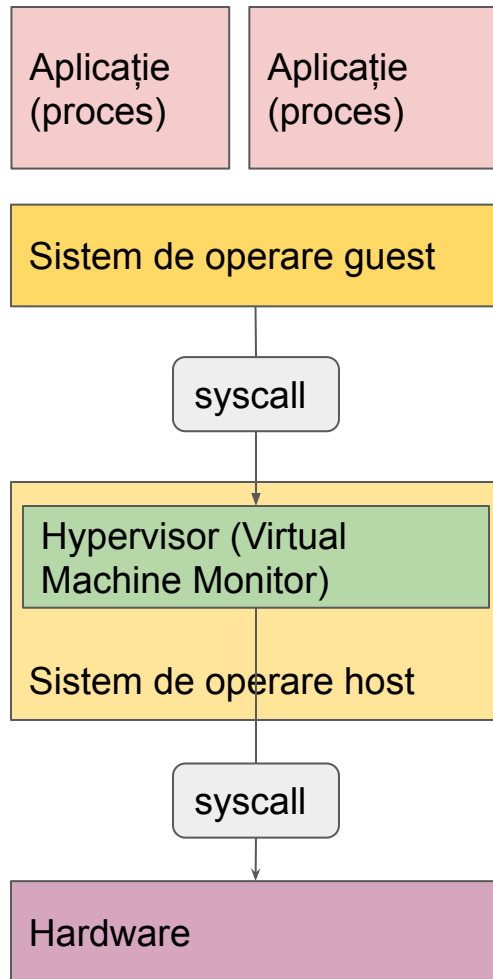
Accesarea dispozitivului se face prin intermediul unei instrucțiuni privilegiate. Sistemul guest rulează în mod user. Accesarea dispozitivului va genera o eroare și va returna controlul hipervizorului care va executa instrucțiunea pe dispozitivul real (dacă există) și va răspunde sistemului guest.



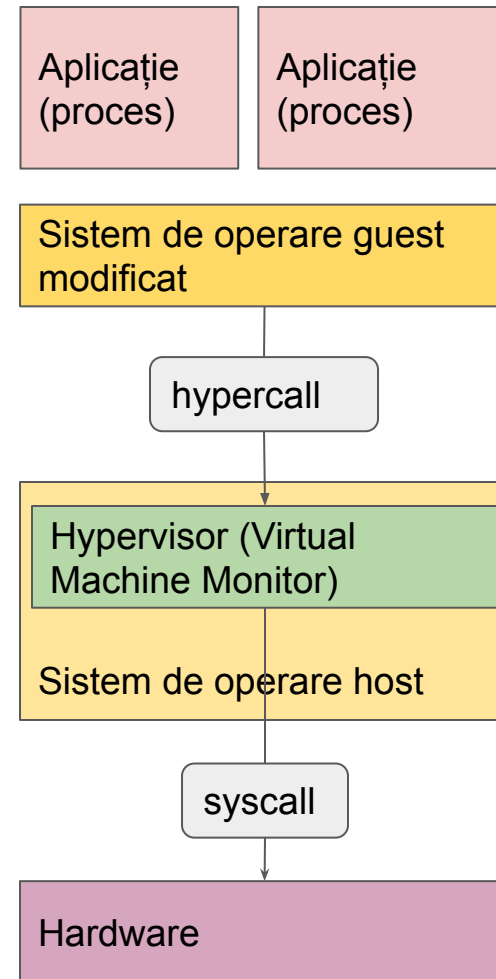
Paravirtualizare

Paravirtualizare

Virtualizare totala



Paravirtualizare



Paravirtualizare

- Tehnica paravirtualizării promite o performanță mult mai bună decât cea a virtualizării totale și, pentru procesoarele lipsite de suport hardware pentru virtualizare, aceasta era adevărat.
- Înlocuirea apelurilor normale de sistem cu apeluri speciale, pe care hipervizorul le putea interpreta fără ezitare, făcea ca timpul pierdut în evaluarea și rescrierea codului binar, problemele legate de managementul memoriei și a dispozitivelor să fie mult mai mic.
- Un alt avantaj al paravirtualizării este faptul că driverele din nucleul sistemului de operare guest sunt și ele modificate și apelurile către acestea treceau prin hipervizor fără nevoia de a fi captate și emulate. Aceasta face ca dispozitivele hardware prezente pe computerul pe care rulează hipervizorul să poată fi accesate de către sistemele de operare guest, fără să fie nevoie de “inventarea” unor dispozitive virtuale.
- Principalul proiect care utilizează paravirtualizarea este proiectul xen (<https://www.xenproject.org>).

Paravirtualizare - xen



CPU

Nucleul sistemului de operare guest este modificat astfel încât să NU mai trimită nici o instrucțiune sensibilă către procesor. Toate apelurile de sistem sensibile sunt înlocuite cu apeluri către hipervizor.



Memorie

XEN expune adresele hardware către sistemele de operare guest. Acestea vor construi și menține un tabel de pagini care va fi real - va conține adrese reale, fără să fie nevoie de traducerea suplimentară din tabelele shadow. Toate modificările solicitate de către guest în tabelul de pagini se fac prin apeluri către hipervizor - acesta va face întreg managementul memoriei.



