David Bednárek:
Technické principy virtualizace

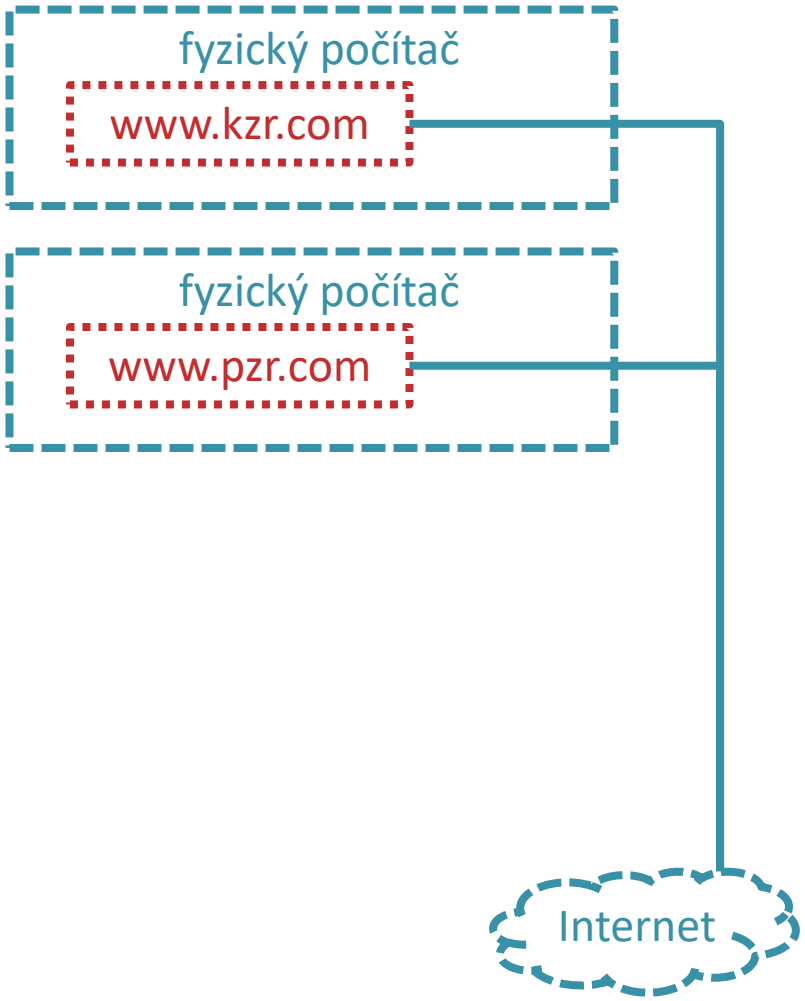
Web Hosting jako motivace pro virtualizaci

Webhosting – postupný vývoj

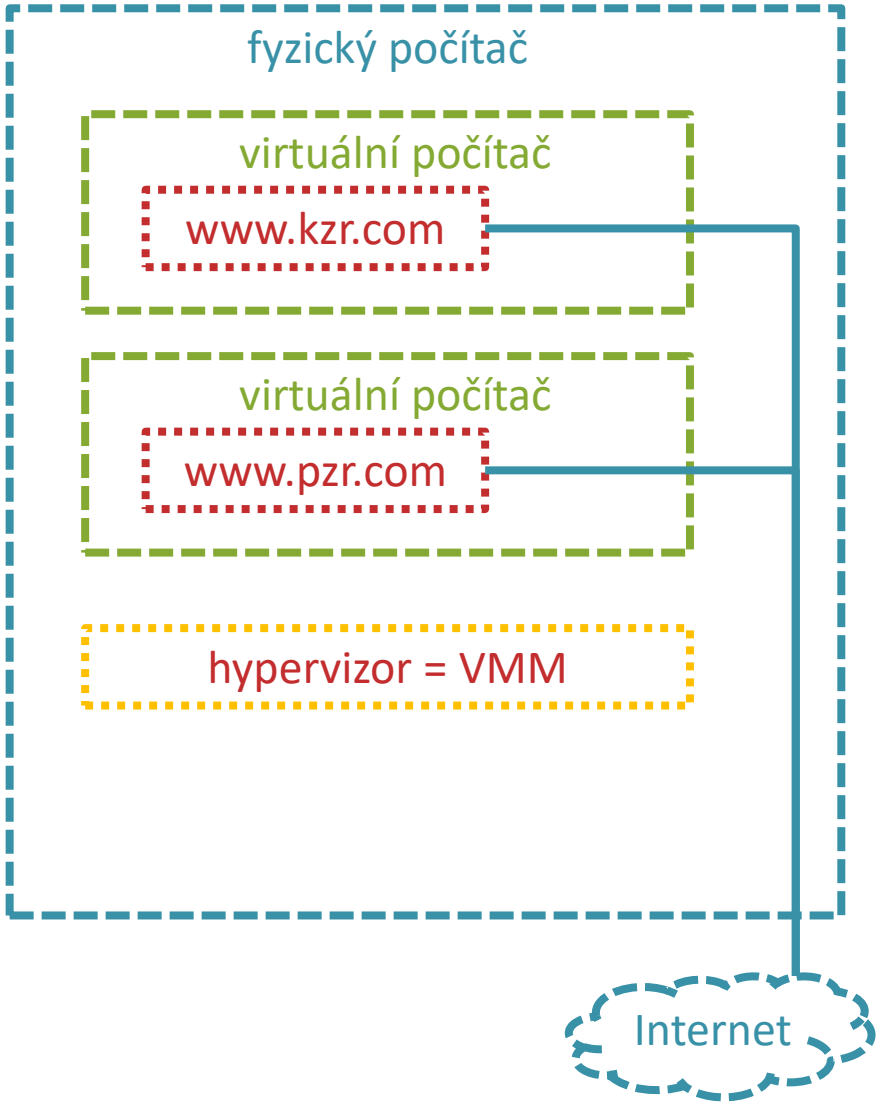
- Statické stránky
  - 1 web = 1 adresář
  - 1 proces (Apache) pro všechny
- Dynamické stránky
  - potenciálně nebezpečný kód
  - 1 web = 1 proces (Apache+PHP)
  - 1 počítač pro všechny
- Uživatelské systémy
  - weby vyžadují odlišné konfigurace
  - 1 web = 1 počítač
  - bez virtualizace zbytečně drahé

Webhosting a virtualizace z rychlíku

Fyzické počítače

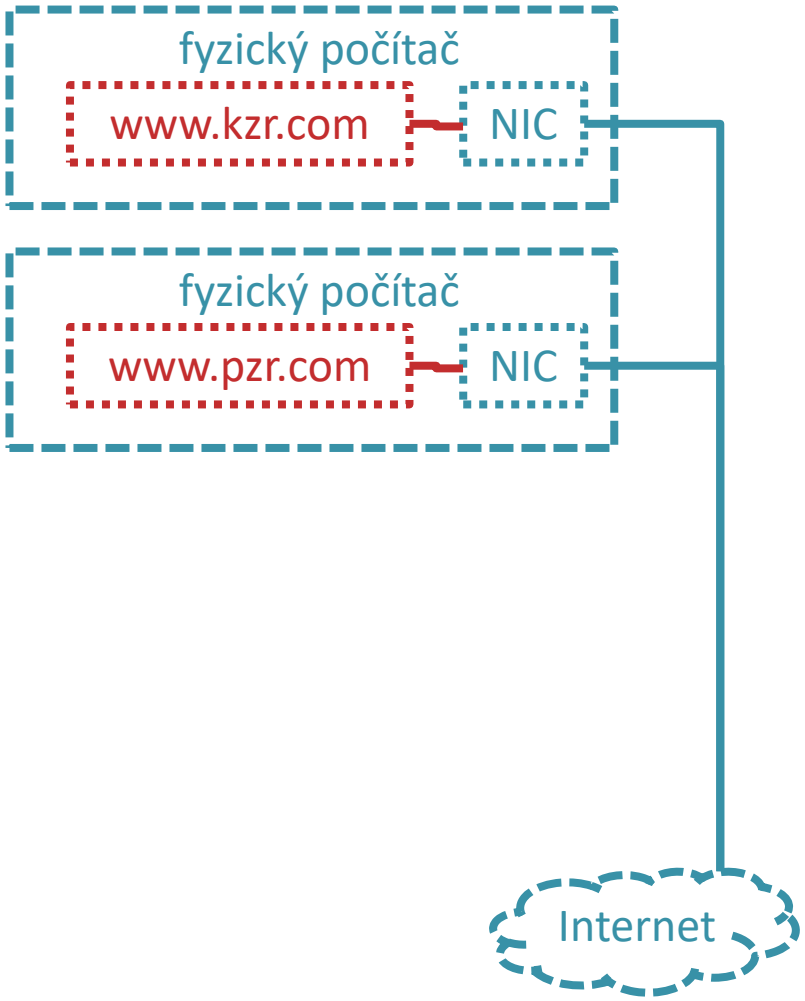


Virtuální počítače

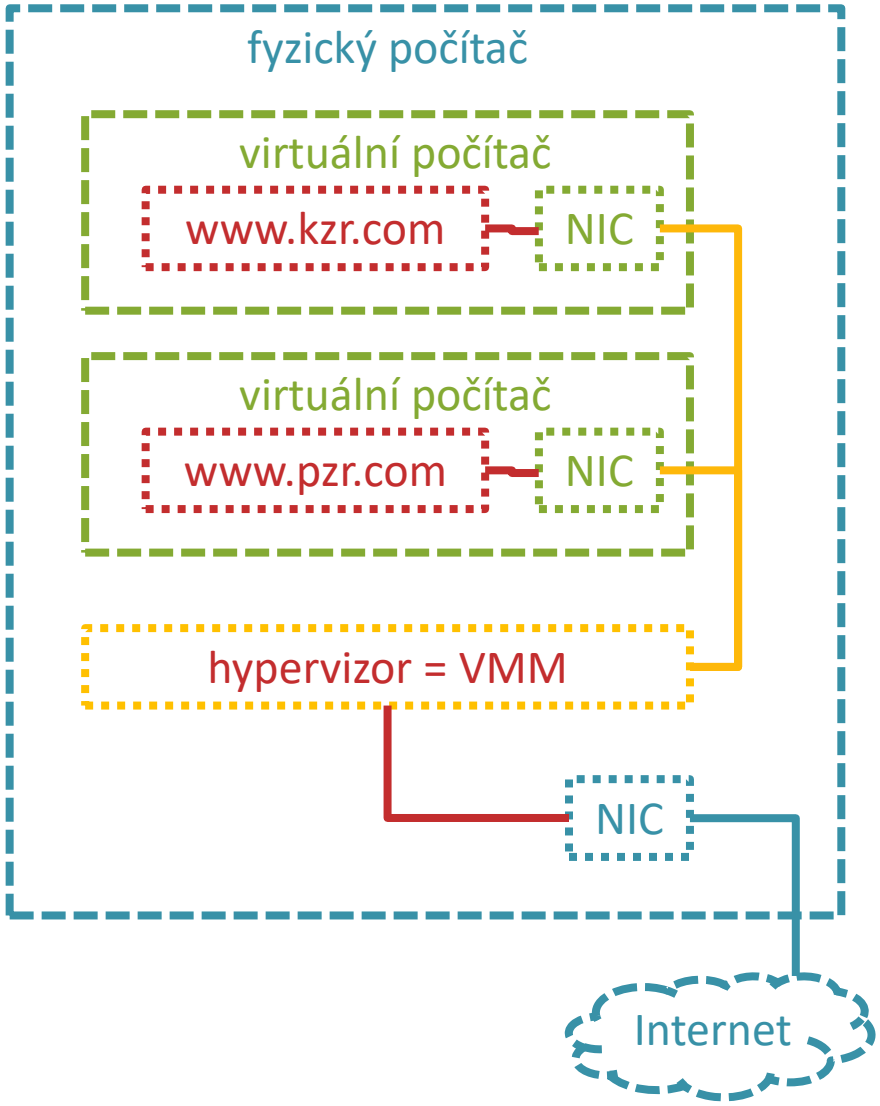


Webhosting a virtualizace podrobněji

Fyzické počítače



Virtuální počítače



Co je to *virtualizace*?

Co je to virtualizace

- virtual
  - Merriam-Webster dictionary
  - very close to being something without actually being it
  - existing or occurring on computers or on the Internet
  - from Latin *virtus* - strength, virtue
    - from *vir* - man

Co je to virtualizace

- Co je to *virtualizace*?
  - Iluze fyzického zařízení, které fyzicky neexistuje
  - Dobře známé rozhraní
    - Fyzického zařízení
      - stránka paměti, počítač, disk, síťová karta, ...
    - Softwarové rozhraní
      - proces-kernel, HAL
  - Implementováno jinak než obvykle
    - softwarově
    - jiným hardware
      - iSCSI řadič
    - v kombinaci software-hardware
      - virtuální paměť, moderní virtualizace procesorů
  - Používání termínu virtualizace je věcí zvyku
    - ssh/RDP není nazýváno "virtualizace konzole"
    - iSCSI disk není "virtuální disk"
    - Java Virtual Machine není virtualizace fyzického stroje

Virtualizace - klady a zápory

Motivace pro virtualizaci

- Lepší využití CPU
  - Většina počítačů se většinu doby fláká
- Lepší využití paměti, diskového prostoru,...
  - OS neumí bezbolestně přidat nový prostor - velikosti bývaji předimenzovány
  - Virtualizace dokáže prezentovat větší než skutečný prostor
- Možnost migrace virtuálních počítačů
  - Load-balancing, fault-tolerance
- Vzdálená správa
  - CD pro instalaci OS lze do virtuální mechaniky vložit kliknutím myši
- Checkpointy
  - Nepovedené změny v konfiguraci lze vrátit
- Výuka uživatelů/správců
- Testování a ladění OS, sítí i aplikací
  - Zkoumání malware

Účel virtualizace (technický pohled)

- Větší počet virtuálních objektů než fyzických
  - virtuální vs. fyzická paměť, virtuální počítač
- Virtuální objekty jiného druhu než fyzické
  - emulace počítače jiné architektury
- Virtuální objekty vzdálené od fyzických
  - vzdálené disky prezentované jako lokální, vzdálená klávesnice+obrazovka
- Virtuální objekty implementované zcela jinak, než fyzické
  - disky implementované souborem
- Virtuální objekty bez vazby na fyzický svět
  - virtuální síťe

- Zásahy do chování, které by bez virtualizace nebyly možné
  - ladění, experimenty, měření
  - šizení všeho druhu (time sharing, thin provisioning)
  - migrace, load balancing

Problémy virtualizace

- Ztráta výkonu
  - Silně závisí na charakteru aplikací i technologii virtualizace
  - Někdy jednotky, někdy desítky procent
- Změna charakteristik při migraci
  - Různá CPU
- Nespolehlivé měření/ladění výkonu

- Nepřipravenost fyzické síťové infrastruktury
  - Migrace virtuálních síťových karet mezi fyzickými

- Nepřipravenost dodavatelů software
  - Nevýhodné licenční podmínky
  - Problémy s individuálními checkpointy v komunikujících systémech

Virtualizace – technické principy

- Virtualizovaná zařízení
  - Počítač
    - Virtualizované rozhraní
      - fyzické rozhraní software-hardware (fyzická virtualizace)
      - softwarové rozhraní uvnitř OS (paravirtualizace)
    - Zahrnuje virtualizaci zařízení uvnitř počítače
  - CPU
    - Virtualizované rozhraní = instrukční sada
      - Aplikační + privilegované instrukce (hardwarová virtualizace)
      - Aplikační instrukce (paravirtualizace)
    - Samo o sobě nema smysl - CPU nemá vstup/výstup
  - Paměť
    - Virtualizované rozhraní = instrukce čtení a zápisu
  - I/O zařízení
    - Virtualizováno
      - na úrovni I/O instrukcí
      - na softwarovém rozhraní uvnitř OS

Úrovně virtualizace

- Sdílení fyzického počítače virtuálními
  - CPU
    - guest OS i hypervizor používají preemptivní multitasking
  - Paměť
    - guest OS už má svou virtuální paměť
    - hypervizor přidává druhou úroveň
  - Disky
    - virtuální disk mapován do společného diskového prostoru
    - iSCSI, SAN, NAS,...
  - Sítě
    - trunk mode, NAT, virtuální sítě,...
  - Další zařízení
    - exkluzivní přístup
    - sdílený přístup
    - vzdálené USB apod.

Virtualizace – abstraktní pohled

Virtualizace – abstraktní pohled

- Virtualizace se týká běhu software
- Cílem virtualizace je
  - sdílet hardware pro běh několika (nesouvisejících) kusů software
  - přesouvat běžící nebo pozastavené kusy software jinam
  - pozorovat chování software apod.

- Software je
  - FORTRAN, C, ... – přeložený program obsahující instrukce fyzického CPU
  - Java, C#, ... – napůl přeložený program obsahující instrukce virtuálního stroje
  - Python, PHP, ... – zdrojové kódy
- Běh software vždy zahrnuje běh knihovních funkcí
  - Část knihoven je vždy v podobě instrukcí fyzického CPU (typicky přeloženo z C)
    - V mnoha případech se v nativních knihovnách odehrává většina běhu
    - Vyšší jazyk a nativní knihovny typicky komunikují sdílením paměti
    - Oddělení běhu nativních knihoven od vyšších vrstev je prakticky nerealizovatelné
  - I v případě vyšších jazyků je vhodné chápat běh software jako provádění instrukcí fyzického CPU (knihovna, kód generovaný JIT překladem, interpreter)

Virtualizace – abstraktní pohled

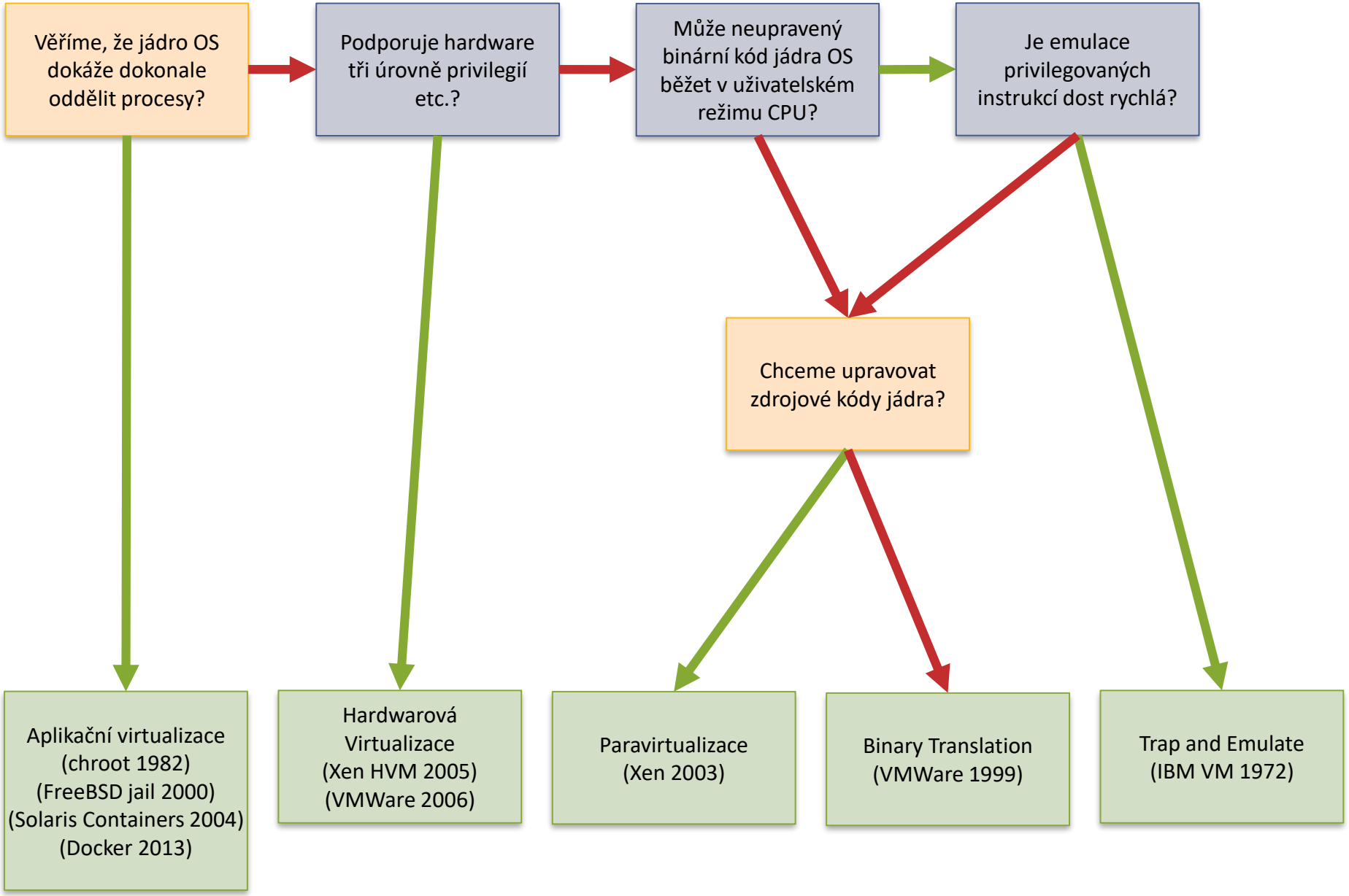
- Software je program prováděný jako posloupnost nativních instrukcí
- Software dnes obvykle neběží jako jeden samostatný program
  - Z hlediska balancování zátěže a migrace je potřeba řešit skupinu programů společně
  - Jejich spolupráce je zajištěna službami OS – tyto služby je nutné také zahrnout do balancovaného/migrovaného softwarového balíku

- Základní otázkou virtualizace je místo, kde je balík software odříznut od okolí
  - Kontejnery – na rozhraní aplikace-OS
  - Paravirtualizace – uprostřed OS
  - Skutečná virtualizace – na rozhraní OS-HW

Varianty virtualizace

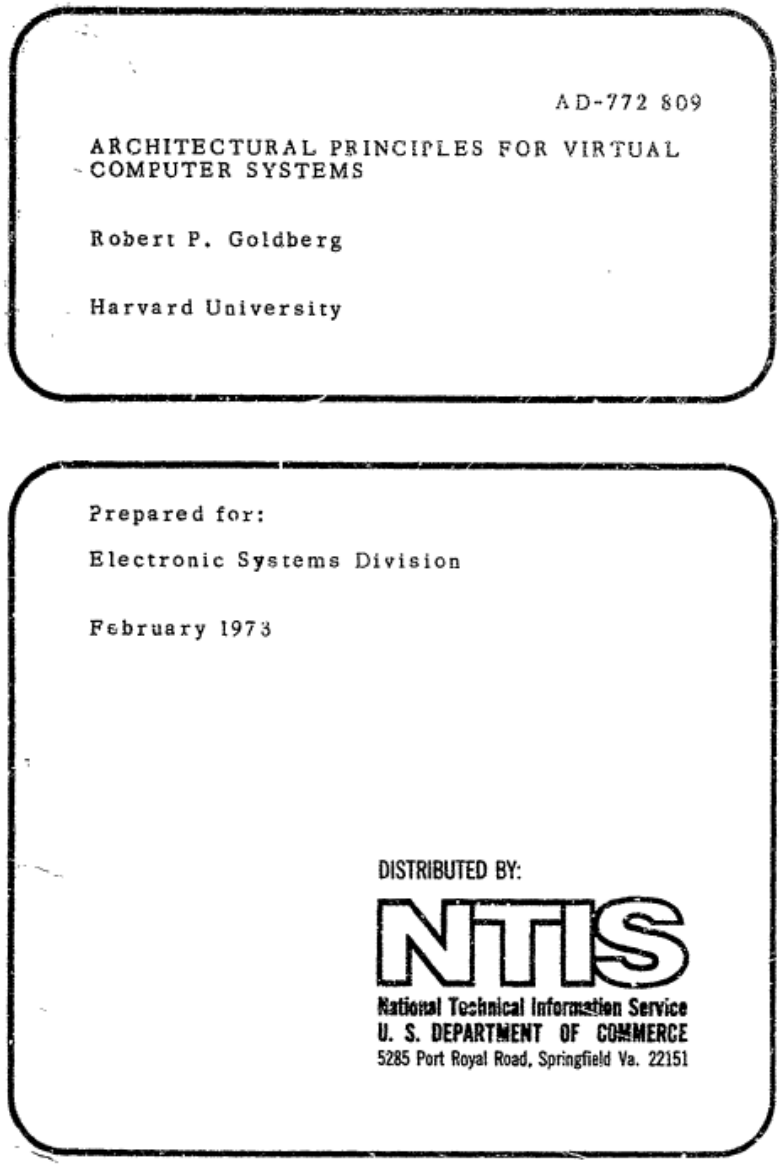
- Virtualizaci lze dělat na mnoha úrovních
  - Aplikační virtualizace
    - chroot, WoW, UAC, kontejnery, bash.exe
    - skupinám procesů je prezentováno jiné prostředí
    - implementováno operačním systémem
  - Paravirtualizace
    - Xen, Microsoft Hyper-V
    - na fyzickém stroji běží několik upravených operačních systémů
    - hypervizor řeší alokaci zdrojů a serializaci přístupu k zařízením
  - (Hardwarová/Klasická) Virtualizace
    - VMWare, Microsoft Hyper-V, Xen HVM
    - na fyzickém stroji běží několik neupravených operačních systémů
    - hypervizor vytváří každému z nich iluzi fyzického hardware