Driveri

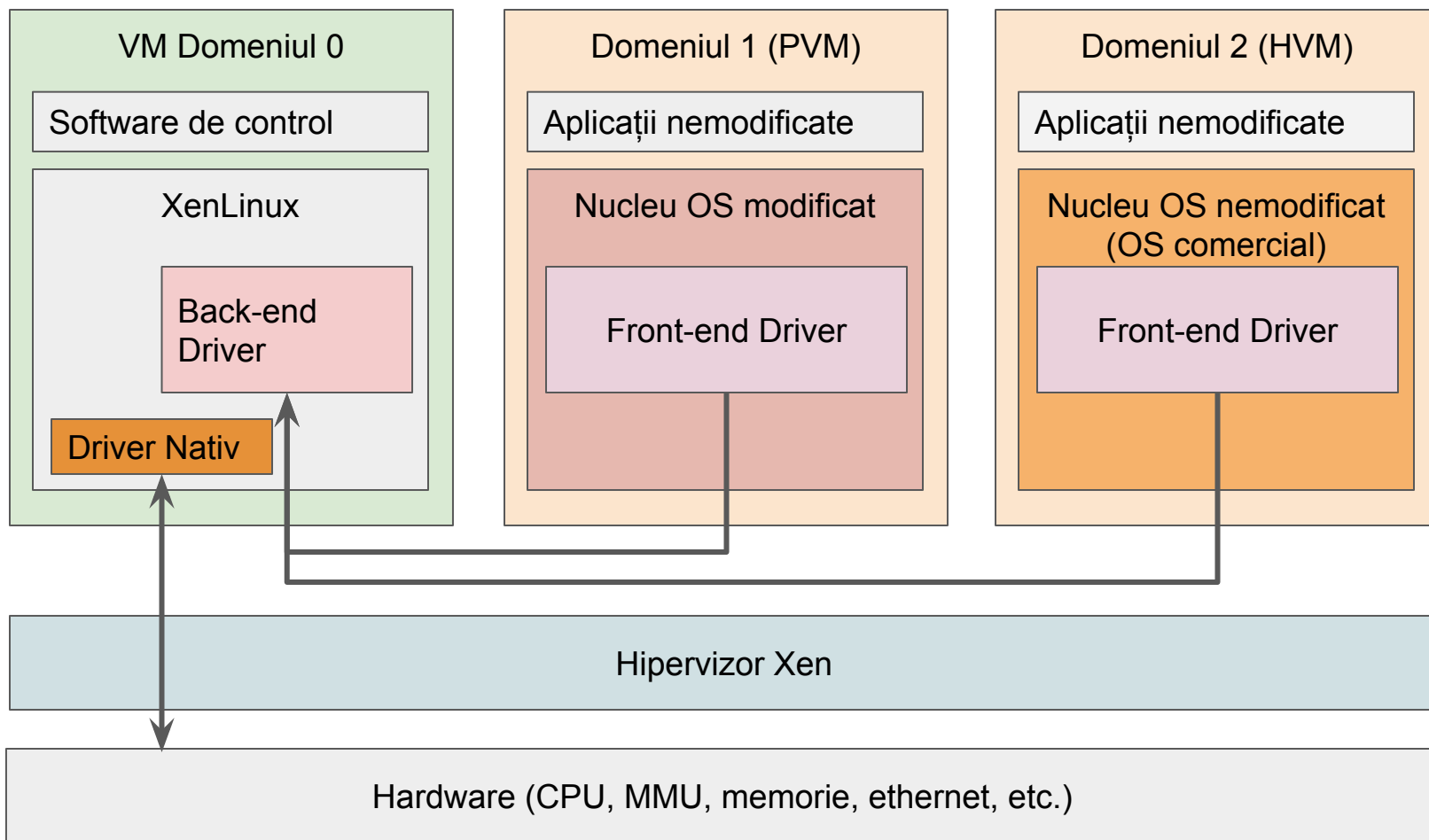
Pentru a putea utiliza aproape orice dispozitiv hardware real, XEN își instalează un sistem de operare guest special, numit Domeniul 0. Acesta are drivere reale pentru componentele hardware existente. Orice sistem de operare este perfect echipat pentru a construi o abstracție software a dispozitivului și pentru a permite mai multor procese să interacționeze cu acesta fără coliziuni.

Utilizând memoria partajată, hipervizorul XEN va utiliza această capacitate a sistemului din Domeniu 0 și o va exporta către sistemele guest din domeniile următoare, care vor fi echipate cu drivere modificate, care vor accesa de fapt driverele din sistemul principal (Dom 0).

Arhitectura tipică a unui sistem XEN

XEN a reușit să depășească în performanță toate soluțiile de virtualizare totală (vmware, qemu), până la apariția instrucțiunilor care permiteau virtualizarea totală a procesoarelor x86.



Acum, XEN poate avea ca sistem guest și sistemele ale cărui nucleu nu este modificat.



XEN - Paravirtualizare parțială

XEN permite acum utilizarea mai multor nivele de paravirtualizare, unele cu suport hardware.

În acest moment, aproape nimeni nu mai folosește nuclee modificate: paravirtualizarea se folosește aproape exclusiv pentru drivere

	<div><div></div>Optimal performance</div> <div><div></div>Scope for improvement</div> <div><div></div>Poor performance</div>				
	<div>Disk and Network</div> <div>Interrupts, Timers</div> <div>Emulated Motherboard, Legacy boot</div> <div>Privileged Instructions and page tables</div>				
Fully Virtualized (FV)	VS	VS	VS	VH	HVM mode/domain
FV with PV disk & network	P	VS	VS	VH	
PVHVM	P	P	VS	VH	
PVH x86 	P	P	P	VH	PV mode/domain
PVH ARM v7+ 	P	VH	P	VH	
Fully Paravirtualized (PV)	P	P	P	P	

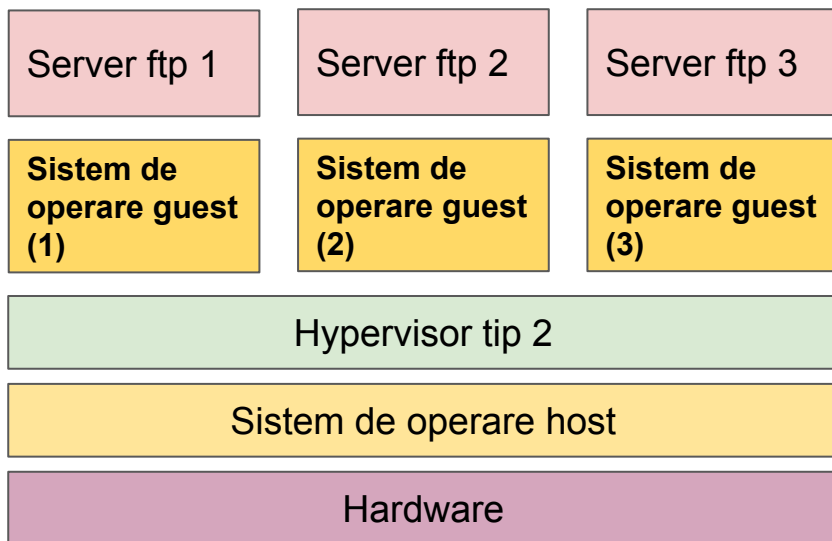


Serviciile de cloud computing oferite de Amazon folosesc XEN, cu tranziție spre KVM.

Virtualizare la nivelul sistemului de operare

Virtualizare la nivelul sistemului de operare

Exemplu: Avem nevoie de trei servere ftp care deserveșc trei grupuri diferite de clienți. Nici unul dintre clienți nu trebuie să poată vedea fișierele celorlalte grupuri. Traficul nu va fi foarte mare și am dori să le instalăm pe același server (hardware).



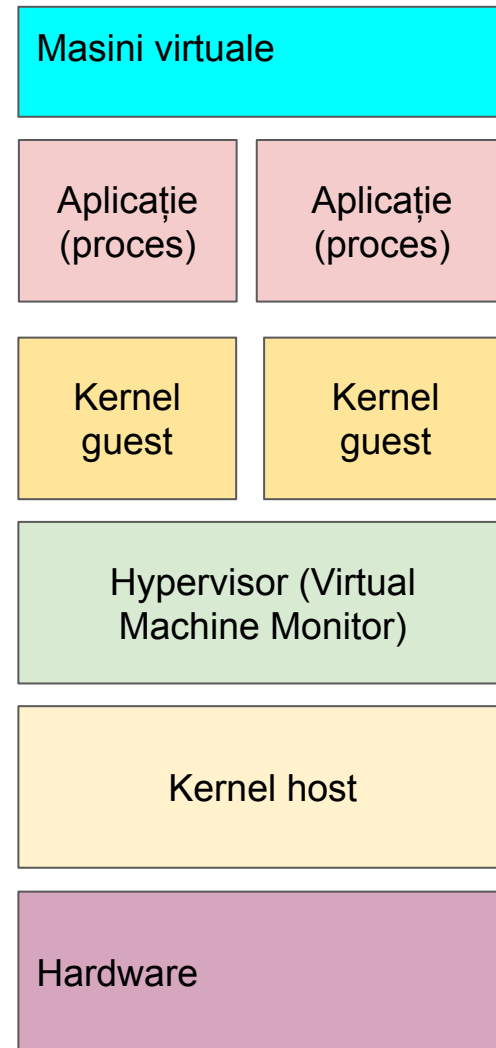
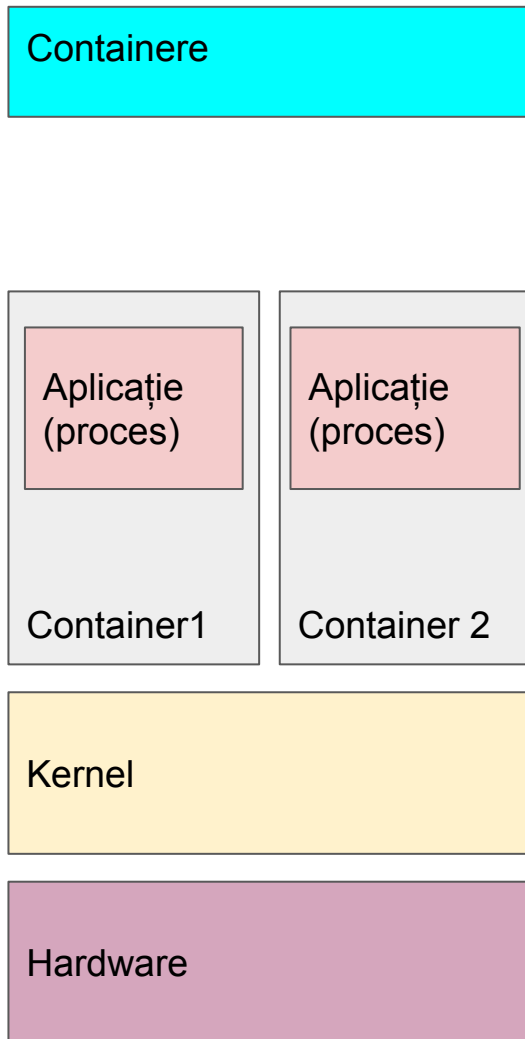
Nu e nițel cam ... mult? Patru sisteme de operare (probabil aceleași), un hipervizor, doar pentru a izola câteva fișiere și câteva liste de utilizatori?