Volba přístupu k virtualizaci



Virtualizace - historie

Virtualizace - historie



AD-772 809

ARCHITECTURAL PRINCIPLES FOR VIRTUAL COMPUTER SYSTEMS

Robert P. Goldberg

Harvard University

Prepared for:

Electronic Systems Division

February 1973

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

- První éra virtualizace
  - 1972 – IBM VM pro S/370
    - Koexistence různých OS
    - Time-sharing a virtuální paměť nad OS, které tyto pojmy neznají
    - Ladění OS
      - Včetně VM ve VM
  - Každý desátý S/370 používal VM
  - 1980... – Postupný zánik
    - Mainframes vytlačeny levnějšími architekturami (minipočítače, PC)
      - Nový hardware neumožňoval virtualizaci
    - Nástup Unixu
      - VM zbytečně komplikují komunikaci mezi procesy

Virtualizace - požadavky

- Požadavky na virtualizaci
  - Gerald J. Popek and Robert P. Goldberg, 1974

- Equivalence / Fidelity
  - Program běžící pod VMM se musí chovat v zásadě stejně, jako by běžel na ekvivalentním fyzickém stroji přímo

- Resource control / Safety
  - VMM musí mít úplnou kontrolu nad virtualizovanými zdroji

- Efficiency / Performance
  - Statisticky převládající část strojových instrukcí musí být prováděna bez zásahu VMM

Virtualizace - historie

- Druhá éra virtualizace
  - 1999 – VMWare Workstation
    - Softwarová virtualizace (BT)
    - VMM jako aplikace Windows NT
  - 2002 – VMWare ESX Server
    - VMM nahrazuje OS hostitele
  - 2003 – Xen
    - Paravirtualizace
      - Modifikace OS hosta

- Architektura x86 je pro virtualizaci nevhodná
  - Dědictví procesoru Intel 80286
    - 1982 – v době první éry virtualizace
  - Pokusy o nápravu
    - 2005 – Intel VT-x
    - 2006 – AMD-V
  - V některých případech stále pozorovatelná ztráta výkonu
  - Výkon se postupně zlepšuje
    - zlepšováním HW podpory
    - paravirtualizací kritických míst OS

Implementace VMM

Virtualizace CPU

Virtualizace CPU

- Rozhraní software-hardware v klasickém CPU (s dvěma módy)
  - Neprivilegovaný mód (aplikační procesy)
    - Instrukční sada (neprivilegované instrukce)
    - Stav CPU
      - Skutečný obsah neprivilegovaných registrů
      - Efekt stavu privilegovaných registrů
    - Paměťový prostor procesu
      - R/W operace
      - Manipulace prostřednictvím služeb OS
    - Mechanismus volání jádra OS
      - Speciální instrukce nebo (úmyslná) chyba
  - Privilegovaný mód (jádro OS) - navíc
    - Privilegované instrukce
    - Stav CPU
      - Skutečný obsah privilegovaných registrů
    - Paměťový prostor procesu
      - Manipulace s HW částí stránkování (TLB)
    - I/O operace
    - Obsluha přerušení synchronních i asynchronních

Virtualizace CPU

- Cíl: implementovat "jinak" rozhraní software-hardware
  - Nepřiznat sdílení fyzického CPU s jinými virtuálními stroji
- Situace v procesorech bez HW podpory - s dvěma módy

- Neprivilegovaný mód (aplikační procesy)
  - Instrukční sada (neprivilegované instrukce)
    - Implementováno bez úprav fyzickým CPU - jinak to kvůli výkonu nelze
    - Fyzické CPU je občas odebráno preemptivním multitaskingem hypervizoru
  - Stav CPU
    - Skutečný obsah neprivilegovaných registrů
      - Preemptivní multitasking hypervizoru vyměňuje obsah registrů
    - Efekt stavu privilegovaných registrů
      - Fyzický stav je nutně jiný, ale odchylka nesmí být viditelná
  - Paměťový prostor procesu
    - Mechanismus virtuální paměti musí kombinovat sdílení paměti mezi VM a sdílení mezi procesy jednoho VM
  - Mechanismus volání jádra OS
    - Nelze přímo připustit původní semantiku (přepnutí do privilegovaného módu a skok do OS)

Virtualizace CPU

- Cíl: implementovat "jinak" rozhraní software-hardware
  - Nepřiznat sdílení fyzického CPU s jinými virtuálními stroji
- Situace v procesorech bez HW podpory - s dvěma módy

- Privilegovaný mód (jádro OS) - navíc
  - Privilegované instrukce
    - Většinou nelze připustit přímé provedení fyzickým CPU
  - Stav CPU
    - Fyzický obsah většiny privilegovaných registrů nelze přiznat
  - Paměťový prostor procesu
    - Manipulace s HW částí stránkování (TLB) musí být emulována
    - Nutnost zaznamenat i manipulaci prováděnou neprivilegovanými instrukcemi
  - I/O operace
    - U sdílených zařízení nelze připustit přímé provedení fyzickým CPU
  - Obsluha přerušení synchronních i asynchronních
    - Přerušení obvykle přepíná CPU do privilegovaného režimu
    - Začátek obsluhy přerušení musí být ve VMM

Přístupy k virtualizaci

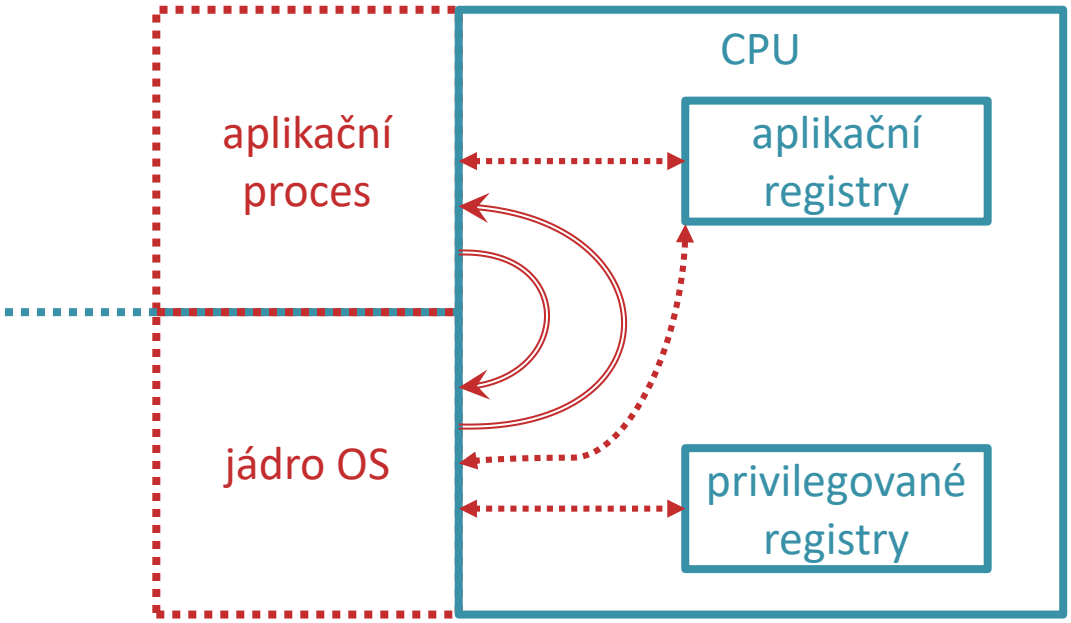
- Aplikační kód je vždy vykonáván fyzickým CPU
- Kód jádra OS lze vykonávat různými způsoby

- Situace v procesorech bez HW podpory
  - Trap and Emulate - Jádro OS vykonáváno fyzickým CPU běžícím v aplikačním režimu [první éra virtualizace]
    - Privilegované instrukce způsobí výjimku a jsou emulovány
      - Časová penalizace každého přechodu aplikace-jádro
    - Neprivilegované instrukce musejí běžet identicky s privilegovaným režimem
      - Nepřiznat změněný stav procesoru neprivilegovanou instrukcí
  - Binary translation - přeložení do "bezpečného" kódu [VMWare x86]
    - Instrukce manipulující s privilegovanou částí stavu jsou upraveny
      - Vyžaduje čas a prostor pro překlad (on demand)
    - Bezpečný kód vykonáván fyzickým CPU [v privilegovaném režimu ?]
      - Šetří čas na přepínání režimu CPU
- Hardwarová podpora virtualizace [VMWare x64]
  - Více úrovní privilegovanosti ve stavu procesoru
    - Na stroji s N úrovněmi lze emulovat virtuální stroj s N-1 úrovněmi
  - Oddělené (stínové) registry pro fyzický a virtuální stav CPU

Přístupy k virtualizaci

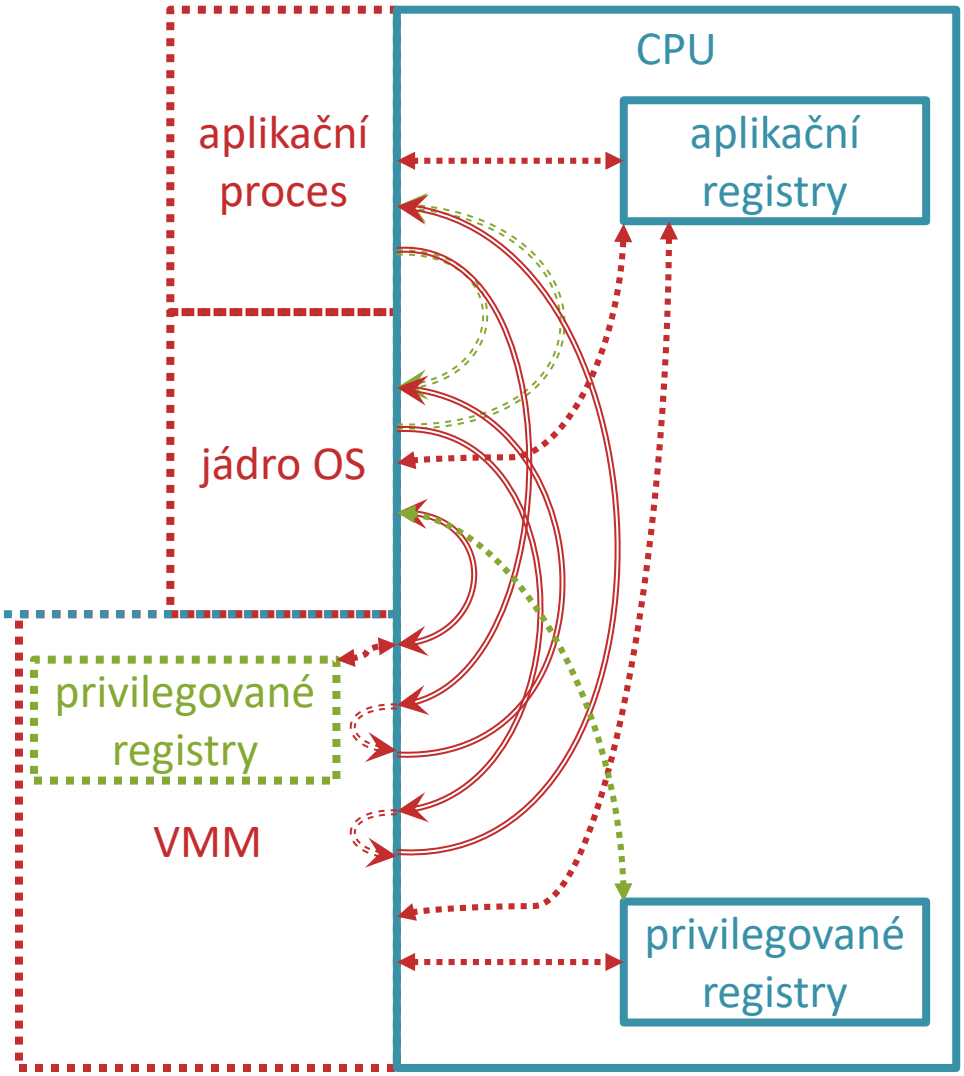
- Aplikační kód je vždy vykonáván fyzickým CPU
- Kód jádra OS lze vykonávat různými způsoby
  - Fyzickým CPU běžícím v aplikačním režimu [první éra virtualizace]
  - Binary translation - přeložení do "bezpečného" kódu [VMWare x86]
    - Bezpečný kód vykonáván fyzickým CPU v privilegovaném [?] režimu
  - Hardwarová podpora virtualizace [VMWare x64]
    - Více úrovní privilegovanosti ve stavu procesoru
      - Na stroji s N úrovněmi lze emulovat virtuální stroj s N-1 úrovněmi
  - Paravirtualizace [Xen, některé varianty Hyper-V]
    - Upravený OS nepoužívá privilegované instrukce
      - Ani jinak nezasahuje přímo do privilegovaného stavu (stránkování)
    - OS je vykonáván fyzickým CPU v aplikačním režimu
      - Ochrana proti chybám v OS
    - Privilegované akce nahrazeny voláním hypervizoru
      - Výrazně menší režie než při emulaci rozhraní SW-HW
      - Zůstává režie přepínání aplikace-OS přes hypervizor

Operační systém na fyzickém CPU



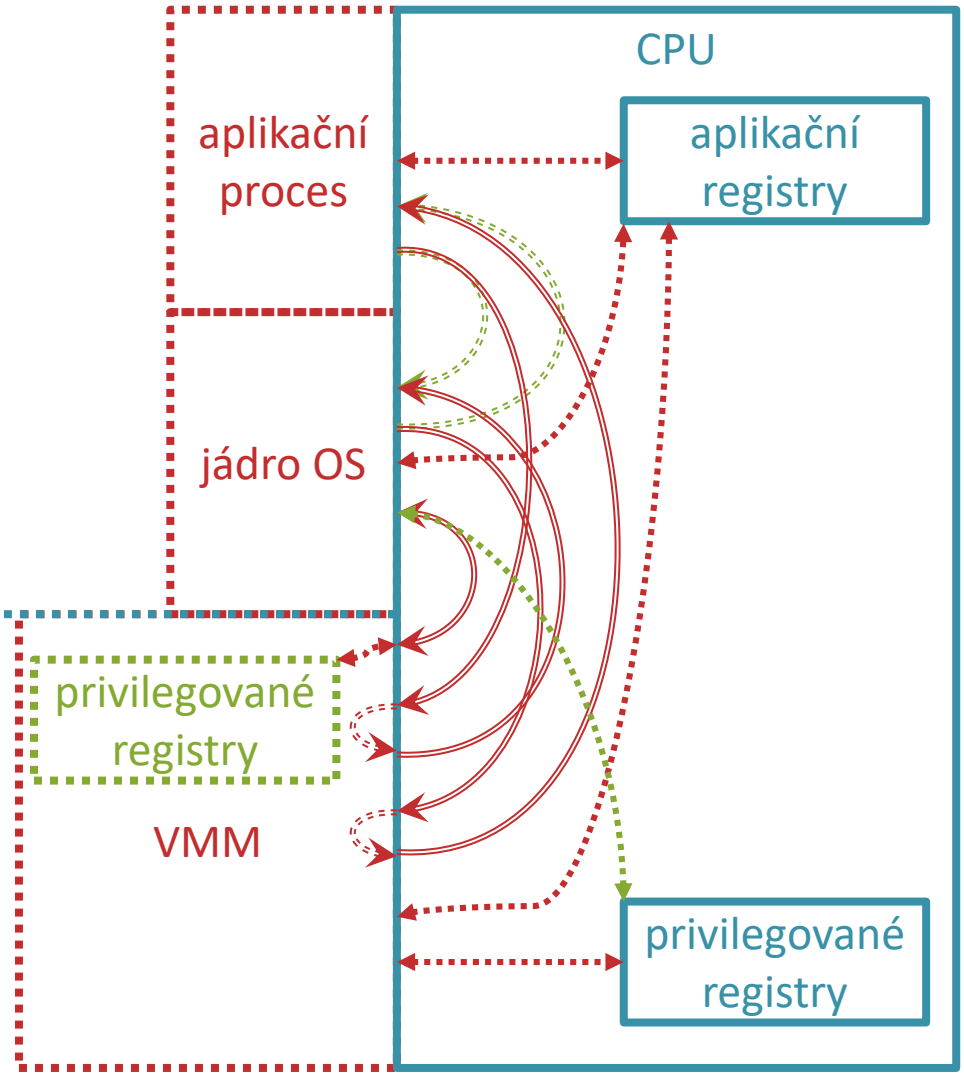
- Režim CPU
  - Aplikační
  - Privilegovaný
  - Odlišeno příznakem v privilegovaném registru
- Vstup do privilegovaného režimu
  - Přerušení
  - Instrukce pro volání jádra (SYSCALL)
  - Chyba
- Návrat z privilegovaného režimu
  - Instrukce návratu (IRET, SYSRET)

Operační systém na virtualizovaném CPU - Trap and Emulate



- Aplikační proces
  - Pracuje normálně
- Jádro OS
  - Pracuje v aplikačním režimu
  - Privilegovaná instrukce způsobí softwarové přerušení
  - VMM toto přerušení obsluhuje - emuluje instrukci, která ji způsobila
- Privilegované registry virtuálního CPU
  - uloženy v paměti VMM

Trap and Emulate - Nevýhody

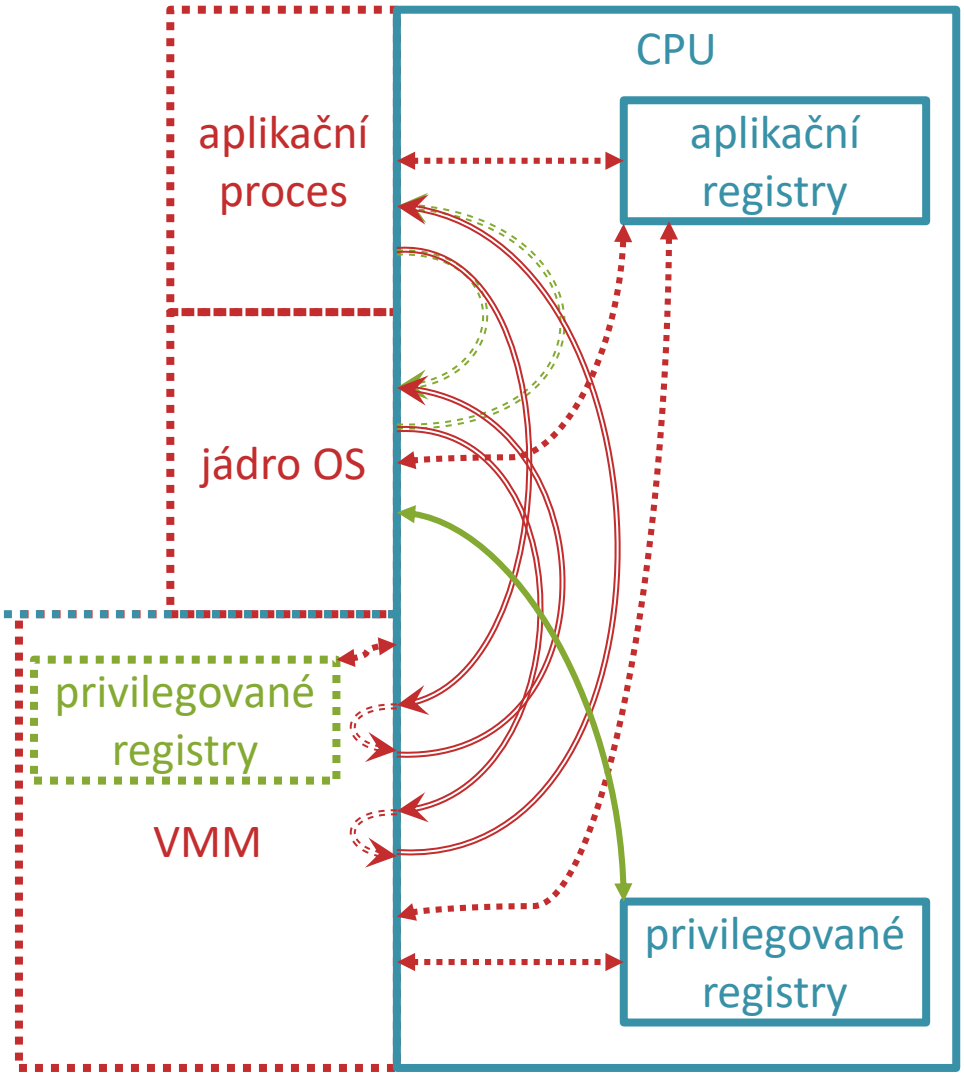


- Jádro OS vyvolává hodně privilegovaných instrukcí
  - Počet závisí na architektuře CPU, systému a OS
- Softwarová emulace instrukcí je pomalá
  - Režie přerušení
  - Režie dekódování
  - Režie závisí na architektuře CPU

Trap and Emulate - Nevýhody

- První éra virtualizace
  - IBM 370
    - I/O řešeno HW kanály = málo privilegovaných instrukcí v OS
    - Mizivý paralelismus = levné skoky
    - Jednoduchá a pravidelná instrukční sada = levné dekódování
    - Monolitické aplikace = málo meziprocesové komunikace
  - Trap and Emulate byla vhodná technika
- Dnes
  - Jádro OS vyvolává hodně privilegovaných instrukcí
    - Intenzivní komunikace mezi procesy a s I/O zařízeními
  - Softwarová emulace instrukcí je pomalá
    - Režie přerušení
    - Režie dekódování
    - Režie závisí na architektuře CPU

Trap and Emulate - Nevhodné procesory

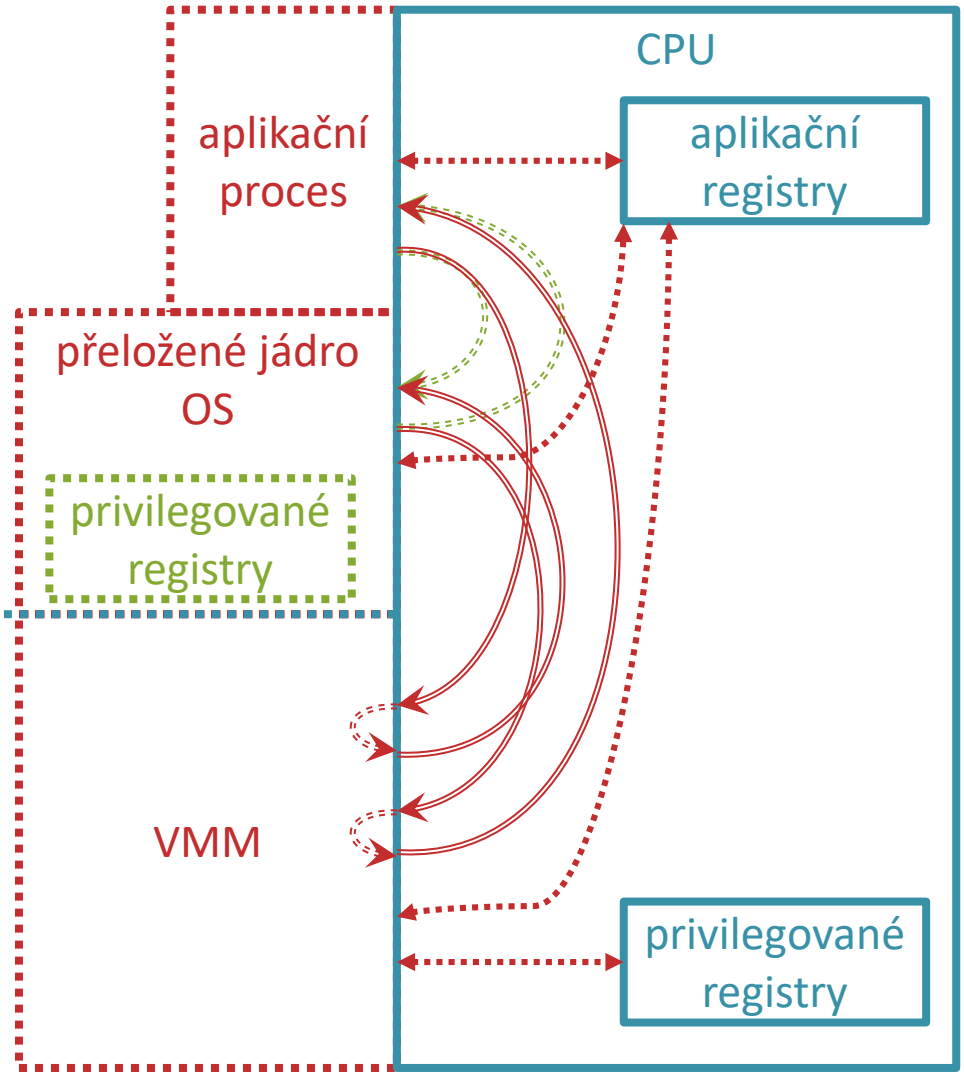


- Nevhodná architektura procesoru neumožňuje použití Trap and Emulate
  - Intel x86
- Typické chyby
  - Část privilegovaných registrů je čitelná neprivilegovanou instrukcí
  - Některé instrukce se v různých režimech chovají různě
  - Příliš mnoho instrukcí jádra OS vyvolává v aplikačním režimu chybu

Problémy režimu trap-and-emulate

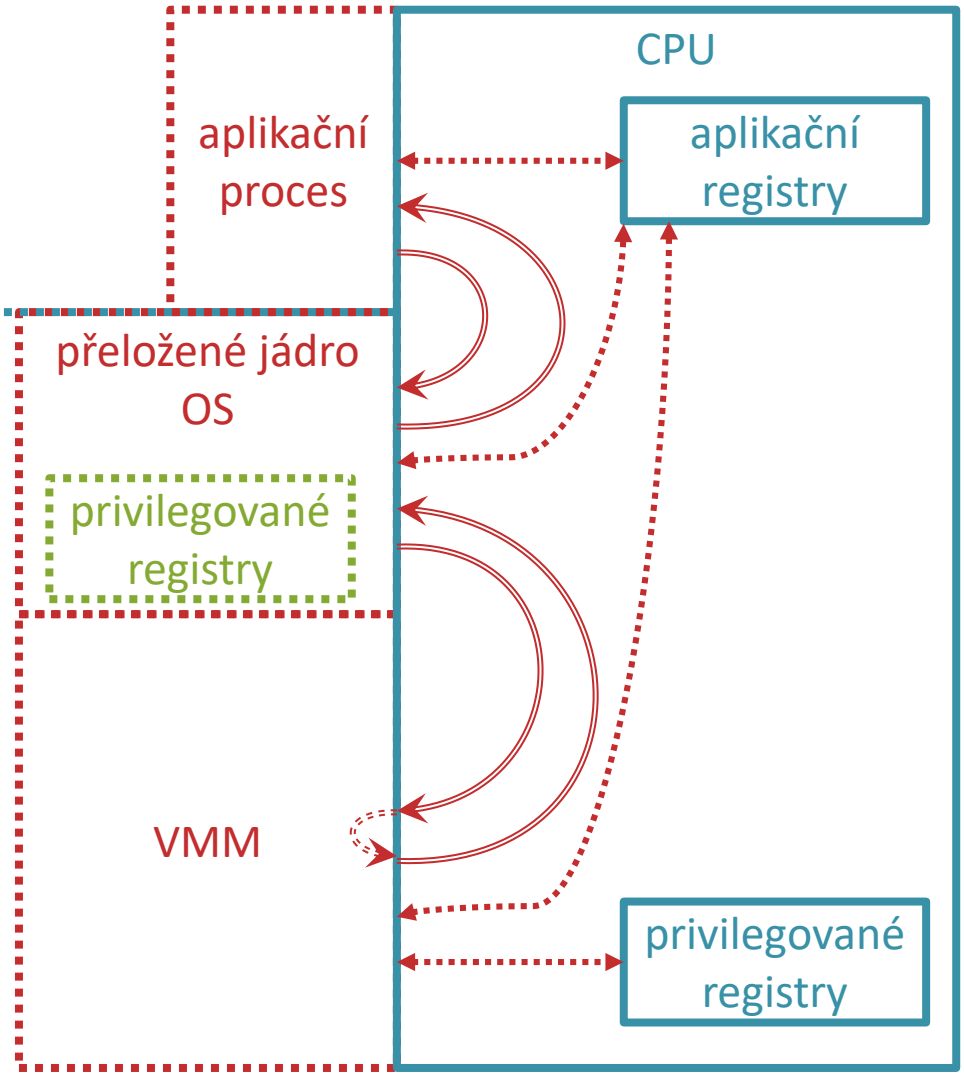
- *Komprese privilegií*
  - Jádro virtualizovaného OS pracuje na jiné prioritní úrovni, než si myslí
    - Některé instrukce se chovají jinak (intel x86)
    - Řešeno velmi pracně překladem (VMWare)
- Společný adresový prostor
  - CPU nepřepíná adresový prostor při volání jádra
    - Ochrana OS řešena privilegovanými stránkami
  - Virtualizovaný OS nakládá s virtuálním adresovým prostorem jako s vlastním
  - Nezbývá místo pro VMM
    - VMWare: řešeno segmentací (dostupná pouze v 32-bit režimu)
- Příliš mnoho přechodů VM-VMM

Softwarová emulace s překladem (neprivilegovaný mód)



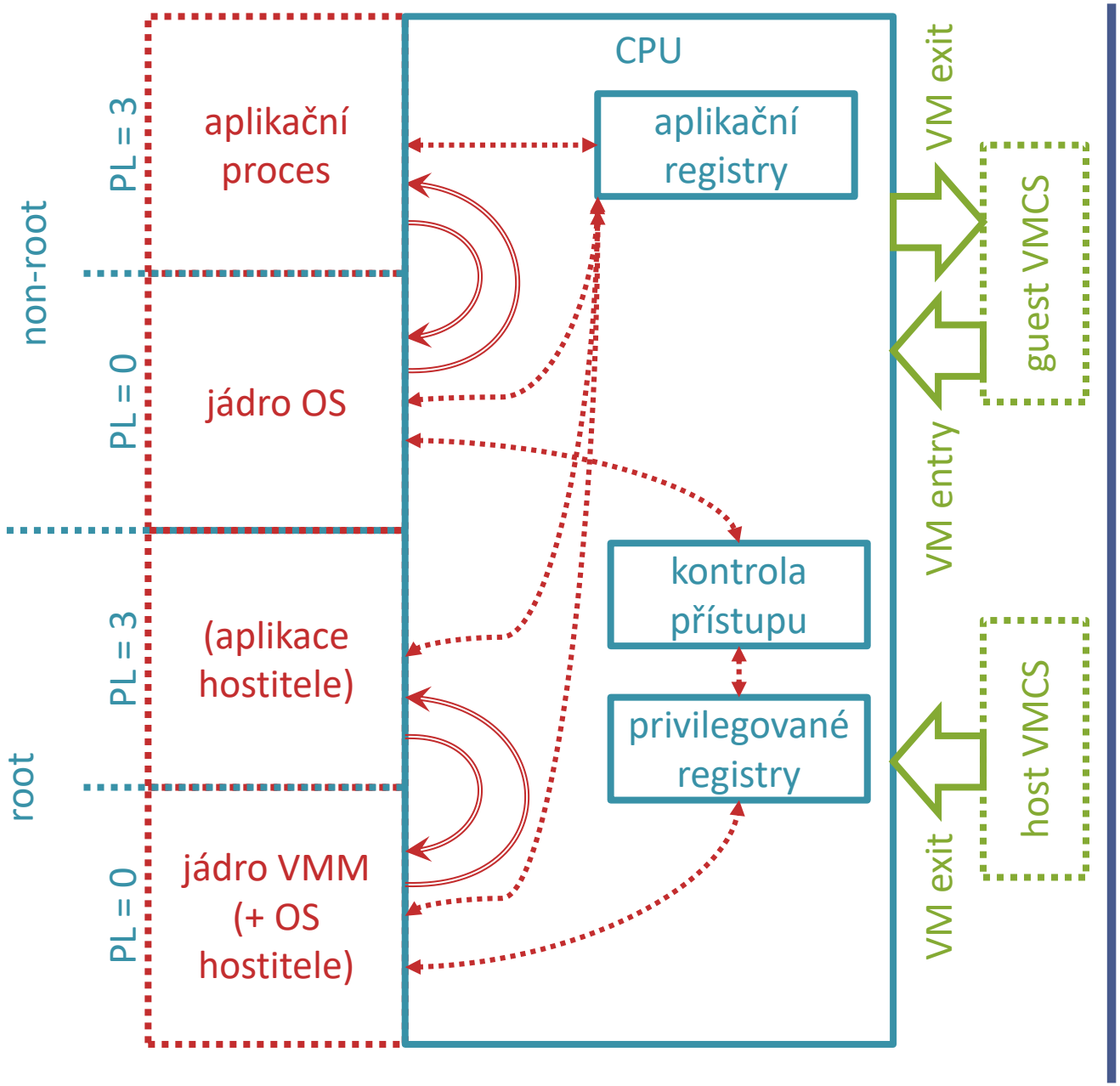
- Úprava jádra OS
  - Binární kód jádra je překladačovými technikami upraven tak, aby neprováděl privilegované operace
  - Privilegované registry CPU si upravený kód emuluje sám
  - Zásah VMM nutný pro:
    - Přechody aplikace-jádro
    - Akce s významným efektem (např. na stránkování)
    - I/O operace
    - Systém přerušení

Softwarová emulace s překladem (privilegovaný mód)



- Úprava jádra OS
  - Binární kód jádra je překladačovými technikami upraven tak, aby neprováděl privilegované operace
  - Privilegované registry CPU si upravený kód emuluje sám
  - Jádro je překladem fakticky donuceno dobrovolně spolupracovat s VMM
    - Přechody aplikace-jádro jsou stejně drahé jako bez VM
    - Přechody jádro-VMM jsou levné
  - Nutná důvěra v mechanismus překladu

Hardwarová podpora virtualizace – nový rozměr privilegovanosti



- Intel VT-x / AMD-V
  - provedení se liší
- „Root“ režim CPU
  - Odpovídá CPU bez virtualizace
  - Lze využít pro běh hostitelského OS
- „Non-root“ režim
  - Přístup k privilegovanému stavu omezen
  - Nežádoucí akce způsobují „VM exit“
- Přepínání režimu
  - Kritická část stavu CPU se načítá/ukládá do paměti
  - Zahrnuje přepnutí stránkování

Výhody HW podpory

- Odstraněna komprese privilegií
  - Podmínka: Virtualizované OS samy HW podporu virtualizace nevyužívají
    - VMCS Shadowing (Intel 2013): Rekurzivní virtualizace je možná
    - Na IBM VM/370 bylo demonstrováno 5 úrovní vnoření virtualizace
- Přepínání adresového prostoru při VM entry/exit
  - Ochrana paměti VMM, plná transparence pro virtualizovaný OS
    - Komplikuje přístup VMM do paměti VM (při emulaci I/O apod)
- Menší počet přechodů VM-VMM
  - Lze vyladit konfigurací HW kontroly přístupu k privilegovanému stavu
  - Demonstrováno cca. dvojnásobné zrychlení některých úloh
    - Unix fork and wait benchmark
    - Kompilace rozsáhlých projektů s malými moduly

Hardwarová podpora virtualizace – Intel x86

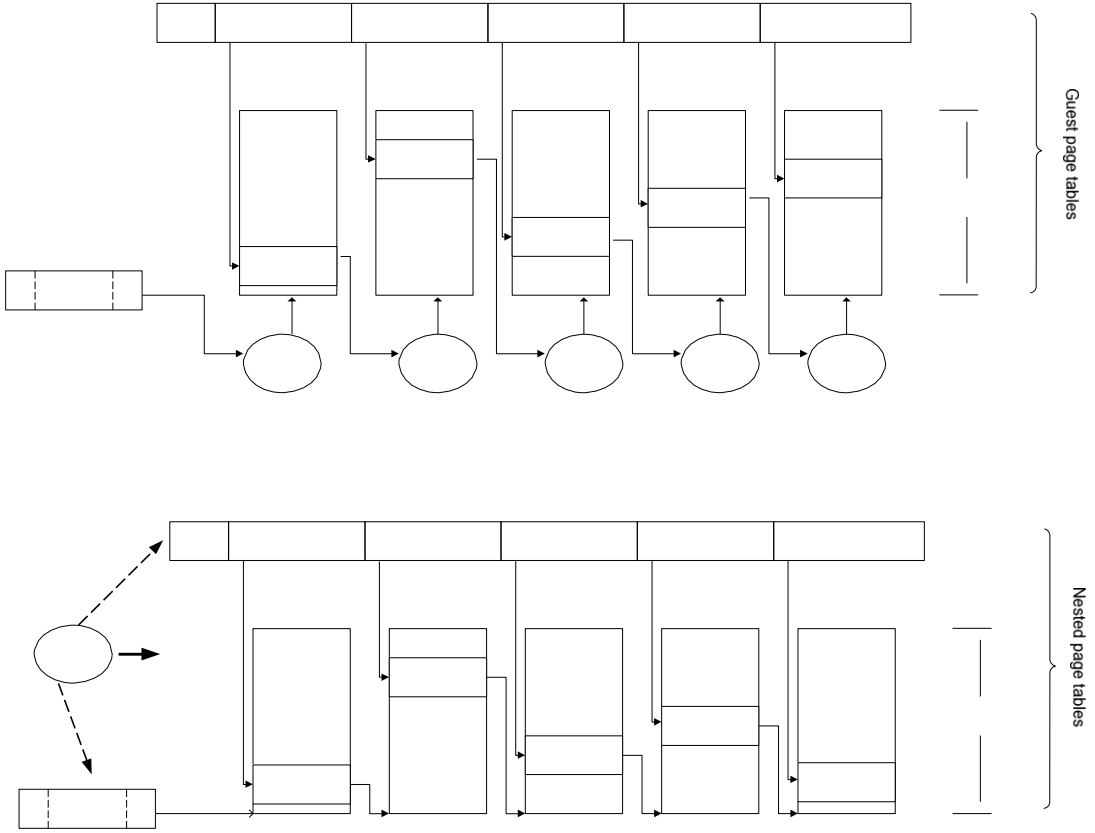
HW podpora virtualizace – x86

- Intel VT-x a VT-d
  - Řada rozšíření CPU i podpůrného chipsetu k podpoře virtualizace
    - Neustále přibývají další
  - Jednotlivé úpravy jsou často použitelné nezávisle
  - Významné virtualizační softwary je využívají téměř všechny
    - Intel spolupracuje s producenty software
- AMD-V
  - Úpravy ve stejném čase (2006) podobným směrem
    - Většina není kompatibilní s Intelem
- Situace znepřehledněna obchodní politikou
  - Různé verze CPU mají různý stupeň podpory
  - Obchodní názvy maskují podstatu věci
    - Některá rozšíření jsou triviality, jiná jsou velmi netriviální

Intel VT-x – klíčové součásti

- Root/non-root execution (2005)
  - Řešení problému komprese privilegií
- Extended Page Table (EPT) (2008)
  - Řešení problému virtualizace virtuální paměti
- VMCS Shadowing (2013)
  - Podpora rekurzivní virtualizace

Virtuální paměť ve virtuálním počítači – EPT (Intel) / NPT (AMD)



- Procesor drží dvě sady stránkovacích tabulek a skládá je
  - Všechny adresy v GPT (počínaje CR3) jsou překládány na fyzické pomocí NPT
    - Pro pětiúrovňové stránkování je zapotřebí 25 přístupů do paměti

Intel VT-x – další součásti

- FlexPriority
  - Virtualizace klíčové části řadiče přerušení (APIC)
- Pause-loop exiting
  - Detekce spin-locků způsobující exit do VMM
    - Pro provoz více virtuálních procesorů na méně fyzických
- VGuest Preemption Timer
  - Časovač s lepší granularitou a rychlejší obsluhou
    - Pro virtualizaci aplikací s mírnými real-time nároky
- FlexMigration
  - Virtualizace identifikace CPU a jeho schopností
- Virtual Processor ID (VPID)
  - Klíč záznamu v TLB obsahuje identifikátor VM
    - Není třeba invalidovat celou TLB při přepínání VM-VMM a VM-VM
- Real-mode support
  - Podpora virtualizace při startu virtualizovaného OS

Intel VT-d – podpora virtualizace v chipsetu

- IOMMU
  - I/O zařízení přistupují k paměti přes MMU podobně jako CPU
  - Address Translation Services (ATS) support
    - Rozšíření standardu sběrnice PCI Express
  - Large Intel VT-d Pages
    - Umožňuje sdílení CPU a DMA verzí stránkovacích tabulek
- Interrupt-remapping support
  - Částečná virtualizace řadiče přerušení
- Virtual Machine Device Queue
  - Network Interface Card s více stavovými prostory pro přímý přístup z VM
- Single-Root I/O Virtualization (SR-IOV)
  - I/O zařízení deklarují své schopnosti virtualizace
  - Rozšíření standardu PCI Express

Intel – další technologie týkající se virtualizace

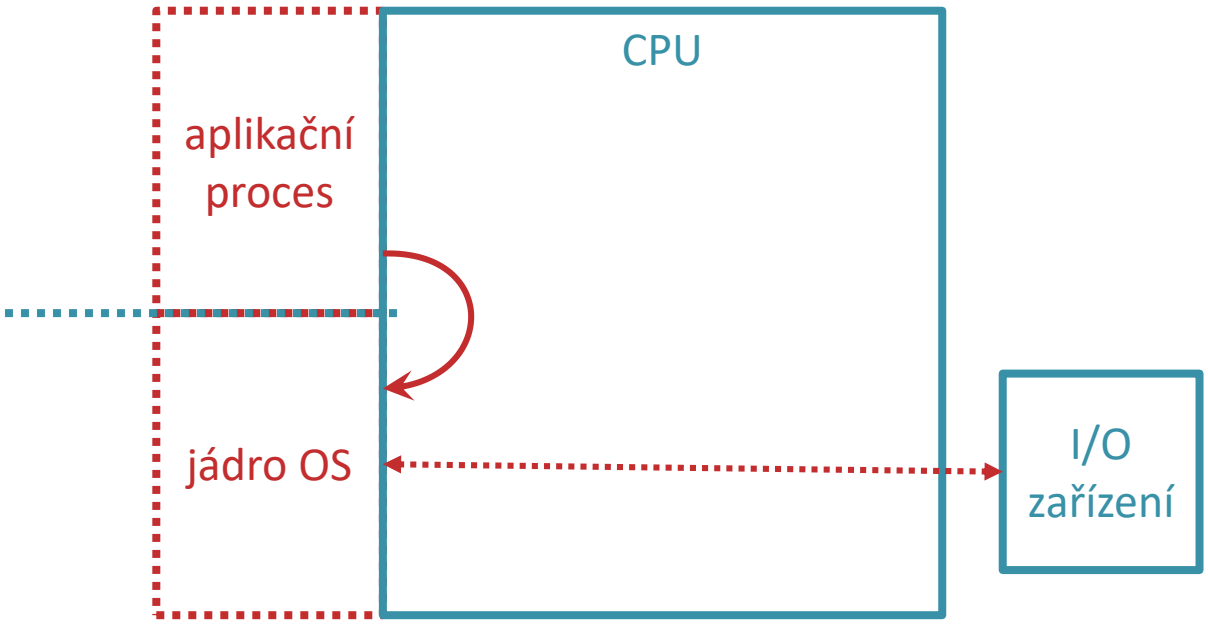
- Graphics Virtualization Technology
  - Využití výpočetní síly (Intel) GPU ve virtuálních strojích
    - Exkluzivní přístup (GVT-D)
    - Sdílený přístup (GVT-S) – vyžaduje přizůsobení ovladačů ve virtuálních strojích
    - Časový multiplex (GVT-G)
- Data Direct I/O Technology (DDIO)
  - Zpřístupnění CPU cache pro DMA – snížení latence síťové komunikace

AMD

- Virtualization Extensions to the x86 Instruction Set
  - Enables software to more efficiently create VMs so that multiple operating systems and their applications can run simultaneously on the same computer
- Tagged TLB
  - Hardware features that facilitate efficient switching between VMs for better application responsiveness
- Rapid Virtualization Indexing (RVI)
  - Helps accelerate the performance of many virtualized applications by enabling hardware-based VM memory management
- AMD-V Extended Migration
  - Helps virtualization software with live migrations of VMs between all available AMD Opteron processor generations
- I/O Virtualization
  - Enables direct device access by a VM, bypassing the hypervisor for improved application performance and improved isolation of VMs for increased integrity and security