Pentru FTP aceasta problema se rezolva mult mai simplu, prin sistemul de hosturi virtuale pe care serverele FTP le au incluse în cod. Fiecare adresa ftp, chiar dacă ajunge pe același server (același IP) va fi trimisa într-un alt director, cu alta lista de utilizatori.

Virtualizare la nivelul sistemului de operare

- Majoritatea programelor software open-source au fost portate pe mai multe platforme, de aceea nu mai este necesar sa amestecăm mai multe sisteme de operare conduse de un hipervizor.
- Sistemele de operare oferă o serie întreagă de instrumente pentru izolarea proceselor (sistemele de operare împreună cu hardware-ul izolează memoria folosită de fiecare proces, de exemplu).
- Având în vedere dificultatea virtualizării platformei x86 (înainte de apariția instrucțiunilor suplimentare), au fost căutate soluții alternative, nu numai la problema tehnologica (ex: XEN) ci și la problema de management (izolarea aplicațiilor mai multor clienți pe același hardware).
- Sistemele de izolare se numesc “sisteme de containere”. Cuvântul “containerizare” este utilizat și rostit fără rușine de către cei care lucrează în domeniu.
- Tehnologia containerelor a ajuns în acest moment să aibă efecte foarte apropiate de cele ale virtualizării, izolând absolut tot mai puțin kernelul.
- Pe același kernel se pot instala și rula mai multe “distribuții de Linux”, fiecare cu propriile fișiere, propria memorie ocupată, propriile dispozitive și adrese, complet separate una de cealaltă, cu excepția faptului că se bazează pe același nucleu.
- Aceasta se întâmplă datorită stabilității ABI-ului nucleului Linux.

Virtualizare la nivelul sistemului de operare



Virtualizare la nivelul sistemului de operare

Avantaje ale containerelor:

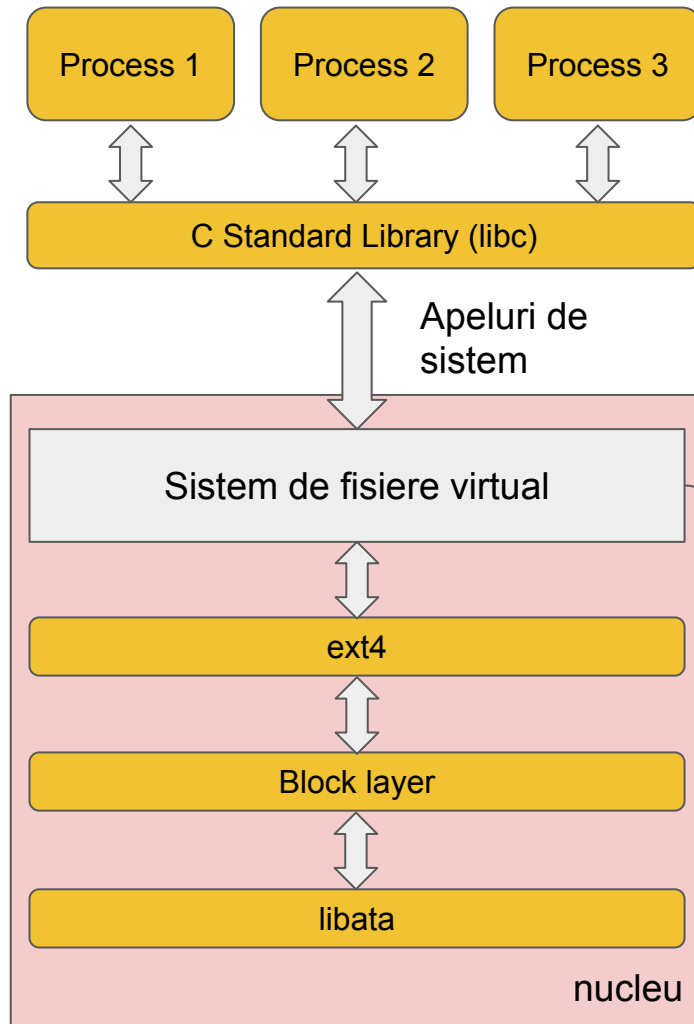
- Nucleul sistemului de operare este unul singur, managementul resurselor este la fel de rapid ca în cazul unui sistem de operare nativ. Toate elementele tuturor containerelor comunică cu nucleul prin intermediul apelurilor de sistem normale, fără nevoie de traducere binara sau emulare.
- Densitatea containerelor este mult mai mare - resursele hardware pot fi partajate mult mai eficient pentru ca sunt sub controlul aceluiași nucleu. Un server poate rula de trei ori mai multe servicii în containere decât în mașini virtuale.
- Containerele sunt mult mai elastice - resursele utilizate de unul dintre containere pot crește pentru moment dacă celelalte containere nu au nevoie de ele și apoi pot fi scăzute la loc dacă celelalte încep să aibă nevoie de ele. În sistemele în care se plătește pentru cantitatea de resurse utilizate, elasticitatea permite obținerea unei cantități mai mari de bani pe același server.
- Companiile care oferă Software as a Service pot alege sistemul de operare utilizat în momentul construcției aplicației, de obicei acesta este același pentru toate serviciile care constituie platforma. De aceea, containerele sunt modalitatea de virtualizare preferată de toate organizațiile care oferă SaaS sau PaaS de foarte mult timp.
- Toate serviciile Google Facebook, Twitter, etc. rulează în containere.

Virtualizare la nivelul sistemului de operare

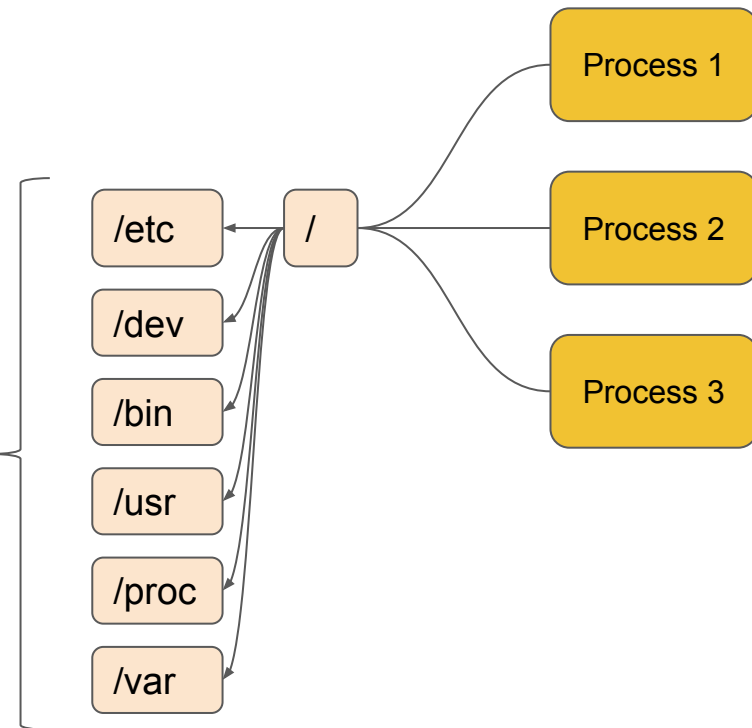
Dezavantaje ale containerelor:

- Organizațiile care oferă Infrastructure as a Service oferă clienților lor posibilitatea de a instala propriul sistem de operare pe serverele virtuale. Containerele nu permit aceasta așa că majoritatea companiilor care oferă servicii de cloud computing utilizează încă metoda virtualizării.
- În infrastructurile software ale întreprinderilor pachetele software nu se aleg în funcție de sistemul de operare pe care rulează ci în funcție de alți factori (caracteristicile acestora, preț, suport tehnic disponibil, compatibilitatea cu sistemele existente, etc.). De aceea, în infrastructura de întreprindere este mult mai probabil să fie nevoie de un mediu eterogen de sisteme de operare (Linux, Windows, fiecare cu anumite versiuni obligatorii). De aceea, infrastructurile întreprinderilor prefera virtualizarea sistemului de containere.
- Trebuie menționat și că în infrastructura de întreprindere serverele se amortizează mult mai repede decât în organizațiile care oferă servicii IT pentru că în întreprinderi profitul se obține din alte surse; computerele sunt doar un ajutor. De aceea, este mai puțin important ca fiecare server să fie utilizat la maximul resurselor sale.