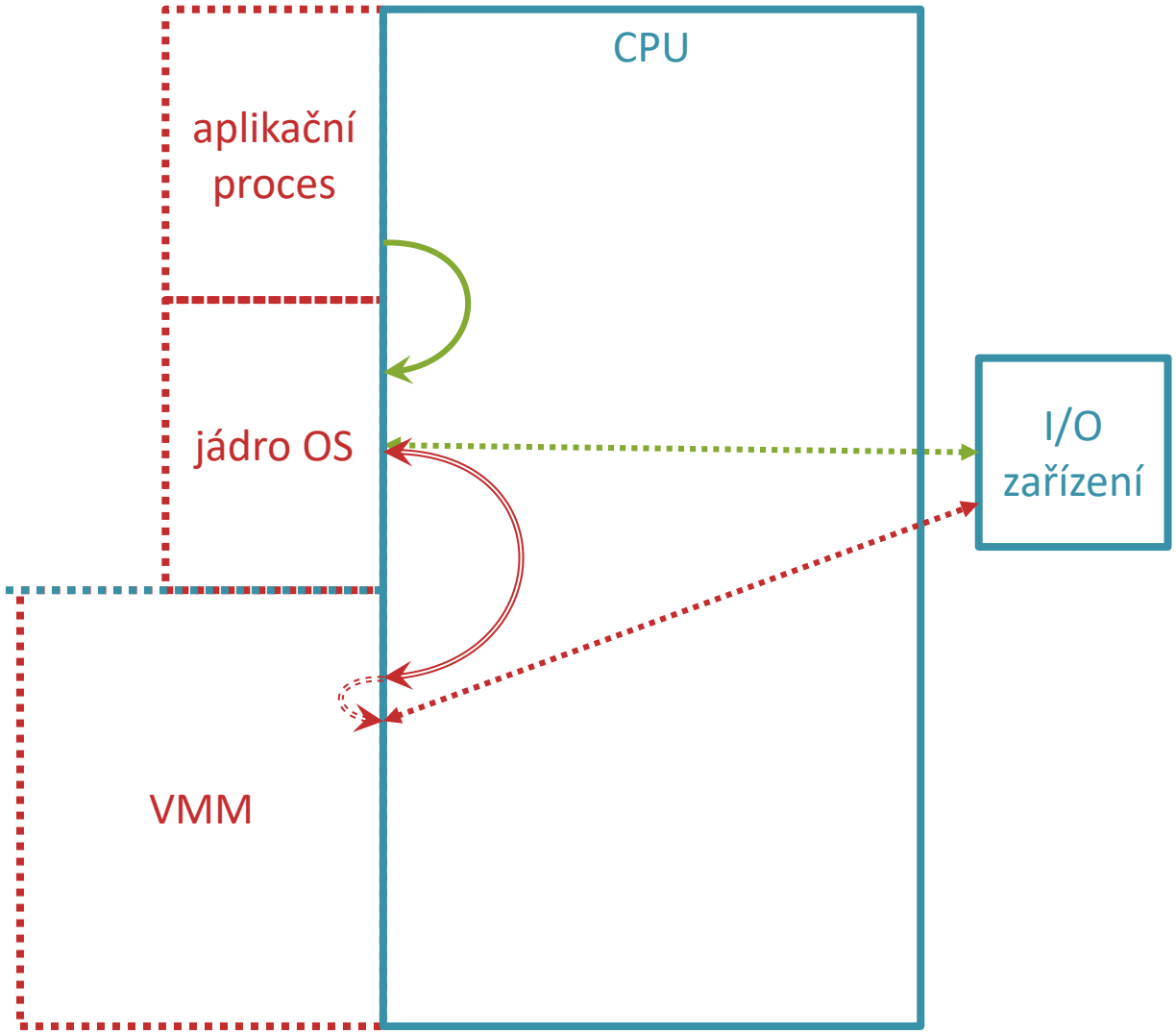
Virtualizace I/O

Přístup k I/O na fyzickém počítači



- Aplikační procesy realizují veškeré I/O voláním OS
- OS komunikuje s I/O zařízením
  - Privilegované I/O instrukce, nebo
  - Paměťově mapované zařízení chráněné stránkovacím mechanismem

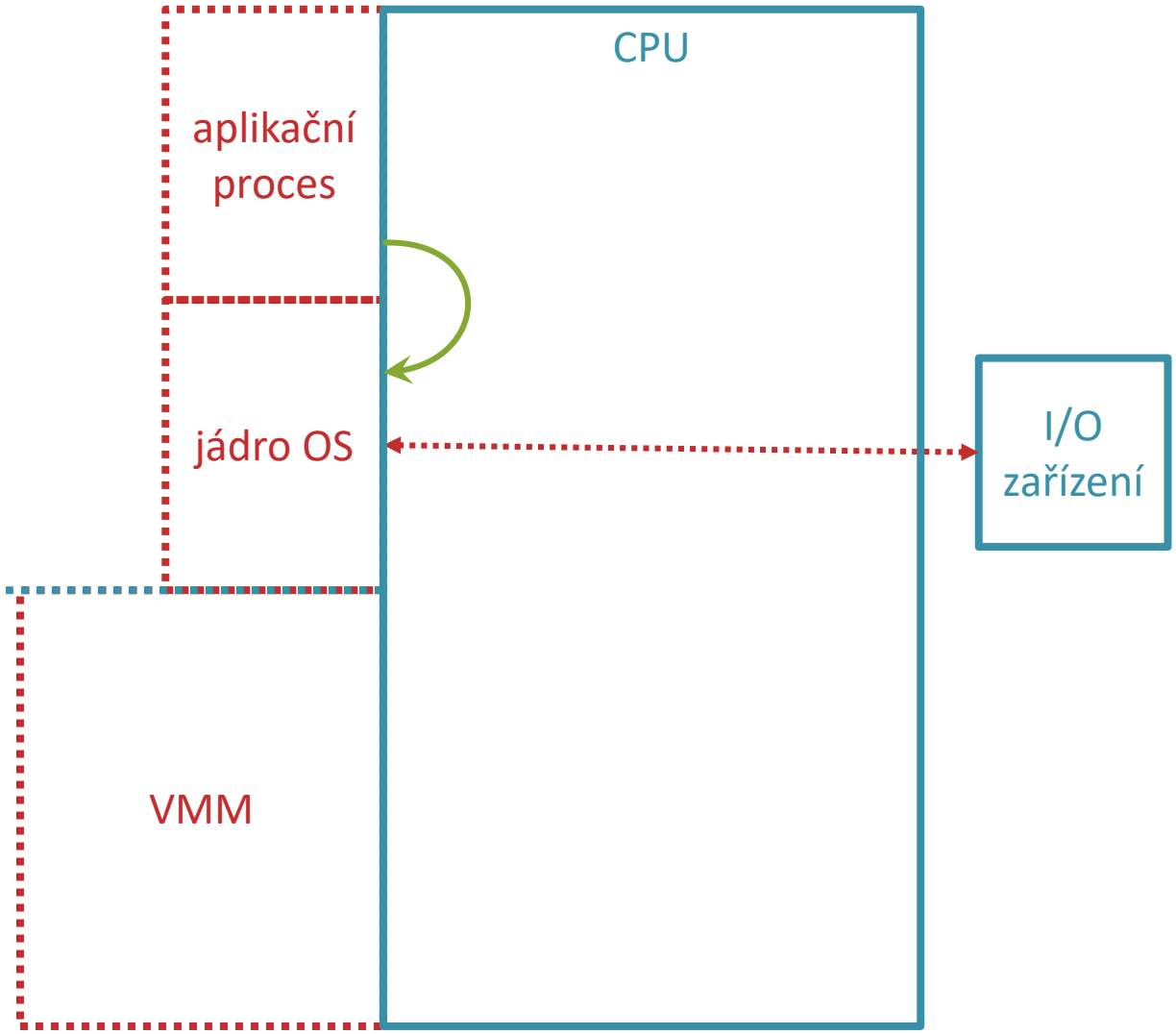
Přístup k fyzickému I/O zařízení na virtuálním CPU



- Privilegované I/O instrukce jsou provedeny emulátorem ve VMM
- Paměťově mapované zařízení může být zpřístupněno přímo

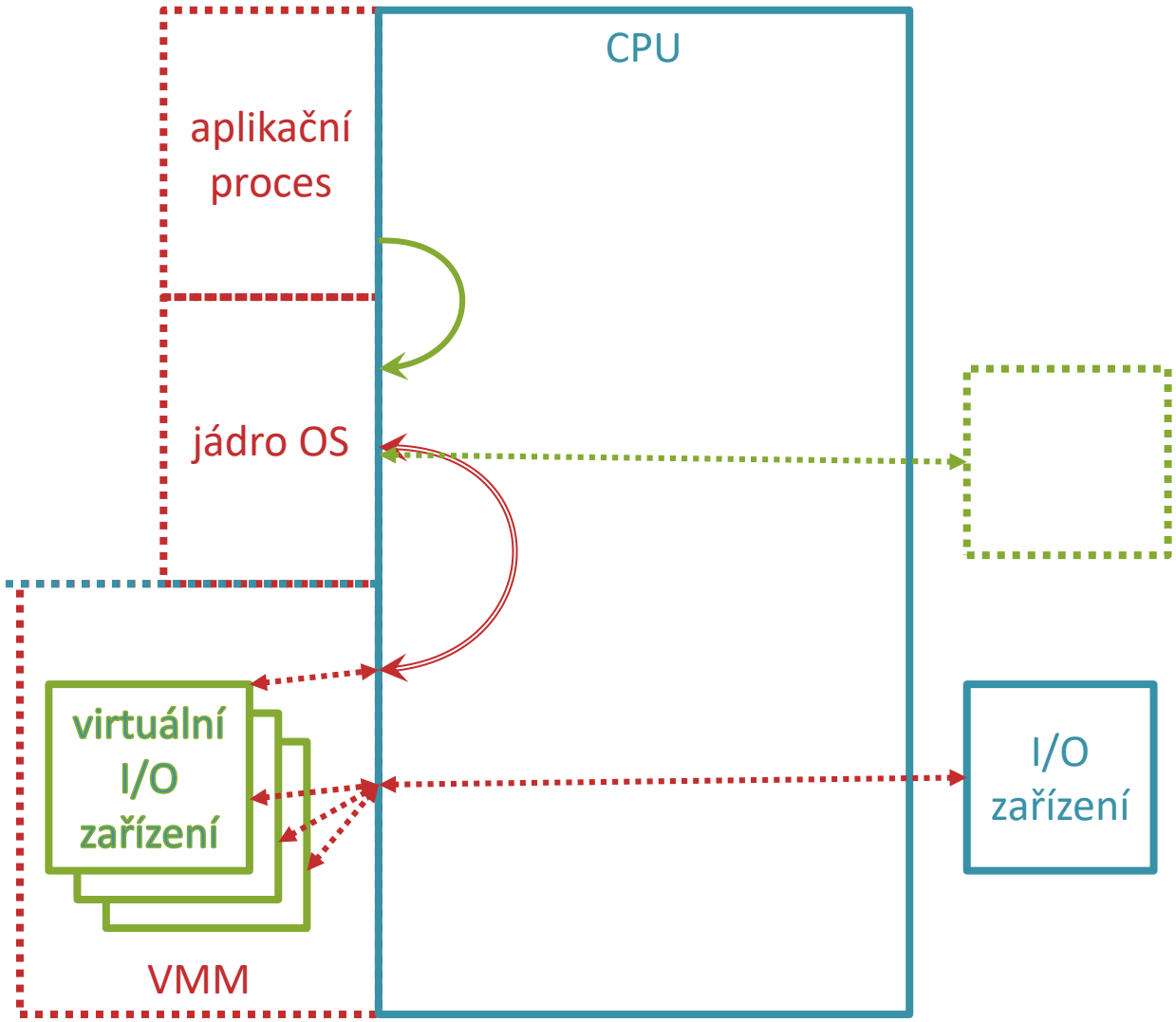
- Exkluzivní přístup
  - K danému zařízení může přistupovat pouze jeden virtuální stroj
  - Kromě samotného I/O zařízení je třeba zpřístupnit nebo virtualizovat systém přerušení, případně DMA

Přístup k fyzickému I/O zařízení na virtuálním CPU



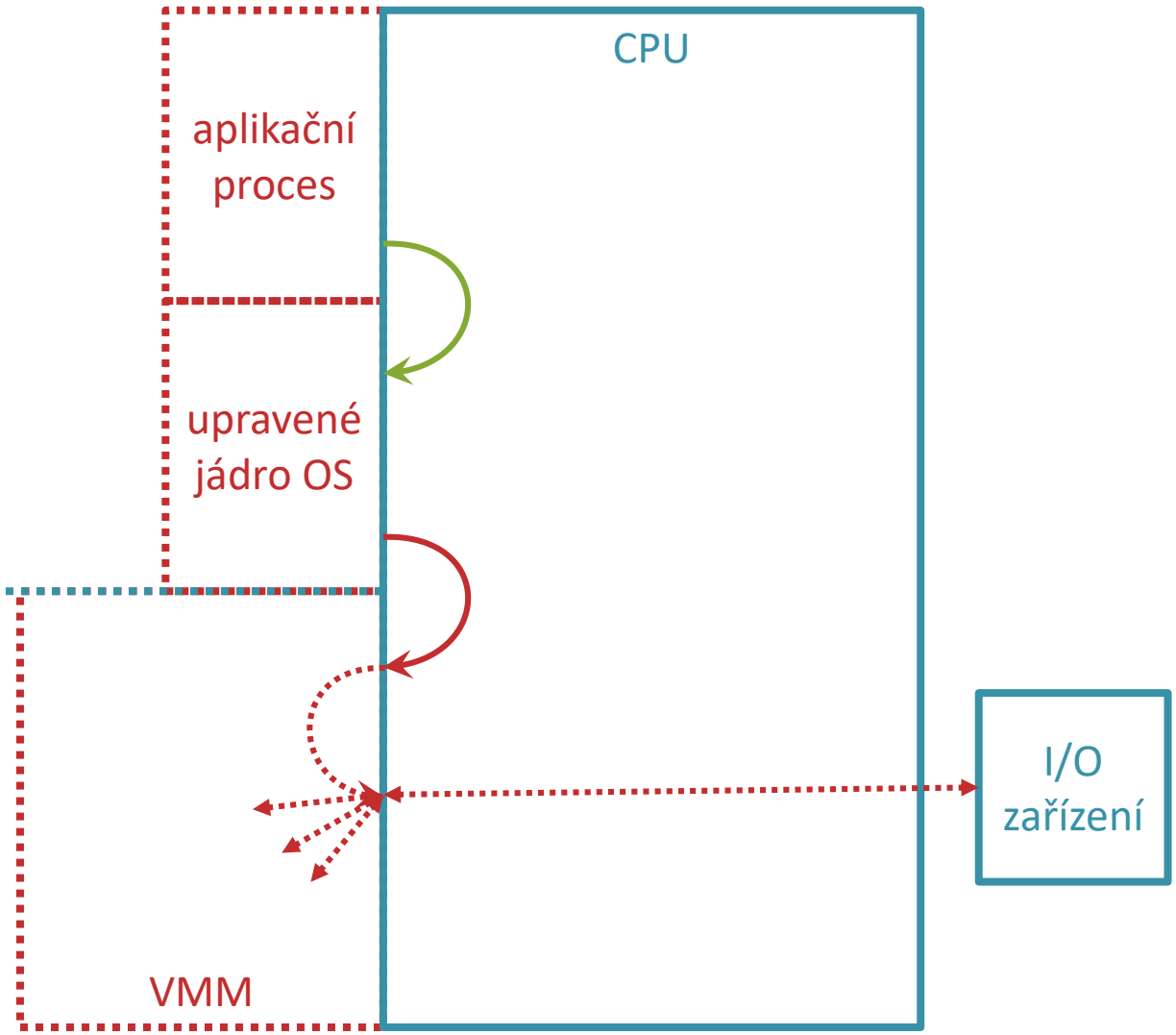
- I/O instrukce pro přístup k danému zařízení nejsou privilegované
- Vyžaduje konfigurovatelnost HW ochrany I/O prostoru
- Exkluzivní přístup
  - K danému zařízení může přistupovat pouze jeden virtuální stroj
  - Kromě samotného I/O zařízení je třeba zpřístupnit nebo virtualizovat systém přerušení, případně DMA - IOMMU

Přístup k virtuálnímu I/O zařízení na virtuálním CPU



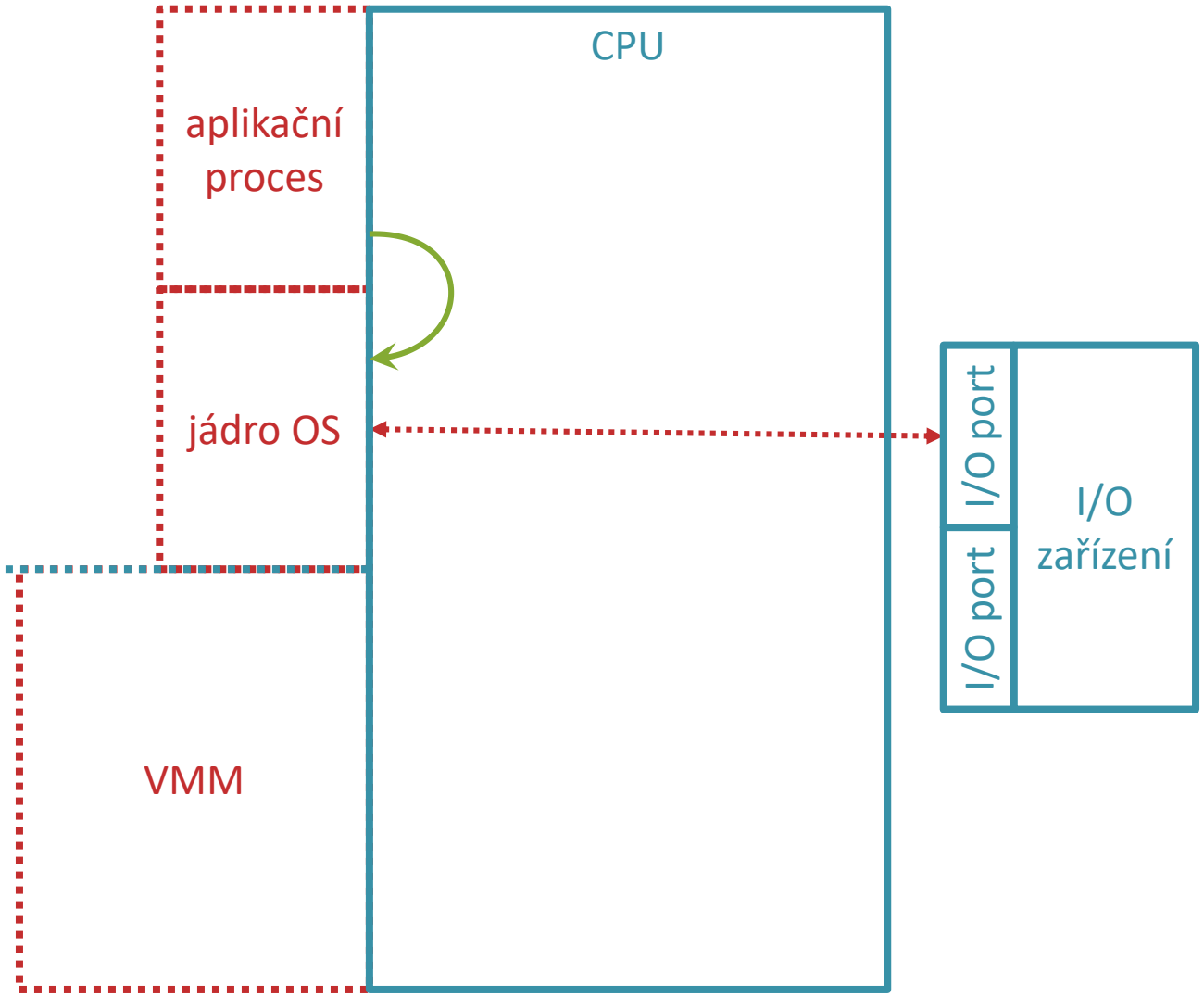
- Privilegované I/O instrukce resp. přístupy na paměťově mapované zařízení jsou emulovány VMM
- VMM pro každý virtuální stroj zvlášť emuluje chování hardware
- Daný typ hardware nemusí fyzicky existovat
- Sdílený přístup
  - VMM z emulovaného hardware extrahuje logické akce
  - Logické akce jsou prováděny fyzickým zařízením

Přístup k virtuálnímu I/O zařízení s úpravou OS



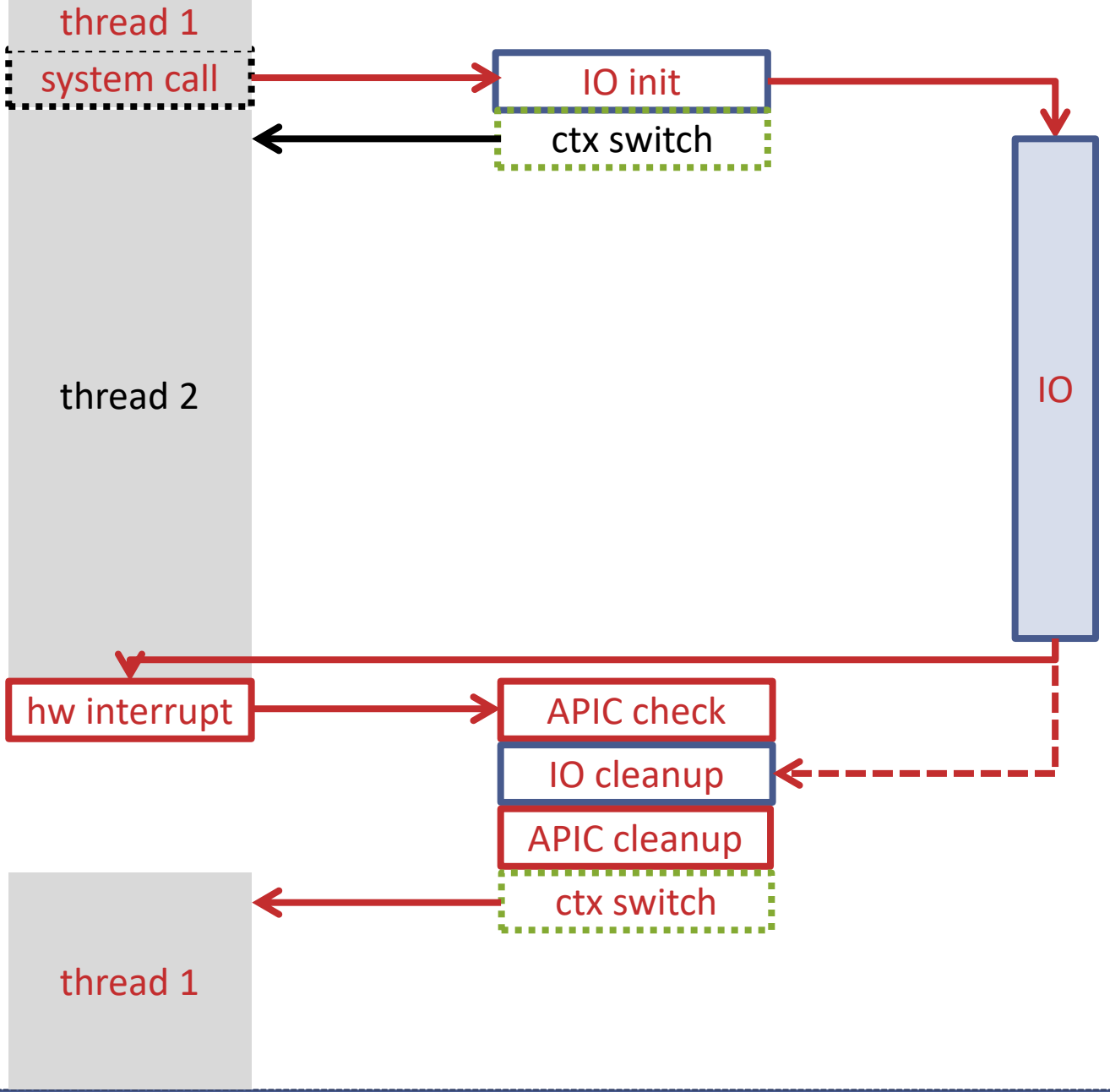
- Zásah do OS
  - Paravirtualizace: OS je výrazně upraven
  - Klasická virtualizace: do OS je přidán ovladač virtuálního zařízení
- Výhody
  - Mezi OS a VMM jsou předávány logické příkazy a nikoliv fyzické I/O
  - Předání nevyžaduje emulaci I/O instrukcí
  - Logických příkazů je méně
  - Serializace příkazů z různých VM je jednodušší