Chroot - cel mai vechi sistem de izolare



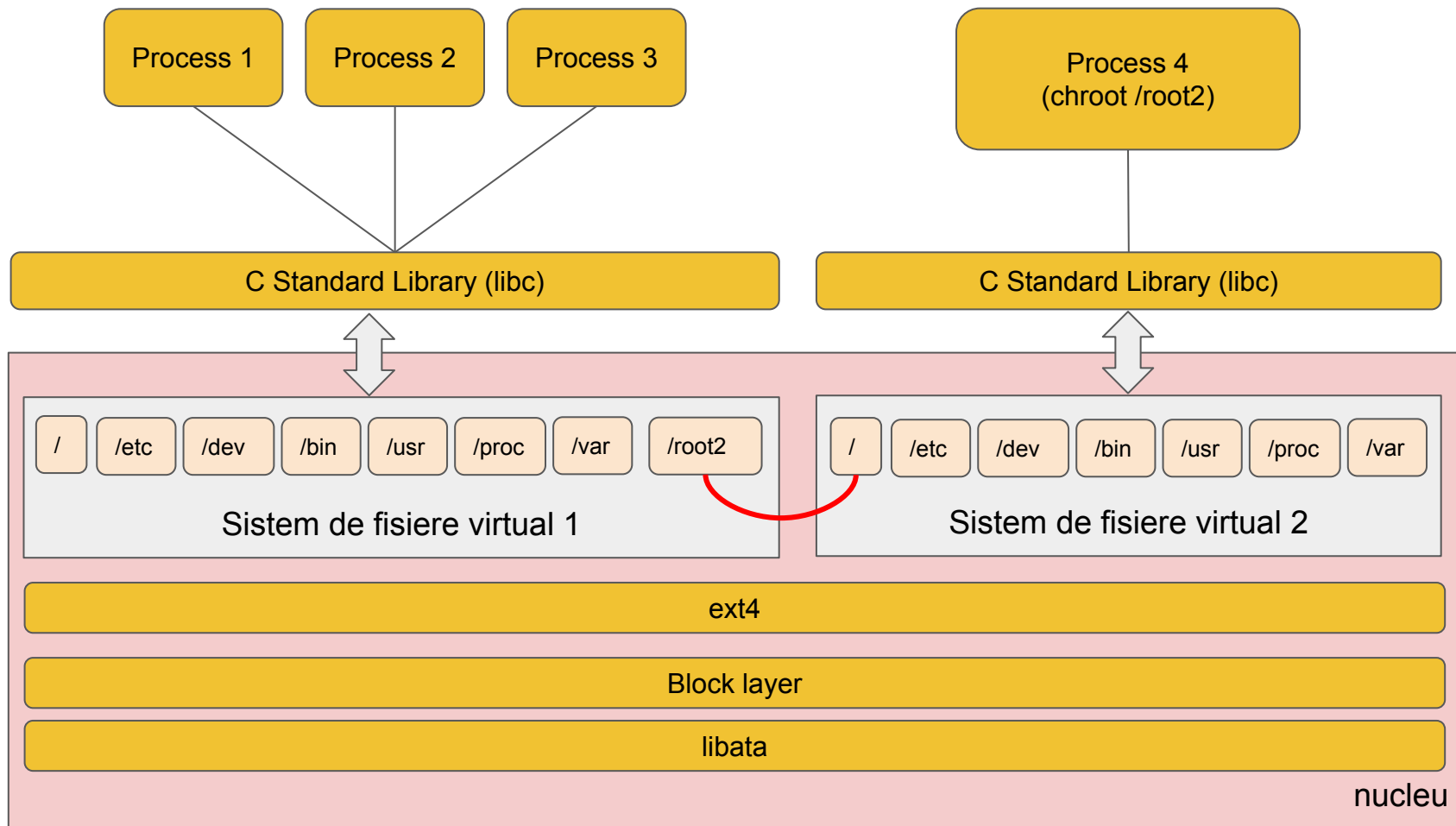
Toate procesele care rulează pe un sistem de operare văd același sistem de fișiere care este o variantă virtuală a sistemului de fișiere de pe partiția principală și de pe celelalte partiții mountate..



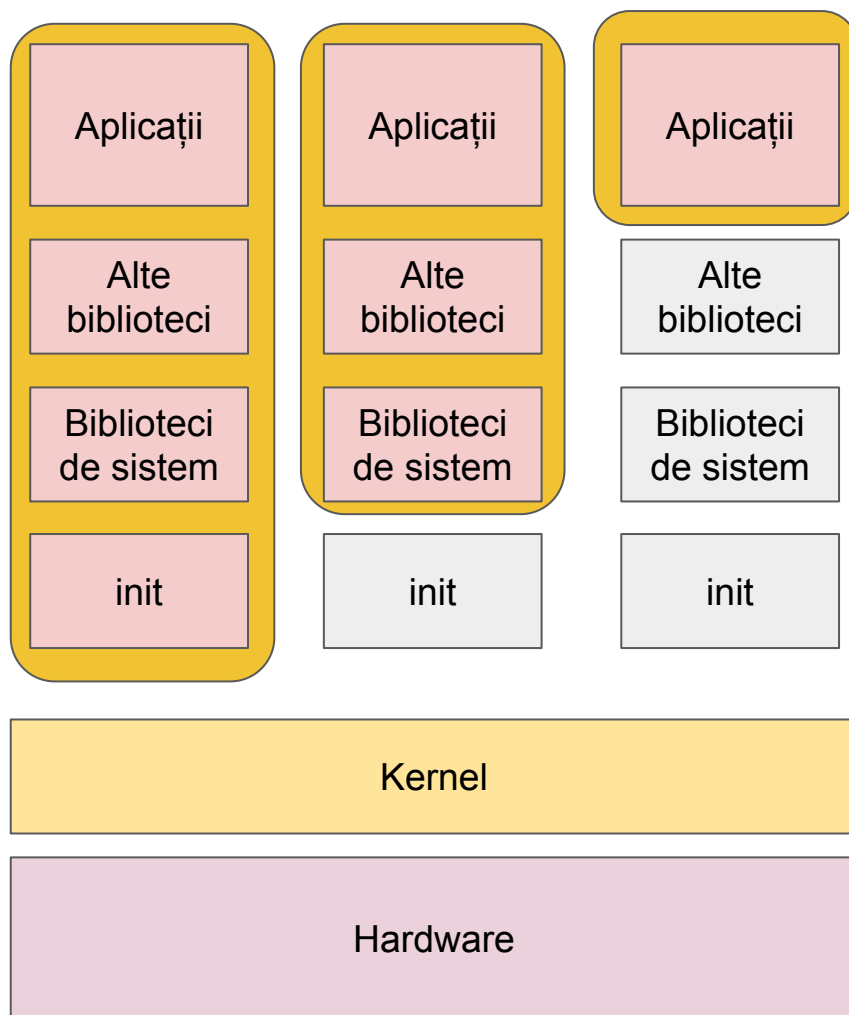
Sistemul de fișiere virtual este construit în memorie de nucleul sistemului de operare, pe baza conținutului partițiilor mountate.