Přístup k fyzickému I/O zařízení na virtuálním CPU



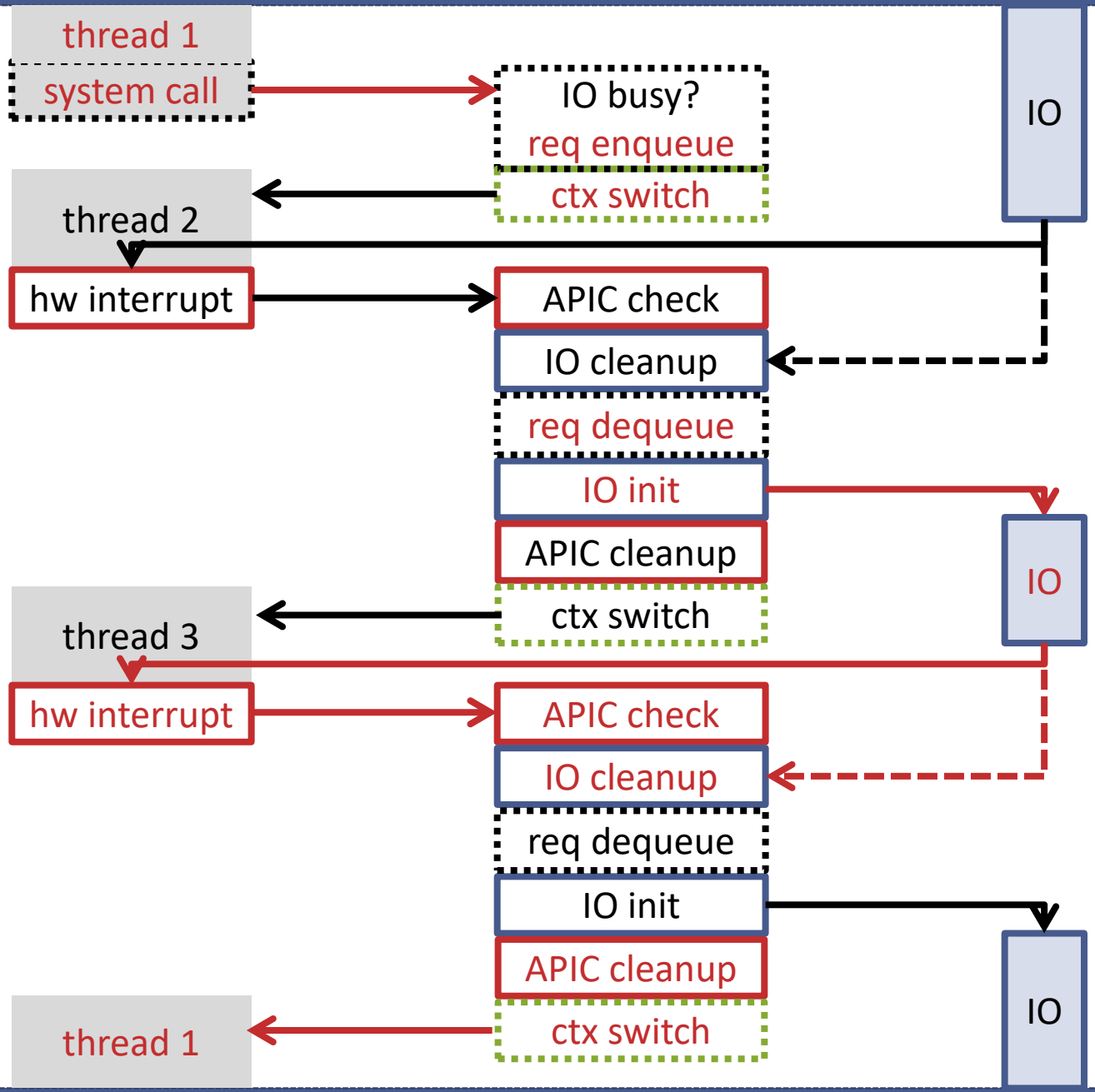
- I/O instrukce pro přístup k danému zařízení nejsou privilegované
- Vyžaduje konfigurovatelnost HW ochrany I/O prostoru
- Sdílený přístup
  - I/O zařízení se prezentuje vícekrát v I/O adresovém prostoru
  - I/O zařízení má pro každou adresu jednu kopii vnitřních stavových registrů
  - Kromě samotného I/O zařízení je třeba zpřístupnit nebo virtualizovat systém přerušení, případně DMA - IOMMU

Obsluha IO požadavku v OS bez virtualizace - zjednodušeno



- Aplikační proces volá jádro
- Jádro nastartuje IO operaci
  - I/O instrukce
- Po dobu čekání běží jiné aplikační vlákno
- Dokončení operace je signalizováno přerušením
  - To putuje od IO zařízení přes APIC
    - (terminologie x86: advanced programmable interrupt controller)
- Obsluha přerušení začíná zkoumáním důvodu
  - APIC sdružuje různé zdroje přerušení
- Jádro testuje úspěšnost operace
  - I/O instrukce
  - Samotný výsledek I/O bývá v paměti
- Ukončení obsluhy přerušení se hlásí APICu
  - Ochrana před rekurzí

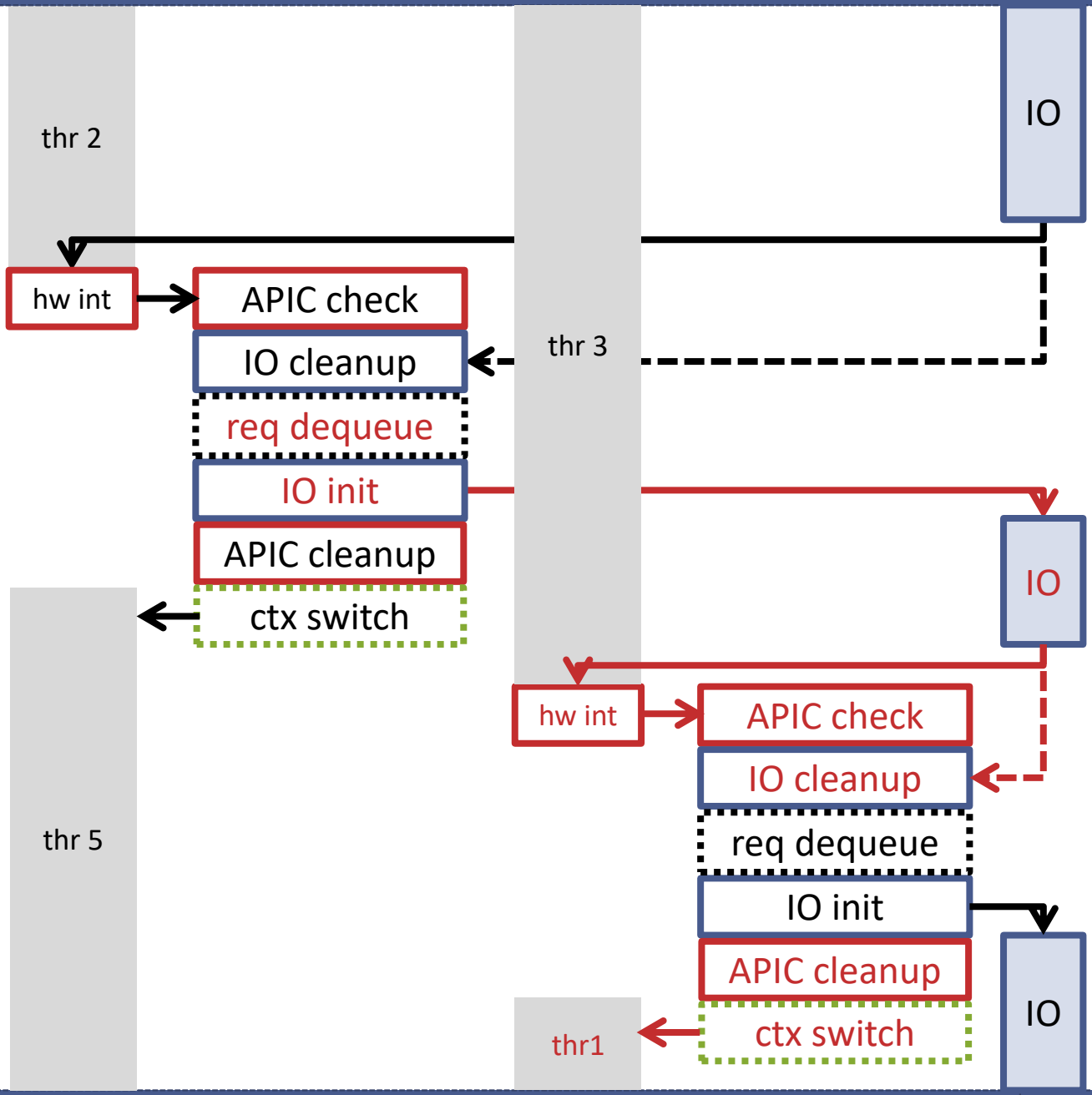
Obsluha IO požadavku v OS bez virtualizace (1 CPU)



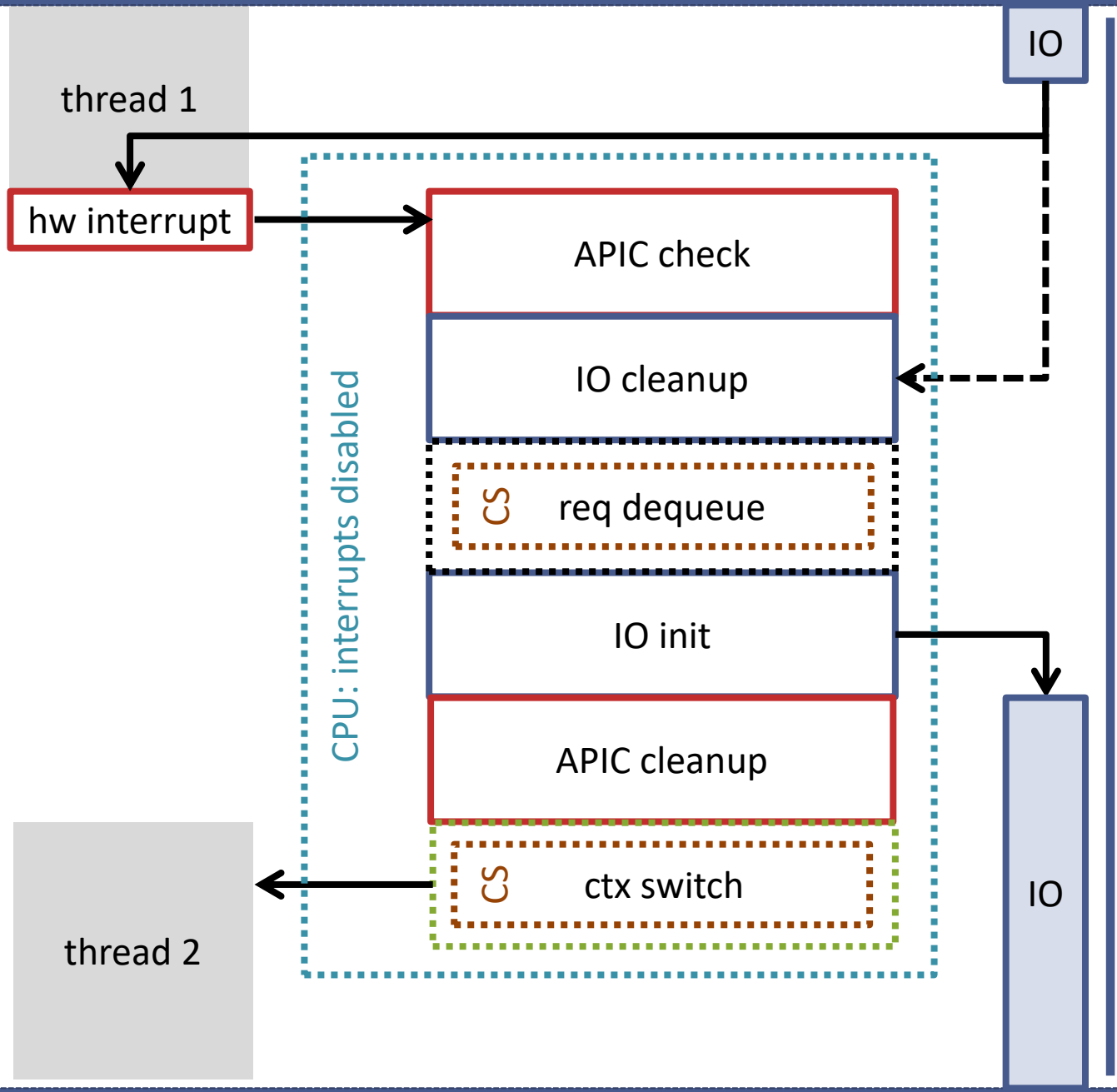
- V reálném případě je IO zařízení často obsazeno
  - Požadavky čekají ve frontě organizované jádrem OS
  - Obsluha přerušení typicky dokončuje starý požadavek a startuje nový
- Zdrojů přerušení je víc než signálů, které vedou k CPU
  - Obsluha jednoho přerušení občas řeší více zdrojů přerušení najednou

Obsluha IO požadavku v OS bez virtualizace (více CPU)

- APIC rozděluje přerušení na CPU
  - často cyklicky
- IO požadavek je dokončen na jiném CPU než vznikl
  - nevýhodné z hlediska cache
  - alternativy:
    - zařadit připravené vlákno do fronty původního CPU
    - přerušit ihned původní CPU (pomocí APIC)

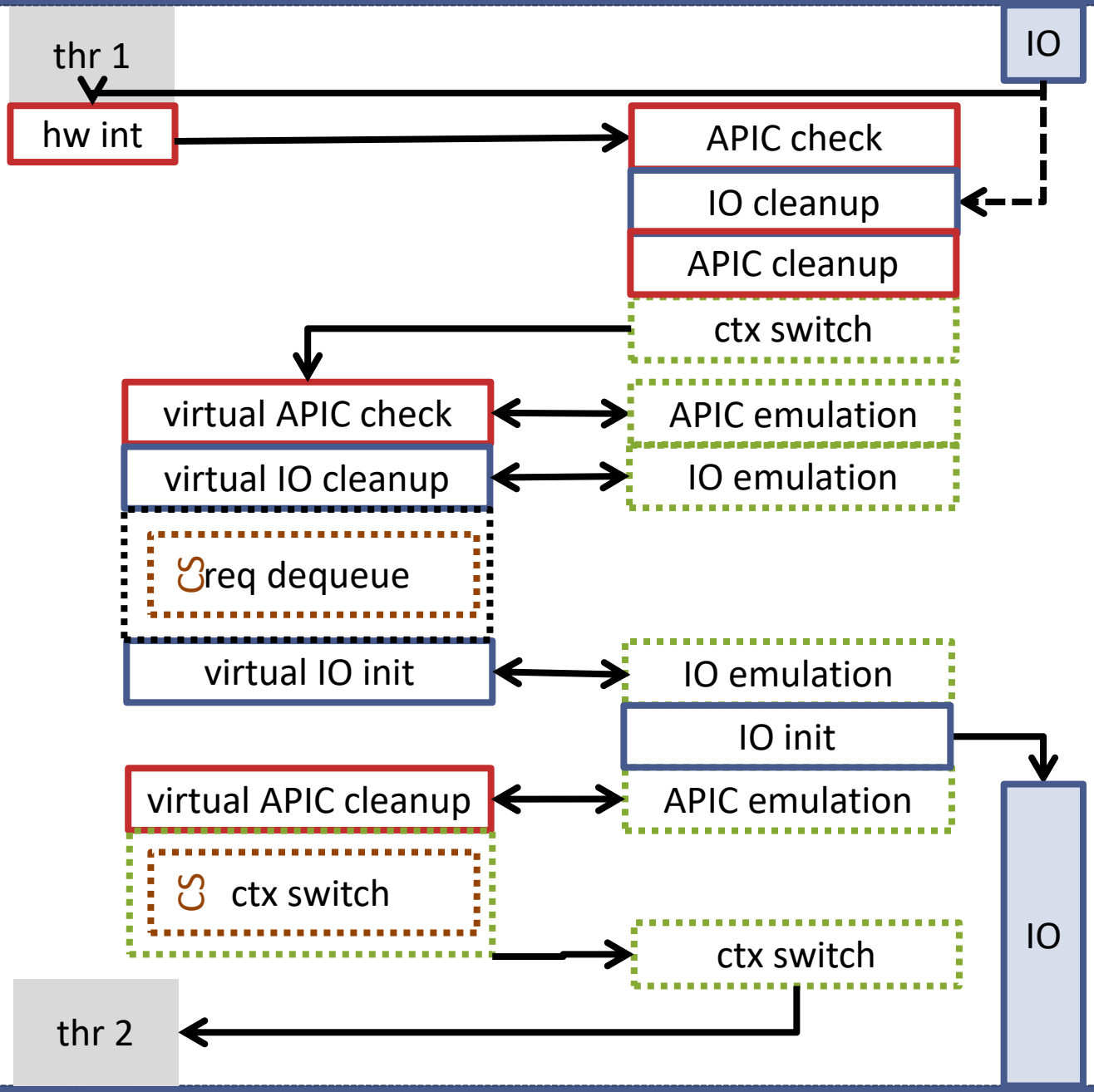


Obsluha přerušení v OS bez virtualizace (více CPU)



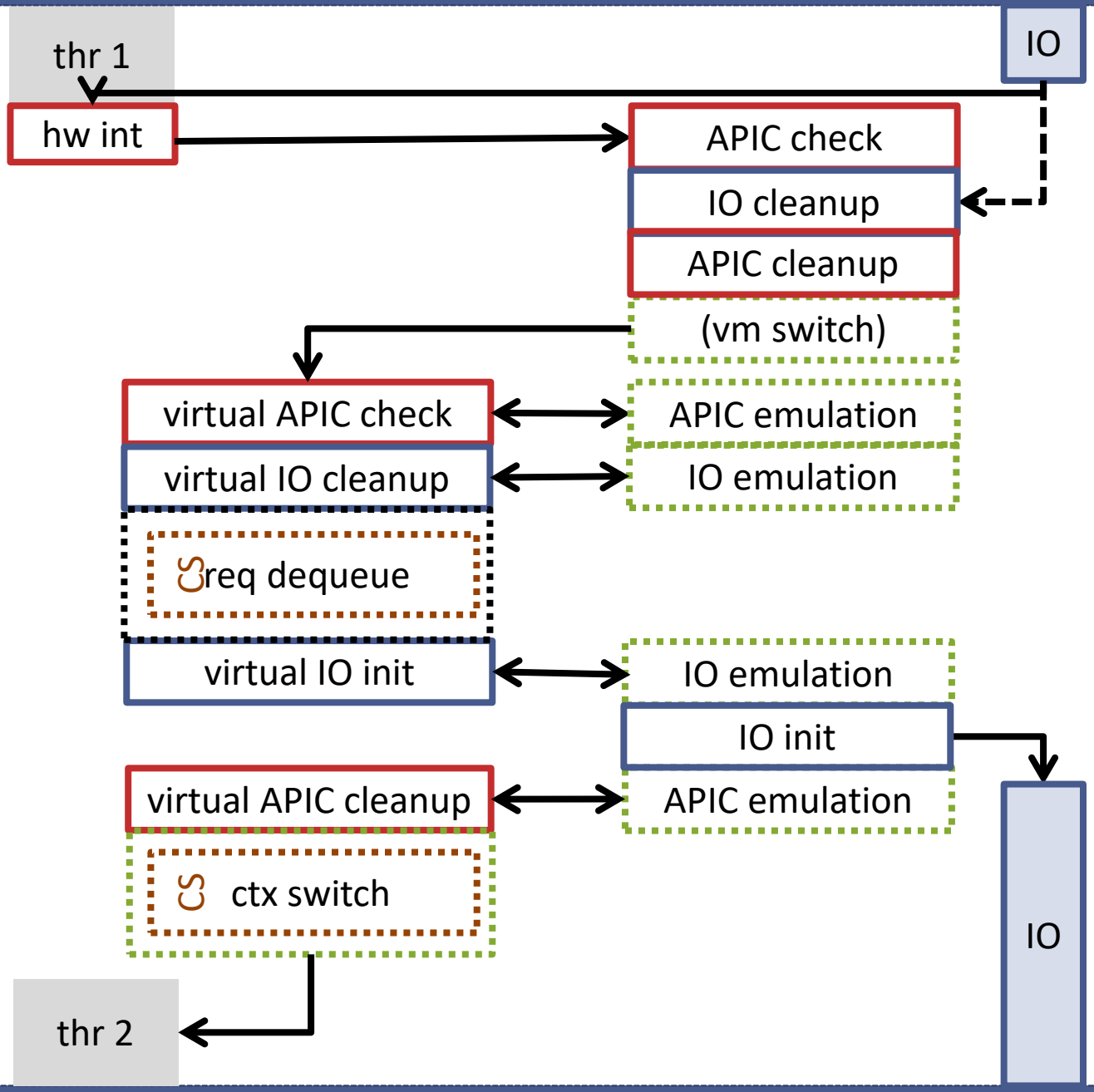
- CPU obvykle automaticky zakáže vnější přerušení při vstupu do jeho obsluhy
  - Na jiných CPU ale přerušení zakázána nejsou
- Datové struktury jádra musejí být chráněny
  - CS - Spinlock

Obsluha přerušení v OS s virtualizací (bez HW podpory virtualizace)



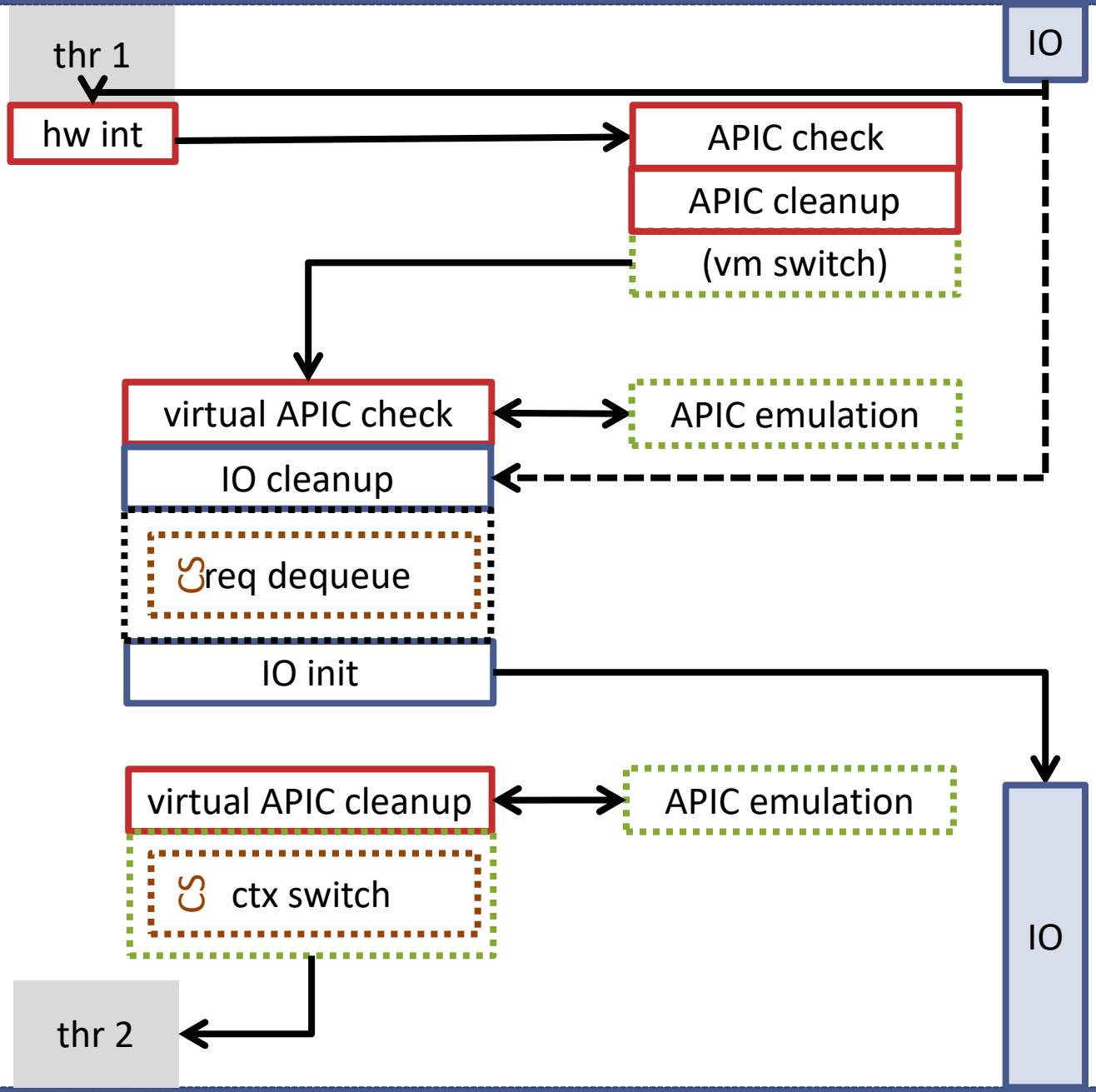
- Jádro OS běží v aplikačním režimu
  - Přerušení jsou fyzicky povolena
  - VMM je musí odložit na přípustný okamžik
- Interakce s IO a APIC musí být virtualizována
- Kritické sekce mohou být přerušeny fyzickým přerušením a přeplánovány
  - Trvají nepřípustně dlouho
  - Ostatní virtuální CPU jsou zablokovány ve spinlocku - aktivně!

Obsluha přerušení v OS s virtualizací (s root/non-root režimy)



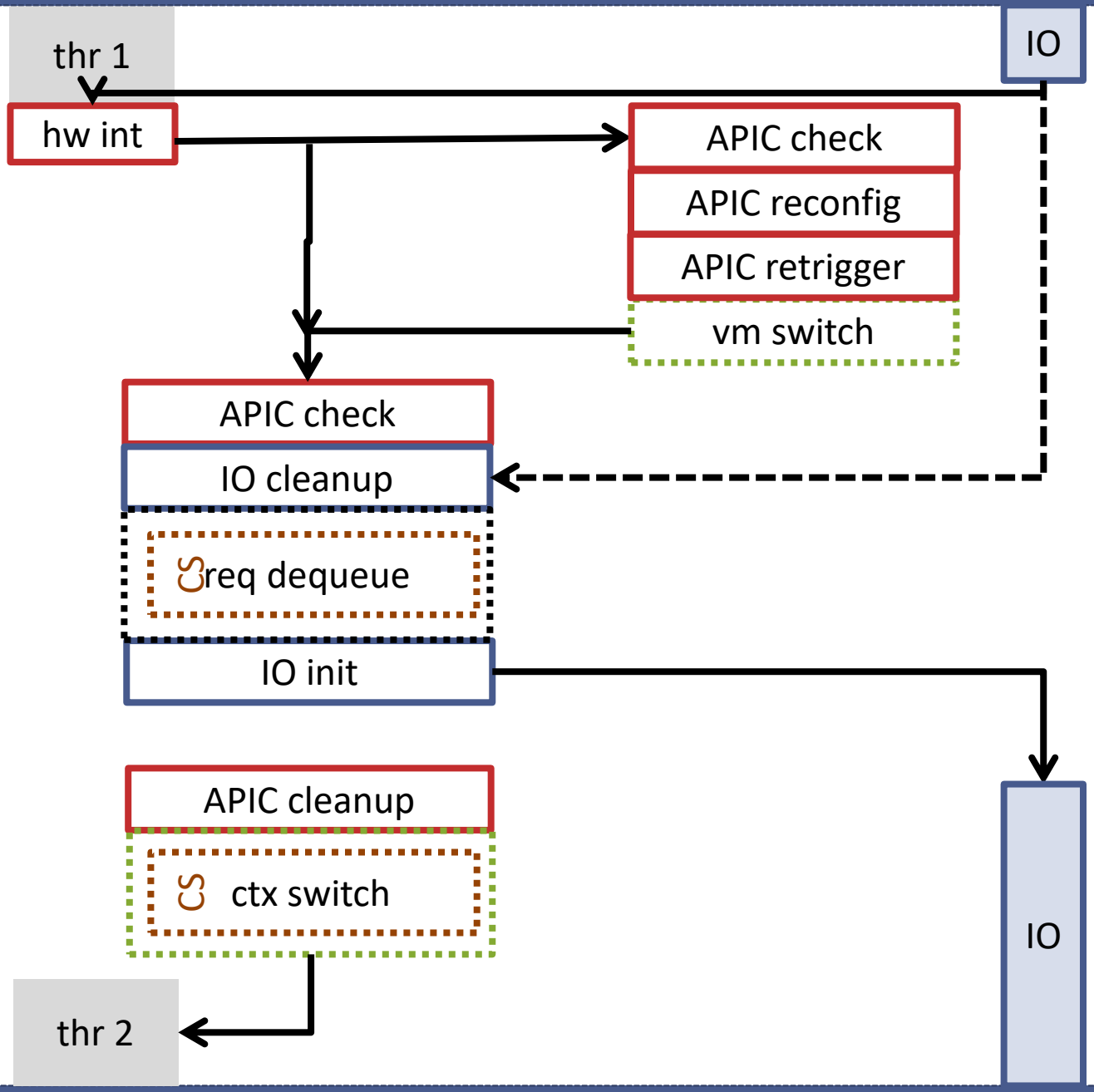
- Jádro OS běží v non-root režimu
  - Přerušení jsou fyzicky povolena, CPU je řeší jako VM-Exit
  - VMM je musí poslat správnému VM
- Interakce s IO a APIC musí být virtualizována

Obsluha přerušení v OS s virtualizací (s root/non-root a VMDQ)



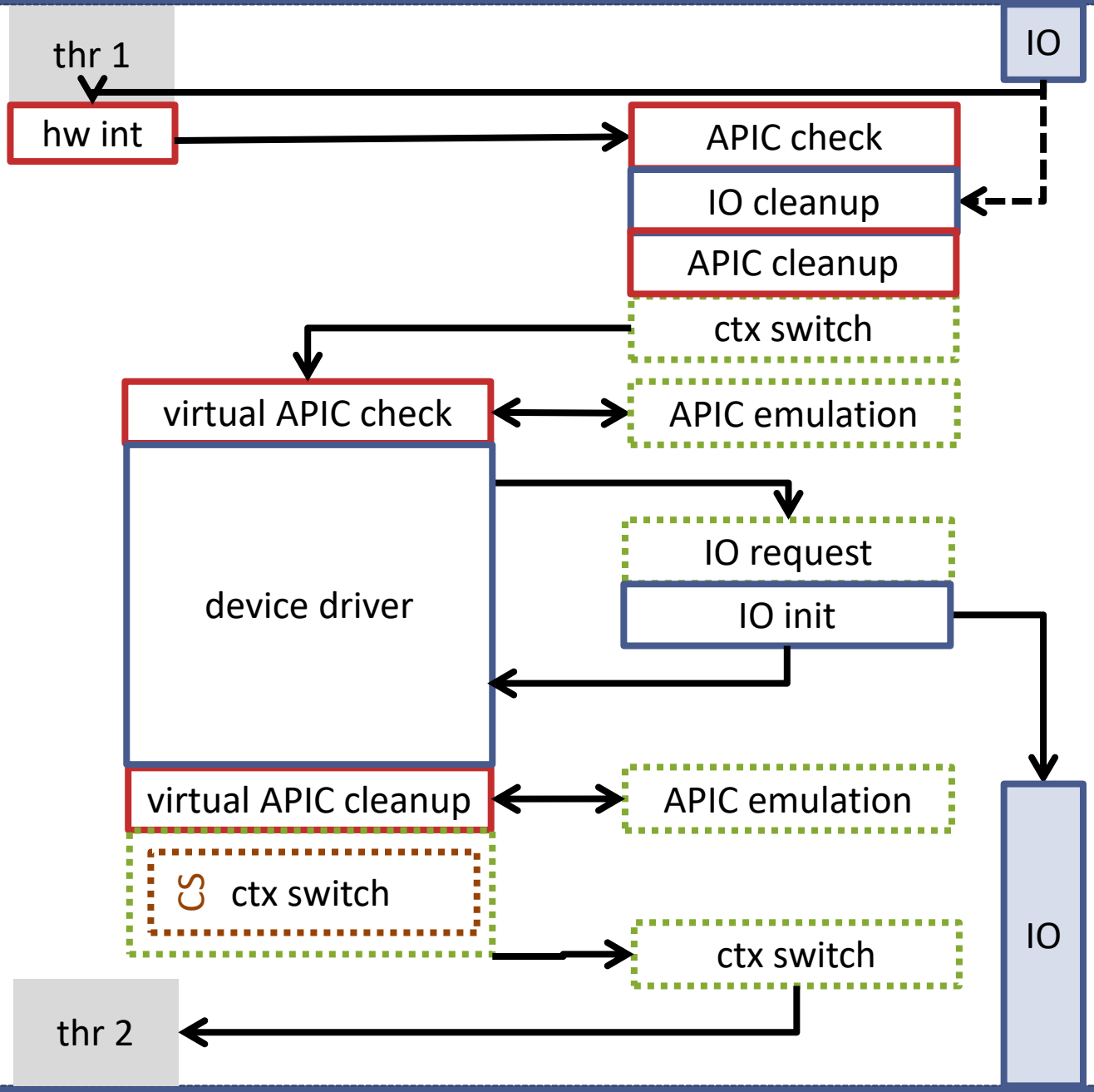
- Jádro OS běží v non-root režimu
  - Přerušení jsou fyzicky povolena, CPU je řeší jako VM-Exit
  - VMM je musí poslat správnému VM
- IO je přímo zpřístupněno v exklusivním režimu (VMDQ)
- APIC stále musí být virtualizován

Obsluha přerušení (s root/non-root, VMDQ a FlexPriority)



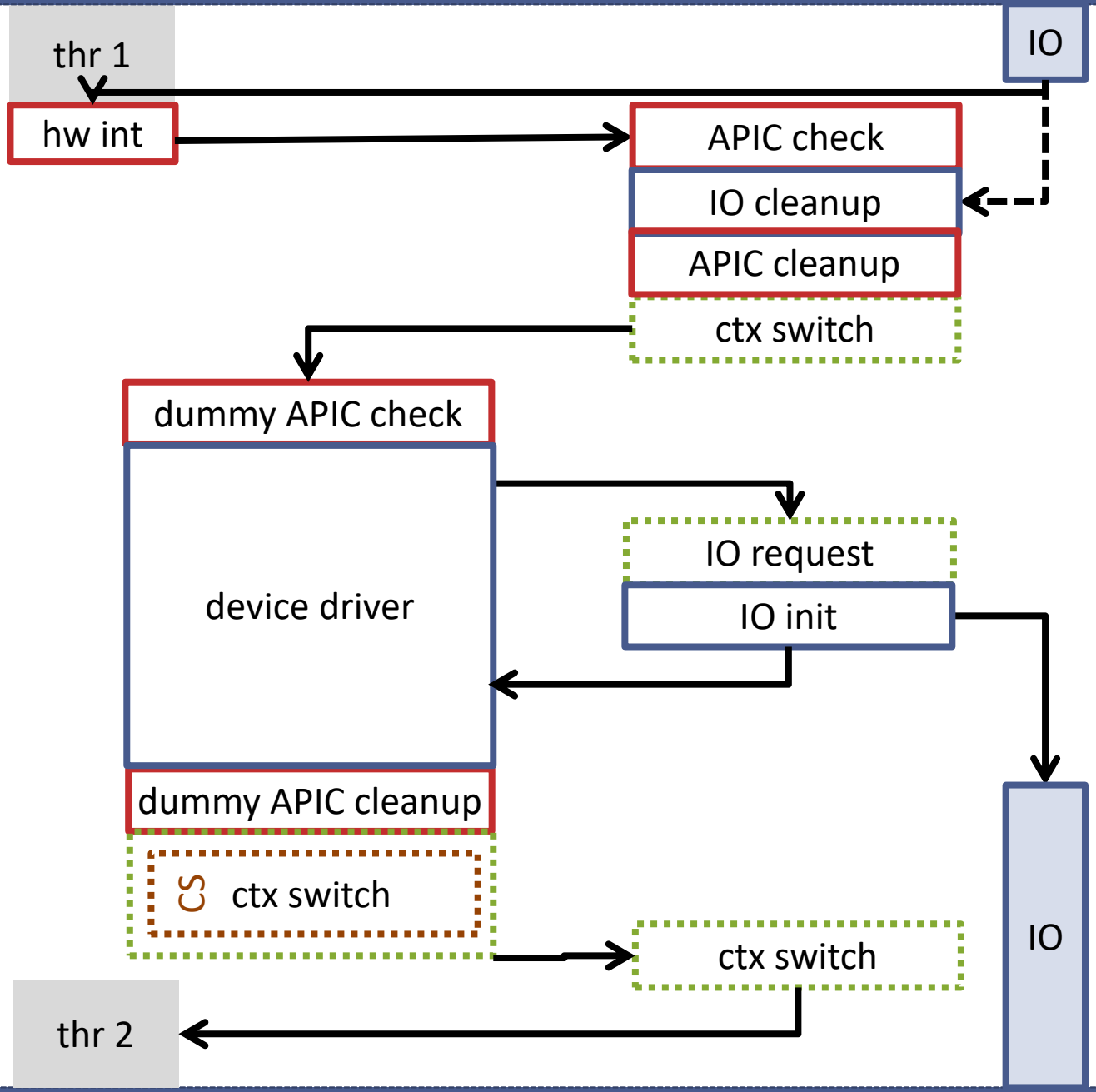
- IO je přímo zpřístupněno v exklusivním režimu (VMDQ)
- Virtualizace APIC je řešena hardwarově
  - Přerušení od zařízení obsluhovaných exkluzivně právě běžící VM jsou v APIC/CPU konfigurovány tak, že nevyvolávají VM exit – obsluhuje je jádro OS
  - Ostatní přerušení VM exit vyvolávají
  - Při přepnuví VM musí VMM rekonfigurovat APIC

Obsluha přerušení v OS s ovladačem virtuálního zařízení



- Instalace speciálního ovladače virtuálního zařízení do OS
  - Vyřeší emulaci samotného IO zařízení
  - Nevyřeší emulaci APIC a režii přepínání kontextu
    - Tyto části bývají společné všem zařízením a nelze je vyměnit bez zásahu do OS
  - Problém spinlocku zůstává

Obsluha přerušení v OS s ovladačem virtuálního zařízení a FlexPriority

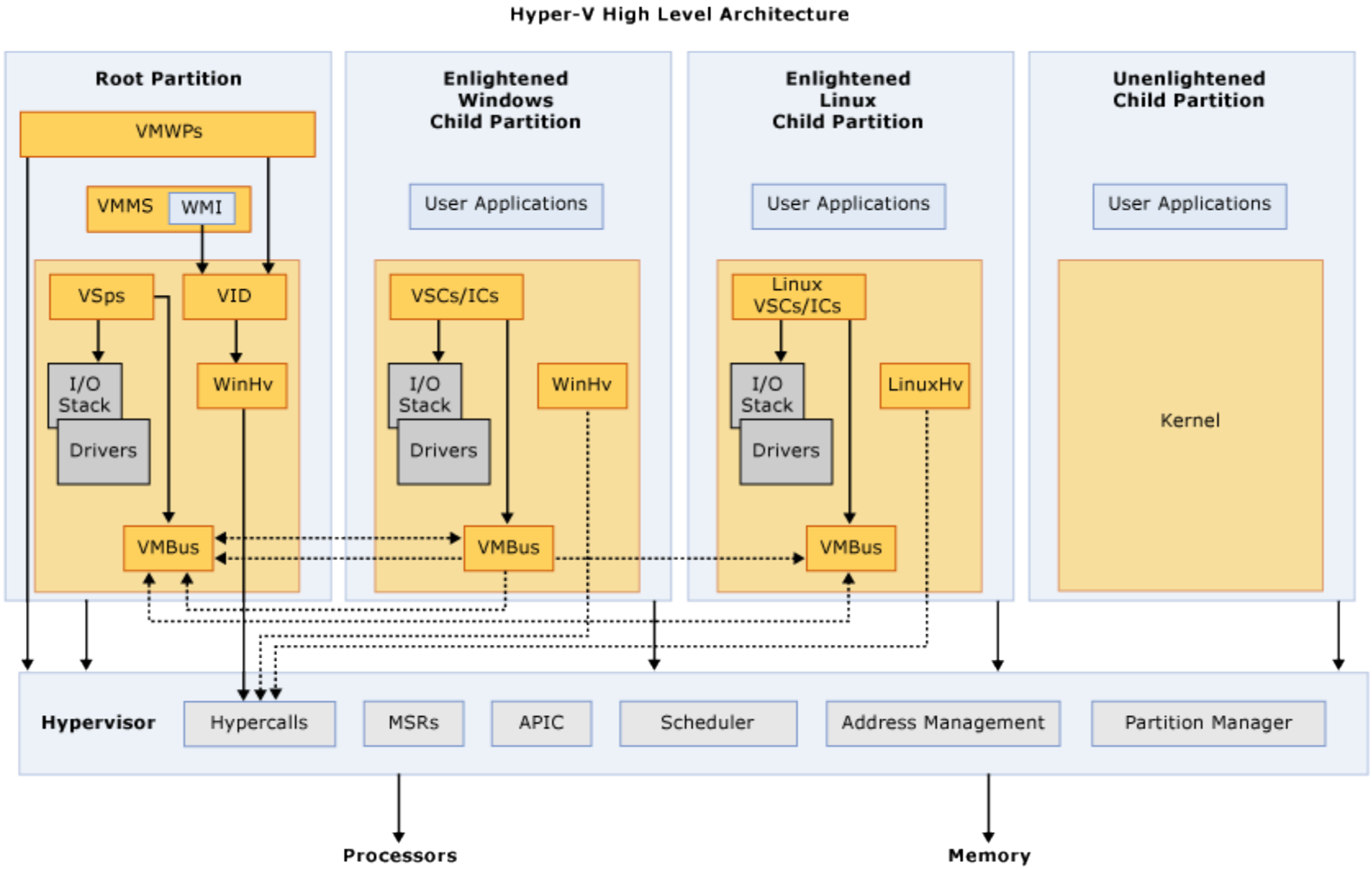


- Instalace speciálního ovladače virtuálního zařízení do OS
  - Vyřeší emulaci samotného IO zařízení
  - Emulaci fiktivního APIC zařídí přímo HW

Plánování vláken v OS i ve VMM

- Každé asynchronní (I/O) i synchronní (syscall) přerušení může způsobit oživení vlákna - co s ním?
  - Přerušené vlákno může také zůstat živé
  - A) přerušené vlákno má přednost před oživeným
    - Počítající vlákna prakticky znemožní běh komunikujících
      - 1 obrátka komunikace za časové kvantum preemptivního plánovače
  - B) oživené vlákno dostane přednost před přerušeným
    - Vlákna intenzivně provádějící I/O potlačí běh nekomunikujících
  - C) něco mezi tím
    - Unix: Dynamické priority
- Jádro OS předpokládá známý počet stále běžících CPU
  - To při virtualizaci neplatí
  - Iluze většího počtu CPU je vytvářena preemptivním plánovačem ve VMM
    - Perioda plánovače rozhoduje o efektivitě komunikujících virtuálních CPU

VM-VMM Communication (Example: Microsoft Hyper-V)


Hyper-V High Level Architecture
Root Partition
VMWPs
VMMS
WMI
VSps
VID
I/O Stack
Drivers
WinHv
VMBus
Enlightened Windows Child Partition
User Applications
VSCs/ICs
I/O Stack
Drivers
WinHv
VMBus
Enlightened Linux Child Partition
User Applications
Linux VSCs/ICs
I/O Stack
Drivers
LinuxHv
VMBus
Unenlightened Child Partition
User Applications
Kernel
Hypervisor
Hypercalls
MSRs
APIC
Scheduler
Address Management
Partition Manager
Processors
Memory

Guest-to-Hypervisor Interface (Microsoft Hyper-V)

- Virtual MSRs
  - Physical MSRs used by Kernels to read/alter CPU configuration
  - VMM emulates additional Machine Status Registers (MSR) not present in HW
    - VMM-aware VM Kernel can read/write virtual MSRs to exchange configuration information with VMM
  - Emulation too slow for real communication
- Hypercall
  - Call Hypervisor from Guest (privileged mode)
  - Exposed as procedure call to a special guest-physical page
    - Provided by Hypervisor on request from Guest (via a virtual MSR)
    - VM Kernel must map the guest-physical page to a guest-virtual page
    - The page contains either special instructions or nothing – both cases cause VM exit
  - Arguments passed/returned in registers or VPAP
- Virtual Processor Assist Page (VPAP)
  - Special guest-physical page per virtual processor (core/logical thread)
    - Both Hypervisor and Guest can read/write

Guest-to-Hypervisor Interface (Microsoft Hyper-V)

- Hypercall
  - Call Hypervisor from Guest (privileged mode)
  - Exposed as procedure call to a special guest-physical page
  - Arguments passed/returned in registers or VPAP
  - One Hypercall may serve several logical requests
    - Chained into an array of arguments
  - All Hypercalls return within 50 microseconds
    - Avoids blocking in the Hypervisor (giant lock?)
    - Longer requests serviced in continuation-style
      - The Hypercall return address is set before the instruction that invoked it
      - Arguments adjusted to indicate that part of the request is already done
      - On the next VM Entry, the Hypercall is entered again

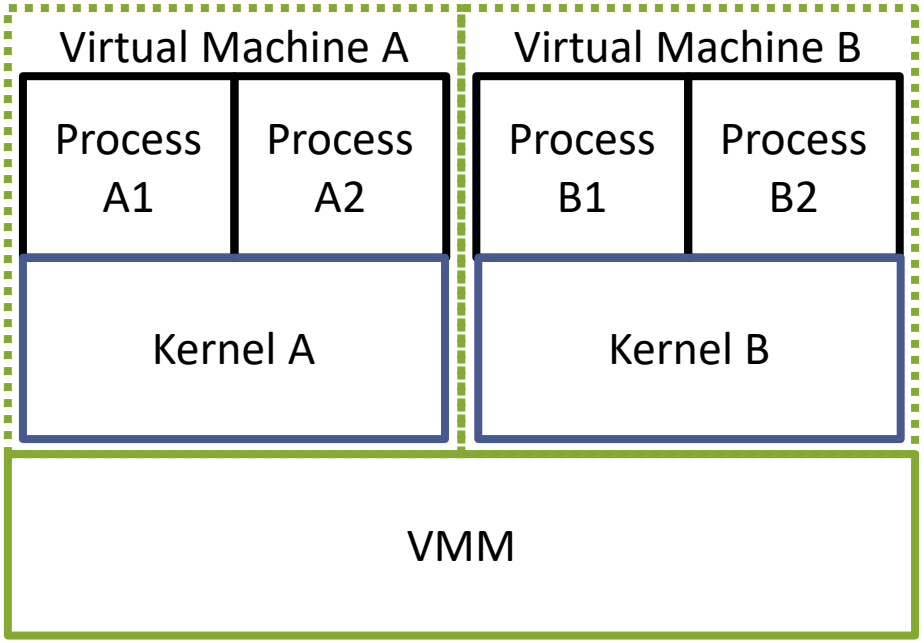
Guest-to-Hypervisor Interface (Microsoft Hyper-V)

- Partition
  - A set of virtual processors and other hardware, plus its configuration
  - Root partition – typically used to run the Host OS and VM Management
- Inter-partition messaging
  - The hypervisor supports a simple message-based inter-partition communication mechanism.
  - Messages can be sent by the hypervisor to a partition or can be sent from one partition to another.
- Guest Physical Address Space
  - The GPA mappings are defined by the partition's parent.
    - At the time they are mapped, they are specified in terms of the parent's GPA space.
- Guest Virtual Address Space
  - The hypervisor exposes operations to flush the TLB (on one virtual processor).