Chroot

Chroot este o comandă, dublată de un apel de sistem, care determină pornirea unui anumit proces pe un alt sistem de fișiere. Comanda chroot este rulată pe sistemul de fișiere virtual 1, astfel, sistemul de fișiere virtual 2 trebuie să fie parte a acestuia, pentru ca chroot-ul să îl vadă. Procesul pornit cu comanda chroot nu va putea vedea decât sistemul de fișiere virtual 2.



Containere - izolare la orice nivel



Containererele pot fi izolate la orice nivel (deasupra nucleului) - pot avea propriul sistem init, biblioteci de sistem, biblioteci auxiliare și aplicații, sau pot folosi init-ul și bibliotecile de sistem ale sistemului de operare principal izolând doar bibliotecile auxiliare și aplicațiile.

Containererele sunt implementate pe Linux folosind două caracteristici relativ noi ale nucleului:

- **Control groups** (câte resurse consumă procesele)
- **Namespaces** (ce anume văd procesele din sistemul de operare)

Control groups

➤ Control groups (cgroups) reprezintă un subsistem al nucleului linux care permite izolarea resurselor ocupate de anumite grupuri de procese.

➤ Echivalentul Windows se numeste “Job Objects”

```
root@student:~# lscgroup | cut -f 1 -d ':' | uniq
```

net_cls,net_prio

Poate atribui diverse prioritati pachetelor care cirtula pe retea. Cu aceste grupuri se poate controla traficul catre si dinspre grupuri de procese

Cpu,cpuacct

Limiteaza si masoara timpul de procesor pe care il consuma procesele

Perf_event

colecteaza date cu privire la performanta proceelor

memory

Izoleaza memoria ocupata de grupuri de procese

hugetlb

Controleaza utilizarea paginarii memoriei cu pagini mari

pids

Limiteaza numarul de fork-uri sau clone-uri ale taskurilor din grup

freezer

Controleaza hibernarea aplicatiilor

devices

Controleaza si restrictioneaza accesul la dispozitive (/dev)

cpuset

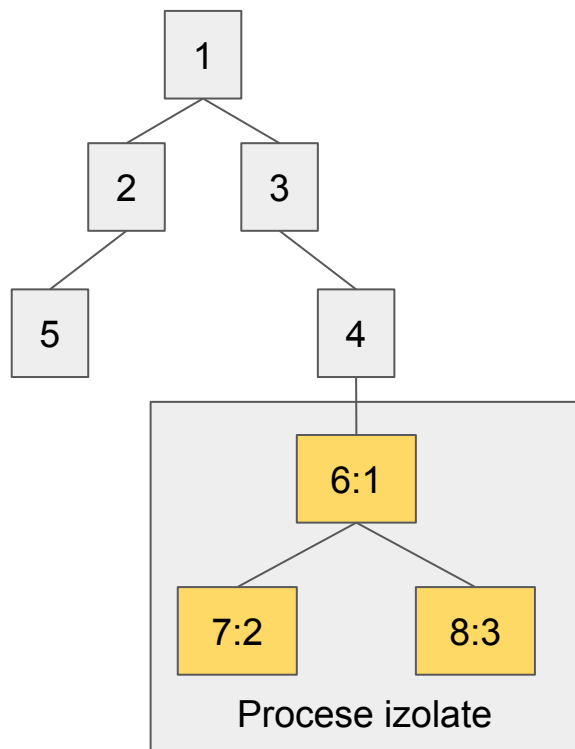
Controleaza accesul la procesor in sistemele multiprocesor

blkio

Controleaza accesul la dispozitivele de tip bloc.

Namespaces

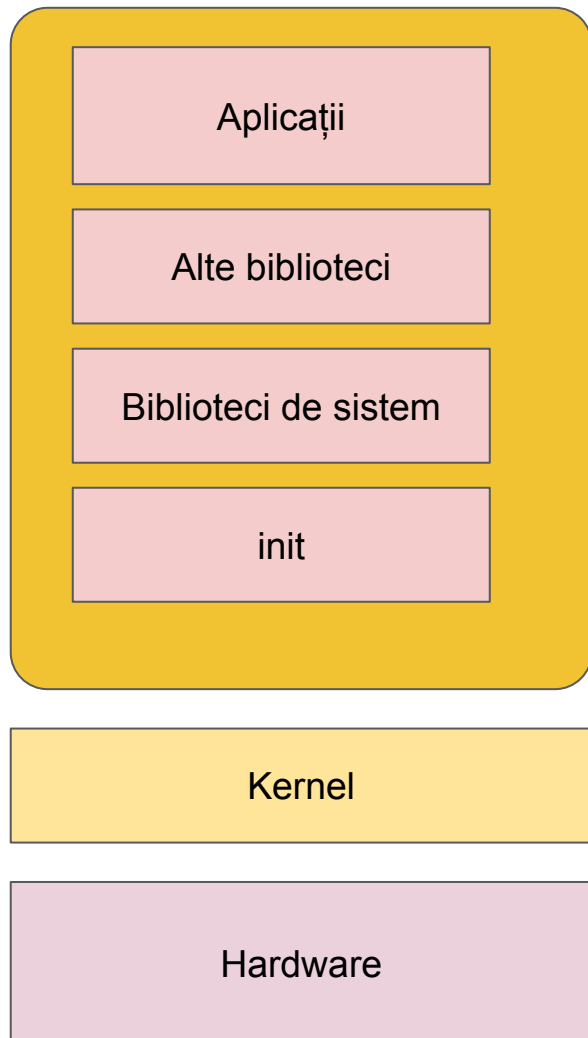
- **Namespaces** sunt un sistem de izolare al resurselor care permit anumitor procese să aibă acces la anumite resurse la care altele nu au acces.
- Prin folosirea namespace-urilor resursele pot fi virtualizate: pot exista mai multe procese cu PID-ul 1 pentru că în realitate PID-ul este precedat de un namespace (asemănător cu chroot).



Procese izolate nu vad procesele părinte; procesele părinte vad procesele izolate

Namespace	Izolare
PID	ID-ul proceselor
NETWORK	Dispozitive de rețea, porturi
USER	ID-uri de utilizator, grupuri
MOUNT	Mount points
IPC	Comunicarea între procese, mesaje
UTS	Nume de host, nume de domeniu

Cgroups + namespaces = containere



Un container se bazeaza pe:

- Propriul arbore de procese, care pare unic pentru aplicațiile de acolo, izolate prin **namespace-uri**.
- Propriul acces la rețea și sistem de fișiere, izolat prin **namespace-uri**.
- Propria listă de utilizatori, izolată prin **namespace-uri**
- Propriul hostname, izolat prin **namespace-uri**
- Resurse limitate pe care le poate consuma (memorie, cpu, I/O, rețea) asigurate prin sistemul de grupuri de control (**cgroups**).

Instrumente pentru containere







- LXC - LXC este un set de instrumente low-level care se constituie în interfața pentru instrumentele de nucleu utilizate în construcția containerelor.
- LXC instalează o serie de comenzi care permit crearea, modificarea și ștergerea containerelor. Conținutul și setările containerelor se găsesc într-o serie de template-uri.
- Template-urile LXC sunt o serie de scripturi BASH care configurează toate elementele containerului: sistemul de fișiere, nodurile pentru dispozitive, permisiuni, etc.
- LXD - LXD este un set de instrumente software care apelează instrucțiunile LXC dar prezintă o sintaxă mai simplă, împreună cu un proces master, cu care se poate comunica printr-o interfață REST. Daemonul LXD permite o serie de operații pe care LXC nu le oferă (decât prin intermediul unor scripturi de automatizare) - migrarea containerelor, backup.
- LXD (și LXC) sunt concepute pentru a porni containere care conțin cât mai mult dintr-un sistem de operare, fiind similare cu hipervizoarele.
- Docker este un sistem de containere asemănător cu LXD (care folosește aceleași instrumente ale LXC) dar care este conceput pentru containere care rulează o singură aplicație, împreună cu librăriile sau dependențele acesteia.

LXD vs Docker

Imagini LXC / LXD (sisteme de operare)

Distribuție	versiune	platformă
alpine	edge	armhf
alpine	edge	i386
archlinux	current	amd64
centos	6	amd64
centos	6	i386
centos	7	amd64
debian	buster	amd64
debian	buster	arm64
debian	buster	armel
debian	buster	armhf
debian	buster	i386
debian	buster	ppc64el

Imagini Docker (aplicații)

 nginx official	7.6K STARS	10M+ PULLS	> DETAILS
 alpine official	2.9K STARS	10M+ PULLS	> DETAILS
 httpd official	1.4K STARS	10M+ PULLS	> DETAILS
 redis official	4.6K STARS	10M+ PULLS	> DETAILS
 busybox official	1.2K STARS	10M+ PULLS	> DETAILS
 ubuntu official	7.0K STARS	10M+ PULLS	> DETAILS