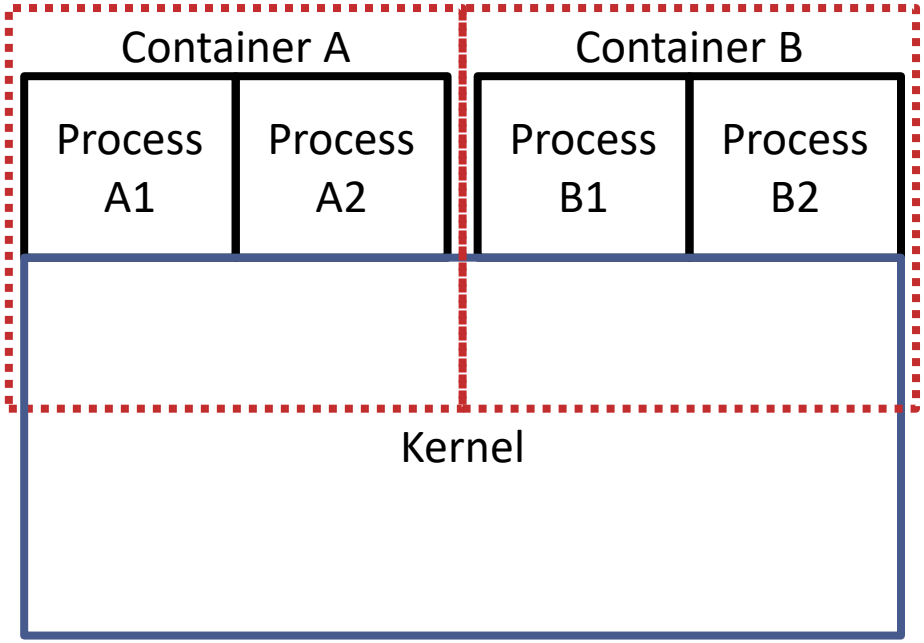
Containers (Linux)

Virtual Machines vs. Containers

Virtual Machines

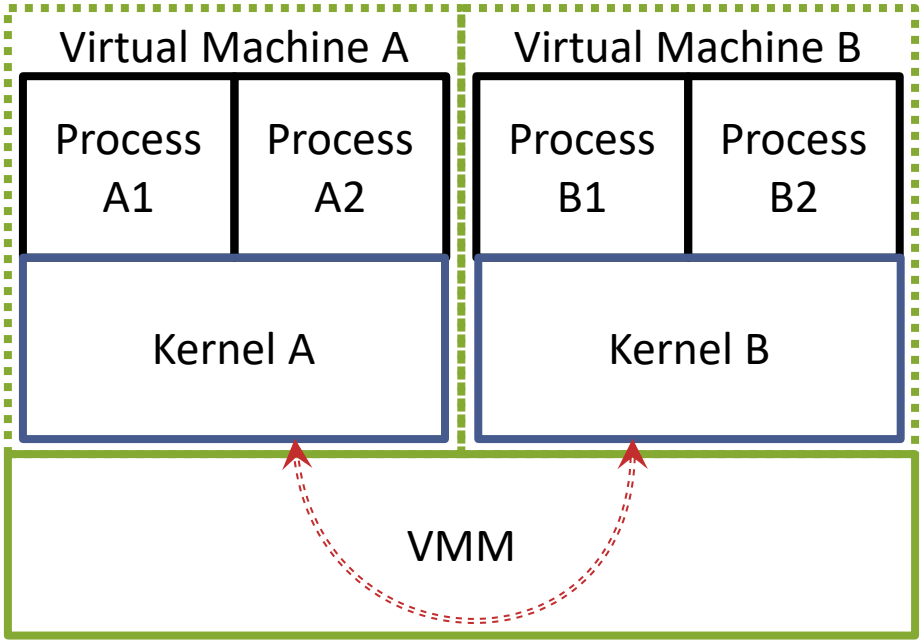


Containers



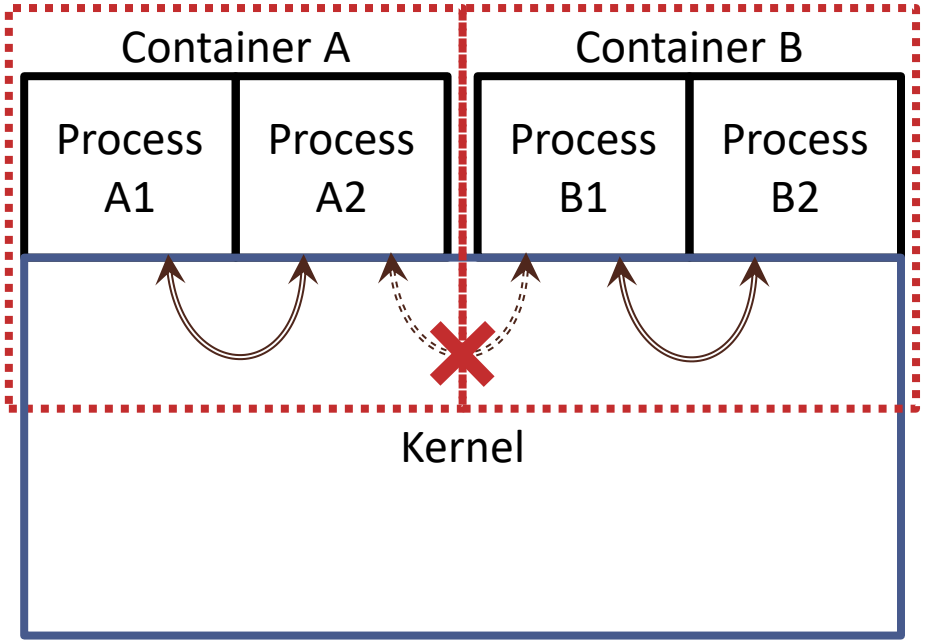
Virtual Machines vs. Containers

Virtual Machines



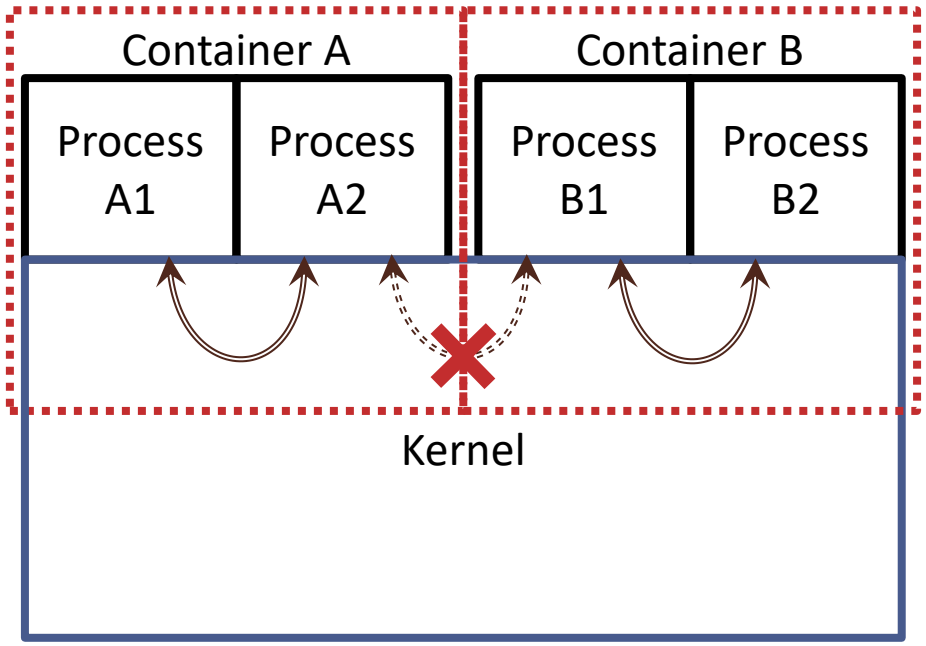
- Inherent safety
  - Kernel-HW interface was not designed for Kernel-Kernel communication
  - VMM adds well-controled holes into a natural barrier

Containers



- Limited safety
  - Process-Kernel interface was designed for Process-Process communication
  - Containerization requires blocking existing communication channels

Containers in Linux



- Linux Kernel technologies
  - Namespaces (2002,2006,2012)
    - Entity identifiers are no longer global
    - Parent process controls separation of child namespaces
  - cgroups (2008,…)
    - Accounting and limiting access to resources
    - Summed across a group of processes
  - UnionFS (2004,2014)
    - Combining filesystems
      - Overlaid instead of disjoint mounts
    - Controlled sharing of files among containers
    - Snapshots etc.

Containers in Linux

- Namespaces (2002,2006,2012)
  - Entity identifiers are no longer global
  - Parent process controls separation of child namespaces

```
clone(…,flags,…)
```

```
unshare(flags)
```

  - mnt – filesystem mounts
  - pid – visible processes
  - net – network interfaces
  - ipc – shared-memory etc.
  - uts – host/domain names
  - user – user/group ids
  - cgroup – resource groups

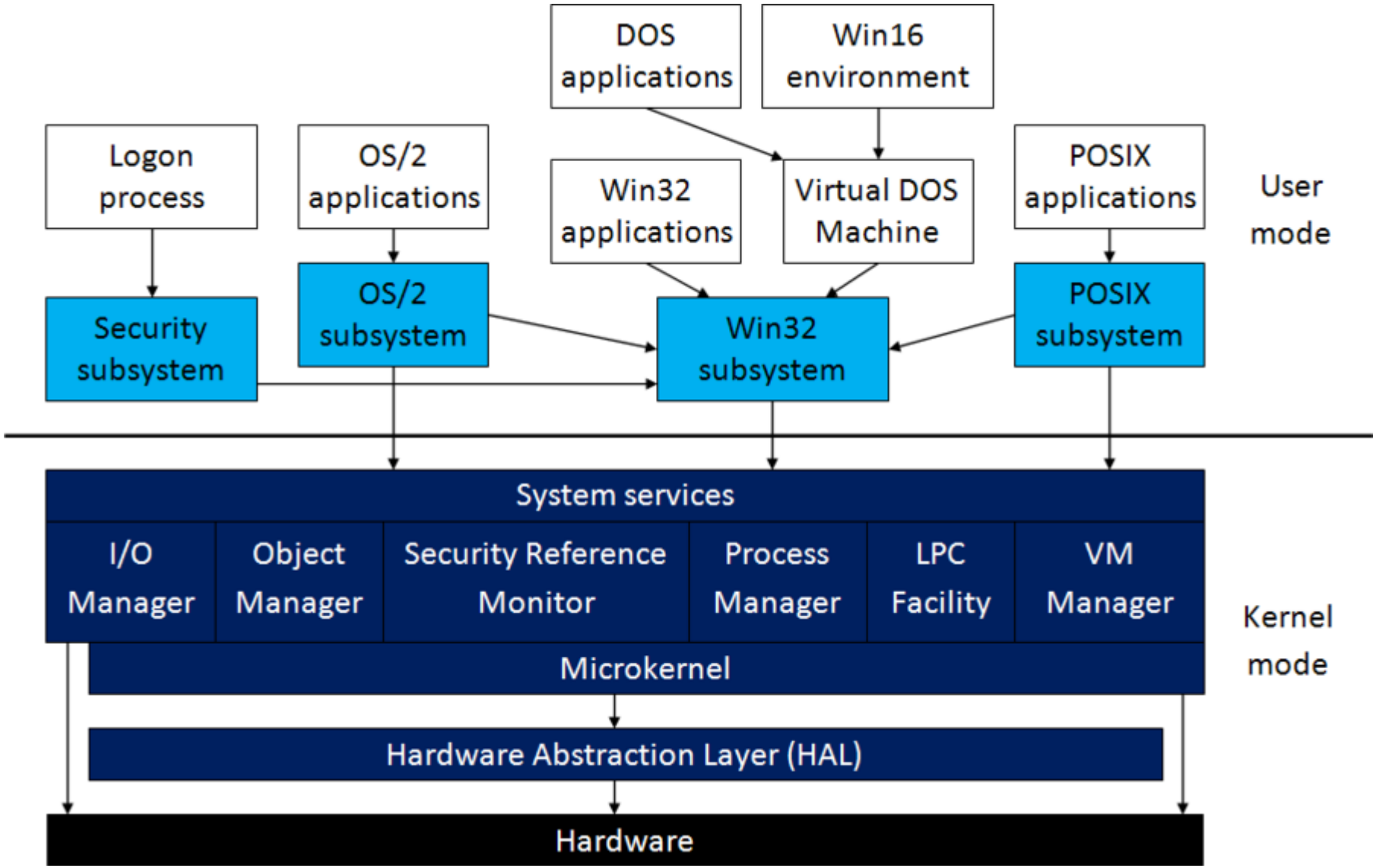
- cgroups (2008,…)
  - Accounting and limiting access to resources
  - Summed across a group of processes
    - Memory
    - CPU time
    - I/O throughput (disk, network)

- UnionFS (2004,2014)
  - Combining filesystems
    - Overlaid instead of disjoint mounts
  - Controlled sharing of files among containers
  - Snapshots etc.

- Docker
  - Creation of Images
    - Sequence of commands producing a container
    - Commands add layers
      - Filesystems combined via UnionFS
      - Network interfaces, virtual networks, port mappings etc.
      - Other namespace and resource-related properties
    - Versioning, uploading, sharing, …
  - Starting Images as Containers
    - Running containers may save space by sharing (read-only) parts of filesystems
      - In theory, they could reuse the underlying physical filesystem (if distro versions match)
      - In reality, the status of the physical filesystem is unknown – containers prefer their own
  - Controlling running Containers
    - Using kernel namespaces, cgroups
    - Snapshots create new Images

Containers (Windows)

Microsoft Windows NT 3.1 (1993)



Containers in Windows

- (Windows) NT kernel was created to support several kinds of apps
  - (IBM) OS/2
  - (Microsoft) Windows 3.1 (binary compatible with non-NT “kernels”)
  - Legacy 16-bit Windows and DOS
  - POSIX
- The NT kernel always included support for namespace isolation and resource limiting
  - In limited use before 2016
- Windows Subsystem for Linux (WSL, bash.exe) – 2016
  - Emulates Linux syscalls on a Windows kernel
    - Does not emulate Linux namespaces and cgroups
- Windows Containers – 2016
  - Existing kernel technology polished
  - A fork of Docker used to create images and control the containers
  - Two modes of container execution
    - Windows Server Containers
    - Hyper-V Isolation

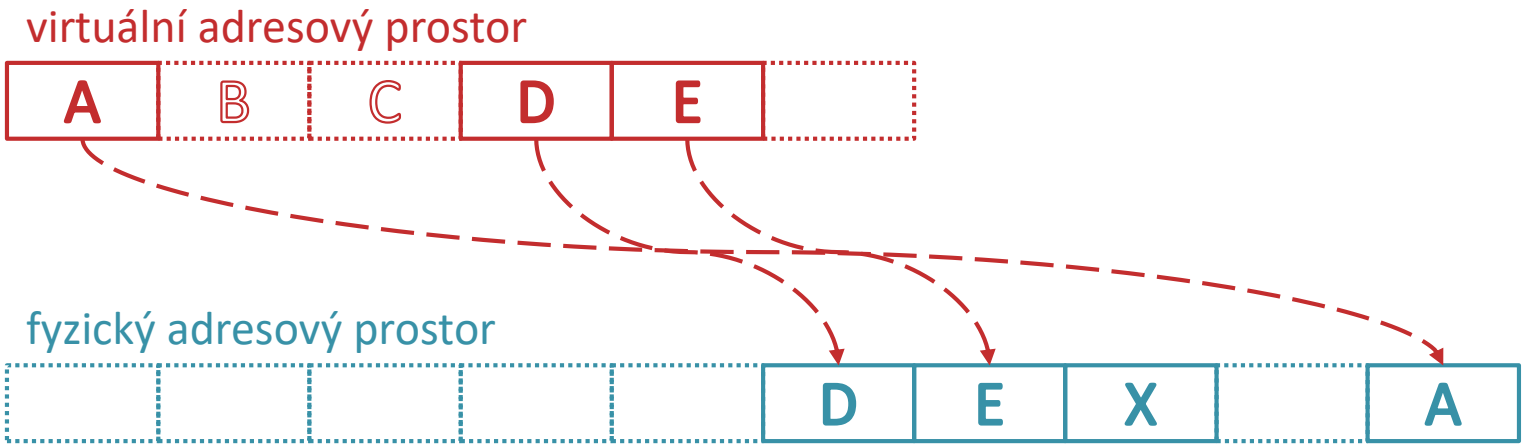
- Windows Containers – 2016
  - A fork of Docker used to create images and control the containers
    - Several formats of images, including Linux-Docker format
  - Two modes of container execution, both controlled by Docker
  - Windows Server Containers
    - 64-bit Windows processes isolated in containers
    - Available only on Windows Server 2016
    - Shares the Windows Server kernel
    - Architecturally analogous to Linux Containers
  - Hyper-V Isolation
    - Virtual machine created to run the container
    - The VM may run
      - a paravirtualized version of Windows Server kernel
        - used to execute Windows Server Containers on Windows 10
      - LCOW: a Linux-compatible kernel to run a Linux Image

Virtualizace virtuální paměti

Virtuální paměť ve fyzickém počítači

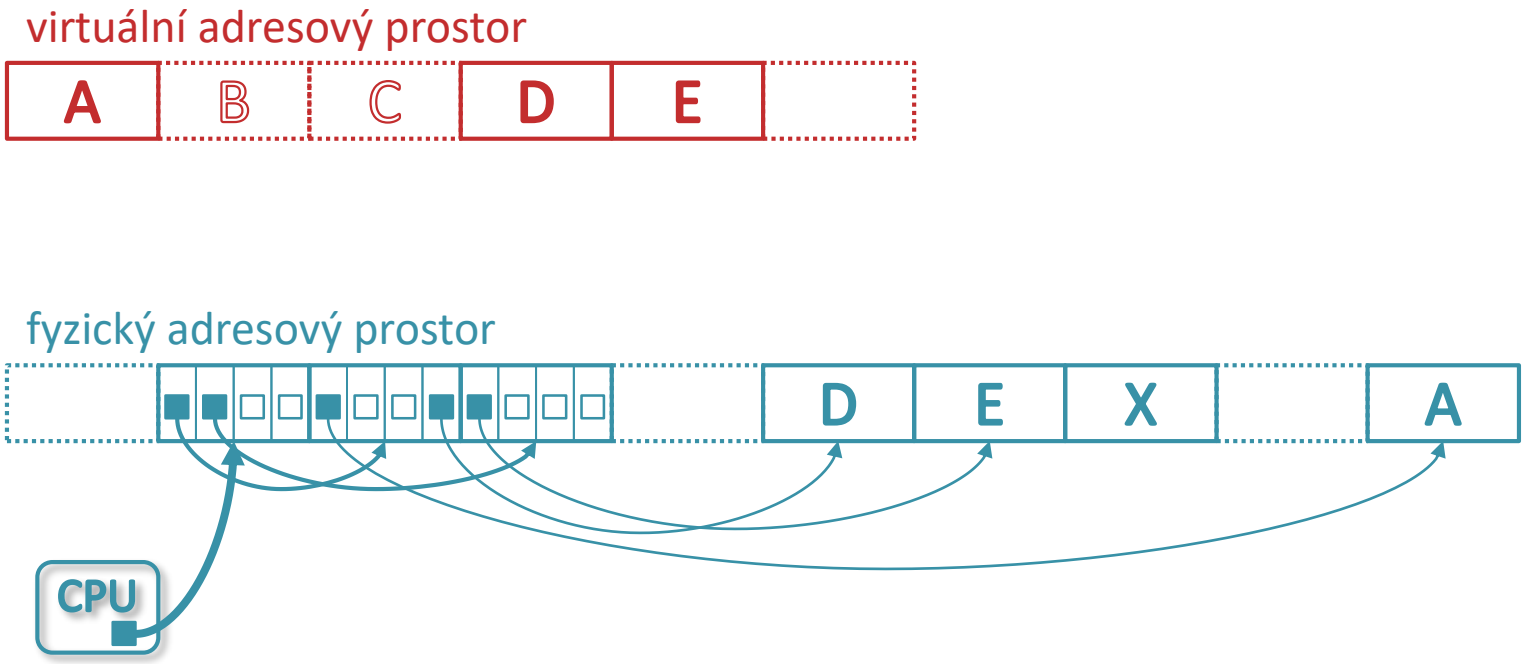
- Adresový prostor z pohledu procesu
  - Jeden nebo více souvislých úseků
    - Rozložení a význam určen dohodou aplikace a OS
  - Dělení na stránky je neviditelné

Virtuální paměť ve fyzickém počítači



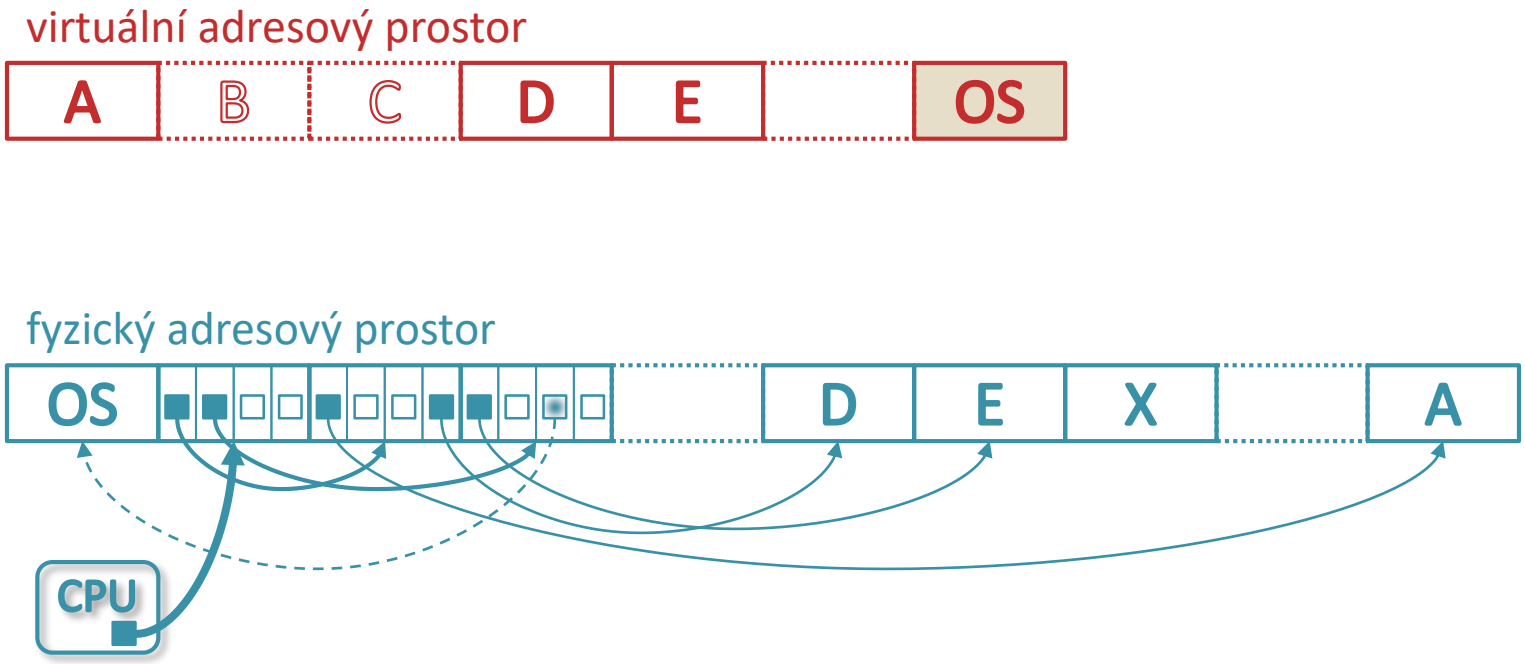
- Abstraktní pohled na virtuální paměť
  - Virtuální stránky jsou mapovány na fyzické rámce
  - Stránky odložené na disk mapovány nejsou
  - Fyzický adresový prostor je sdílen mnoha procesy

Virtuální paměť ve fyzickém počítači



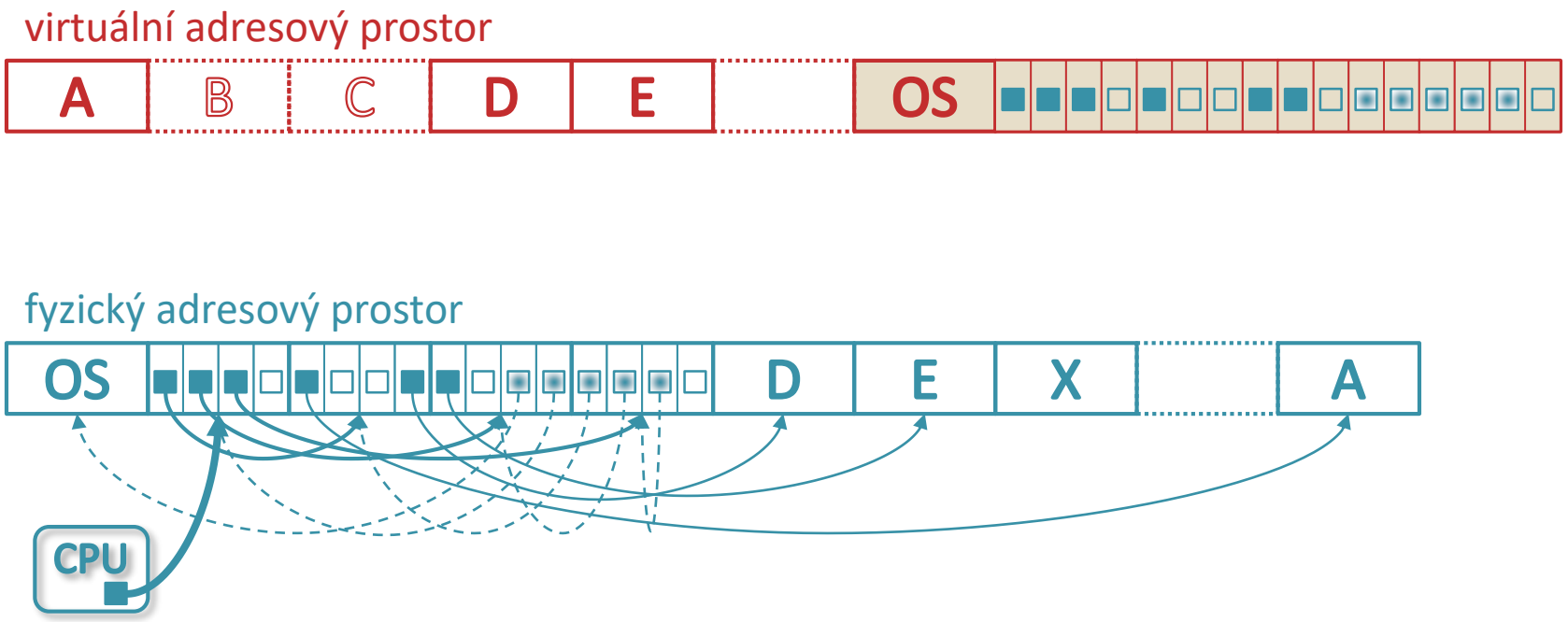
- Realizace dvouúrovňovým stránkováním (x86)
  - Při výpadku TLB procesor prochází dvě úrovně stránkovacích tabulek
  - Stránkovací tabulky uloženy ve fyzickém adresovém prostoru
  - Fyzická adresa kořene uložena v registru CPU

Virtuální paměť ve fyzickém počítači



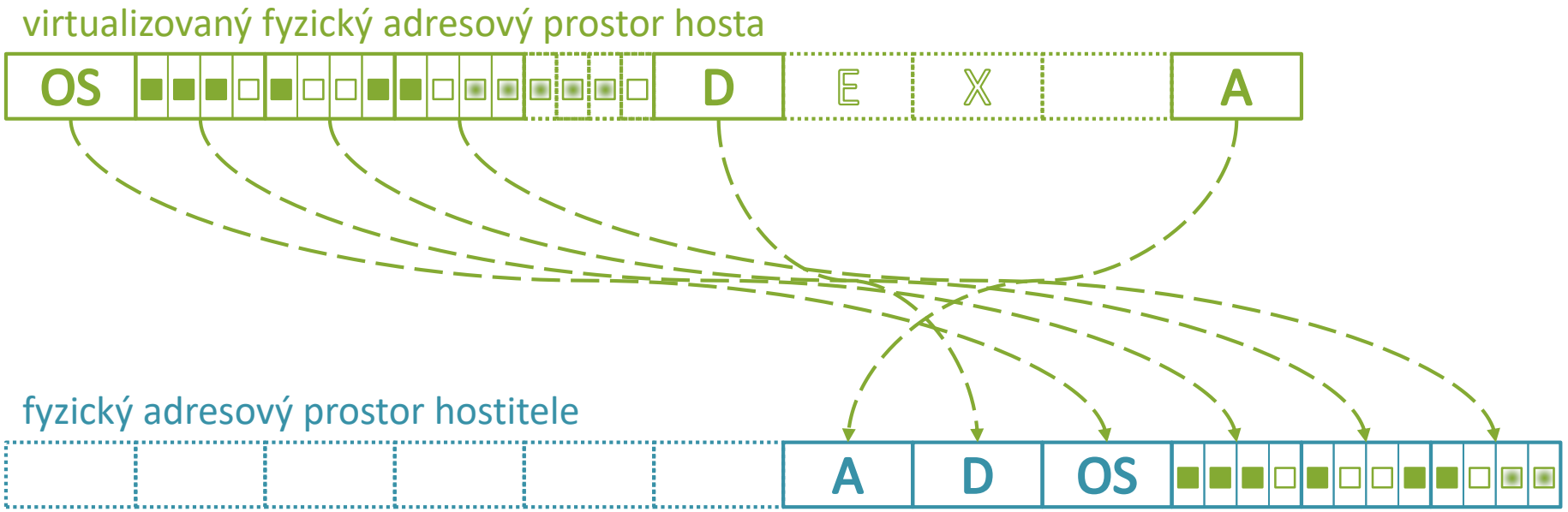
- Kód a data OS bývají součástí virtuálního adresového prostoru
  - Přepínání stránkování při každém volání OS by bylo neefektivní
  - Stránky OS přístupné pouze v privilegovaném režimu procesoru
  - Volání OS provedeno speciální instrukcí, která zapíná privilegovaný režim

Virtuální paměť ve fyzickém počítači



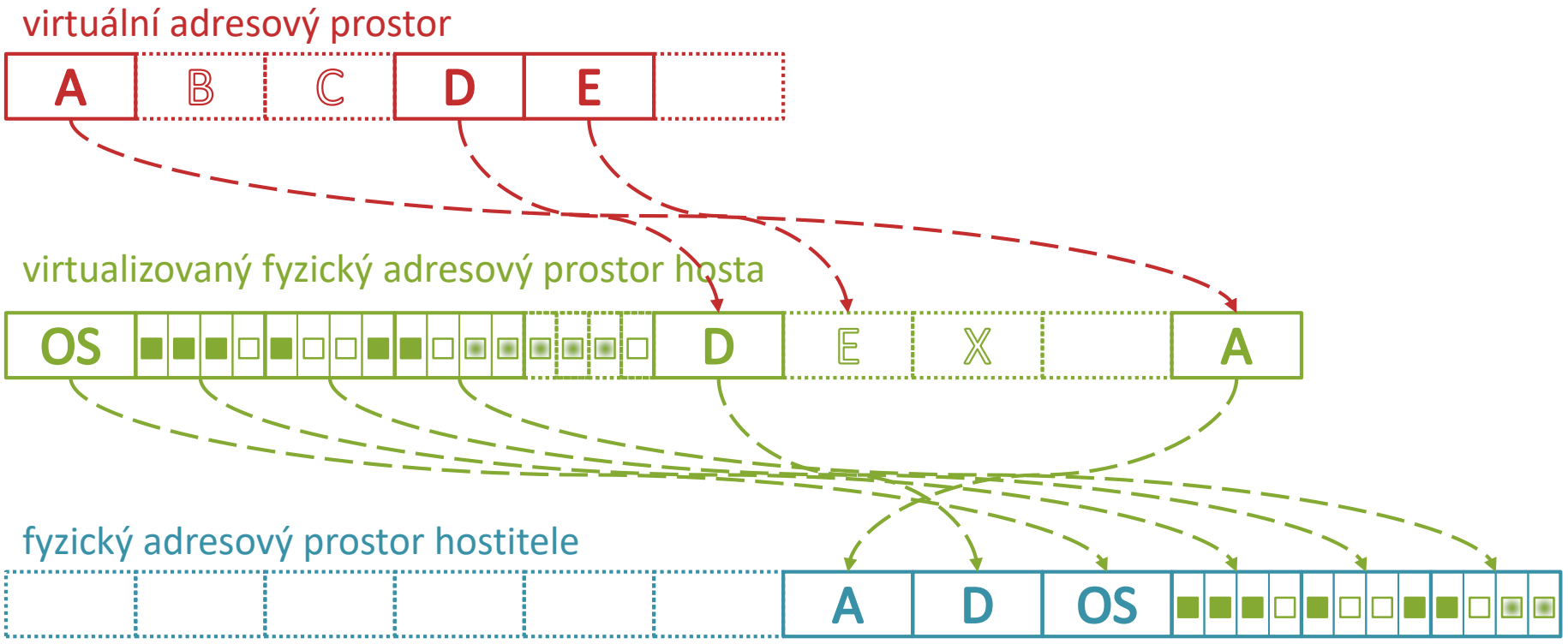
- Operační systém plní stránkovací tabulky
  - Stránkovací tabulky jsou mapovány podobně jako data OS
  - OS zapisuje do stránkovacích tabulek běžnými instrukcemi
  - Zápis většinou musí být následován privilegovanou instrukcí "TLB flush"

Virtuální paměť ve virtuálním počítači



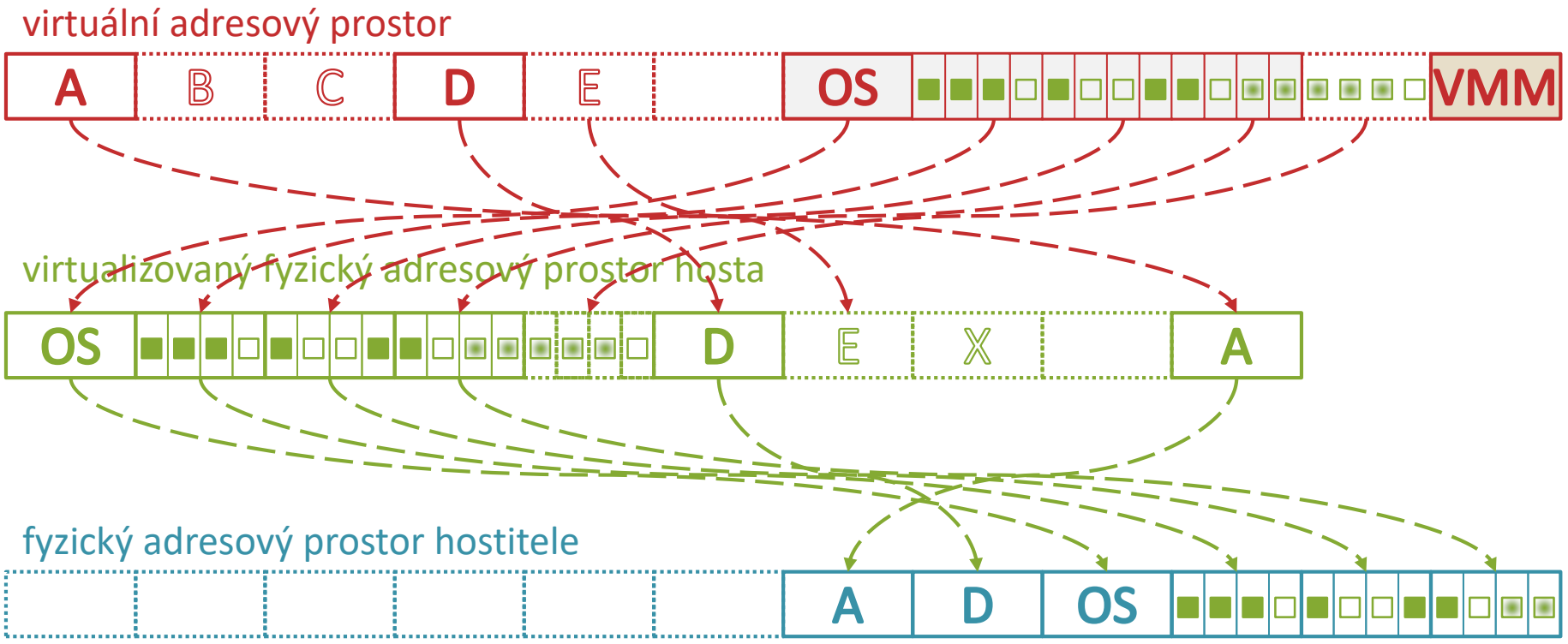
- VMM poskytuje iluzi fyzického adresového prostoru
  - Mapování fyzického prostoru hosta na fyzický prostor hostitele
  - VMM může odkládat stránky na disk podobně jako OS

Virtuální paměť ve virtuálním počítači



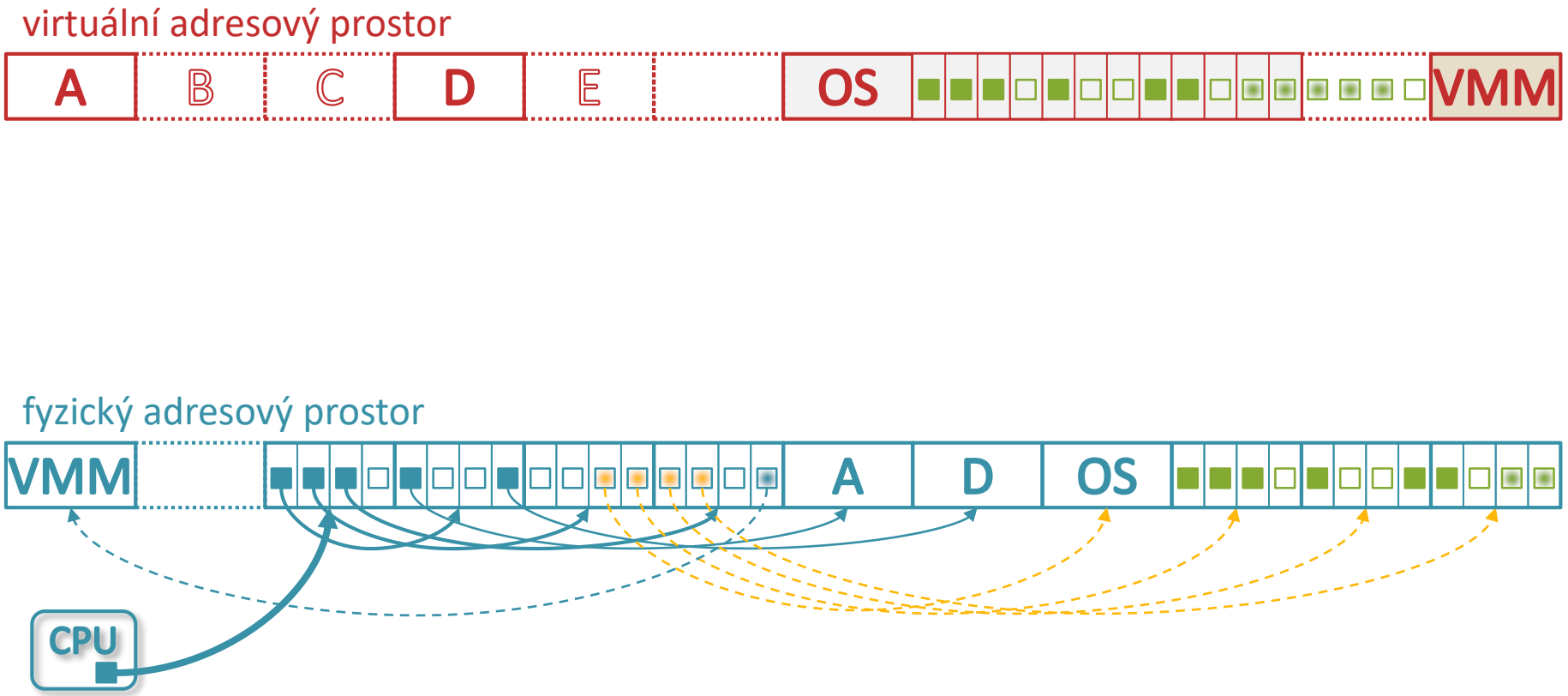
- Prováděný kód pracuje s virtuálním adresovým prostorem hosta
- Potřebné mapování vznikne složením
  - mapování definovaného operačním systémem hosta
  - mapování definovaného virtualizací počítače hosta

Virtuální paměť ve virtuálním počítači



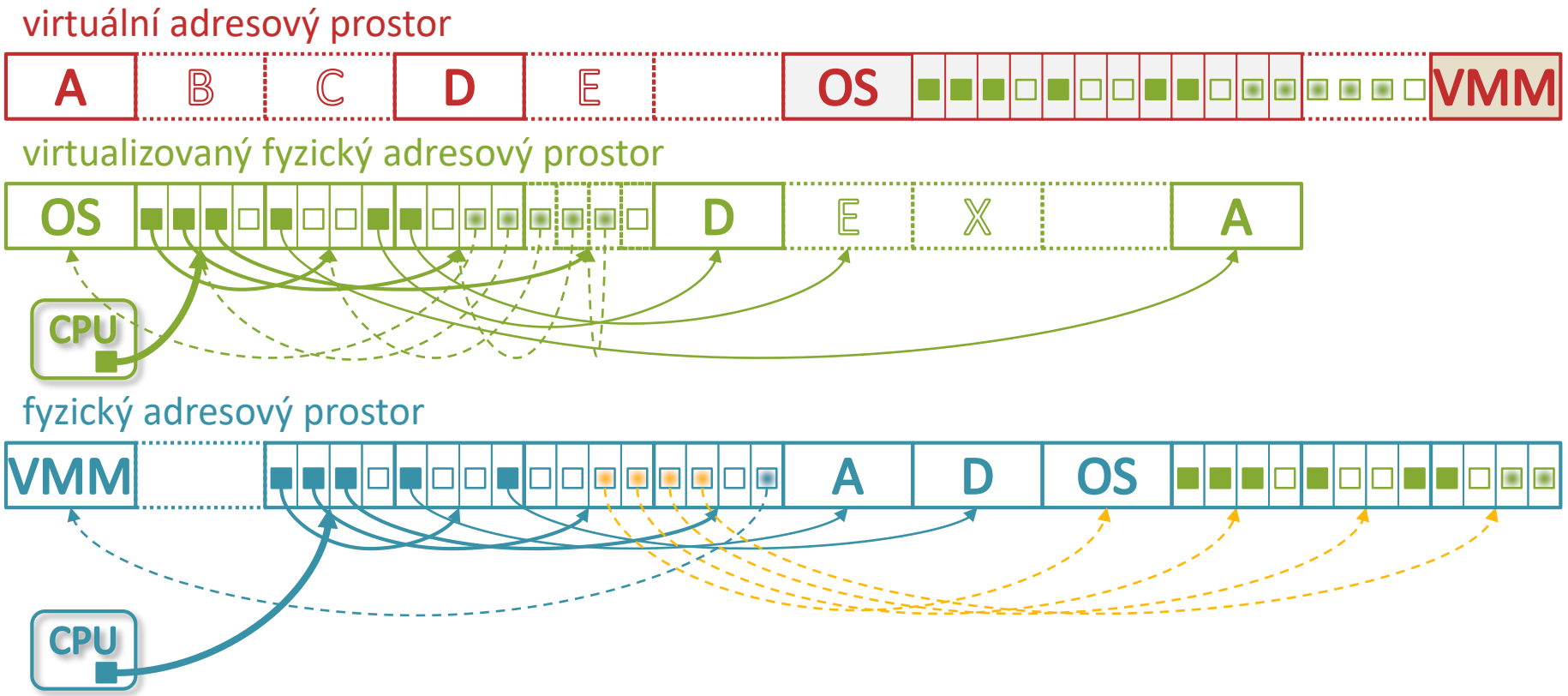
- Ve virtuálním adresovém prostoru běží
  - Aplikační procesy hosta
  - OS hosta
  - VMM
- Potřebujeme 3 úrovně privilegií ke stránkám

Virtuální paměť ve virtuálním počítači - bez HW podpory



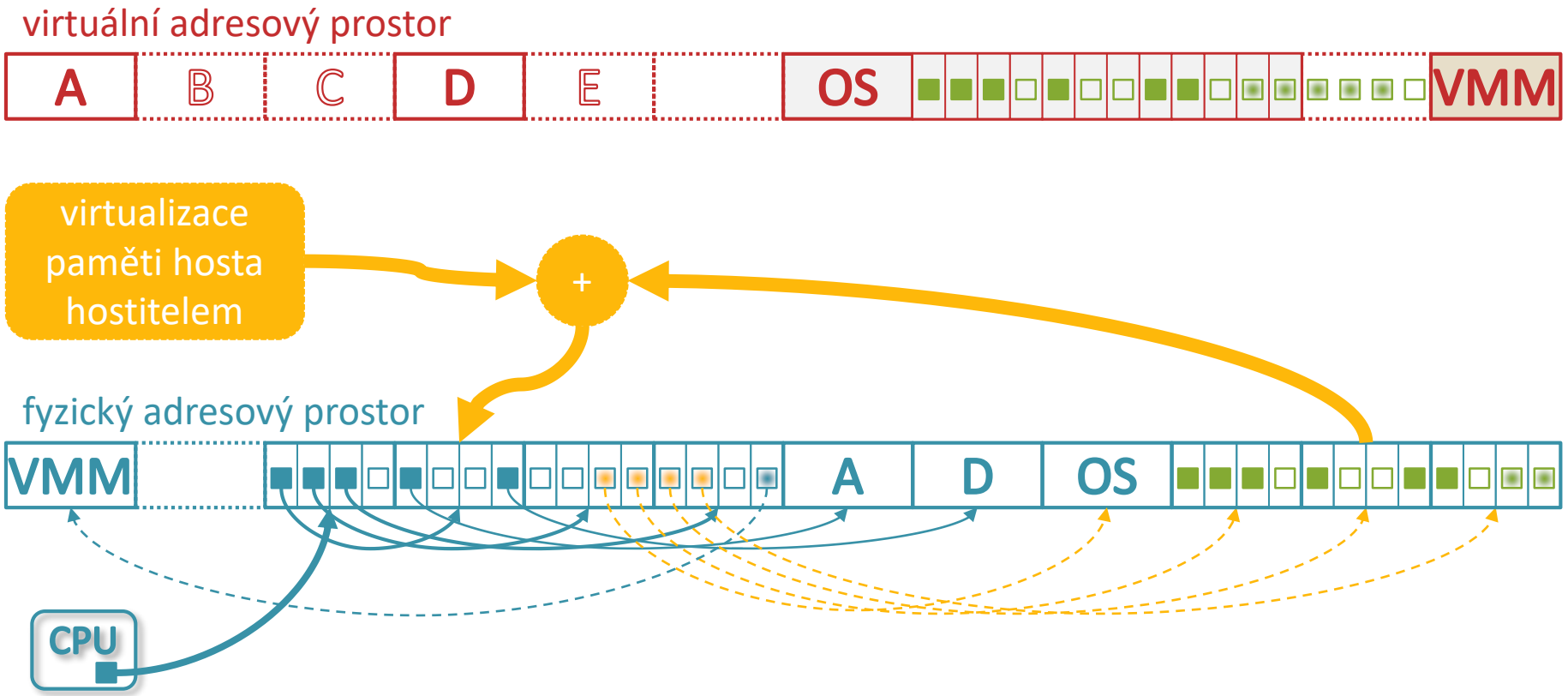
- Realizace složeného mapování
  - Stránkovacími tabulkami ve fyzickém prostoru hostitele
    - Obsahují fyzické adresy v prostoru hostitele

Virtuální paměť ve virtuálním počítači - bez HW podpory



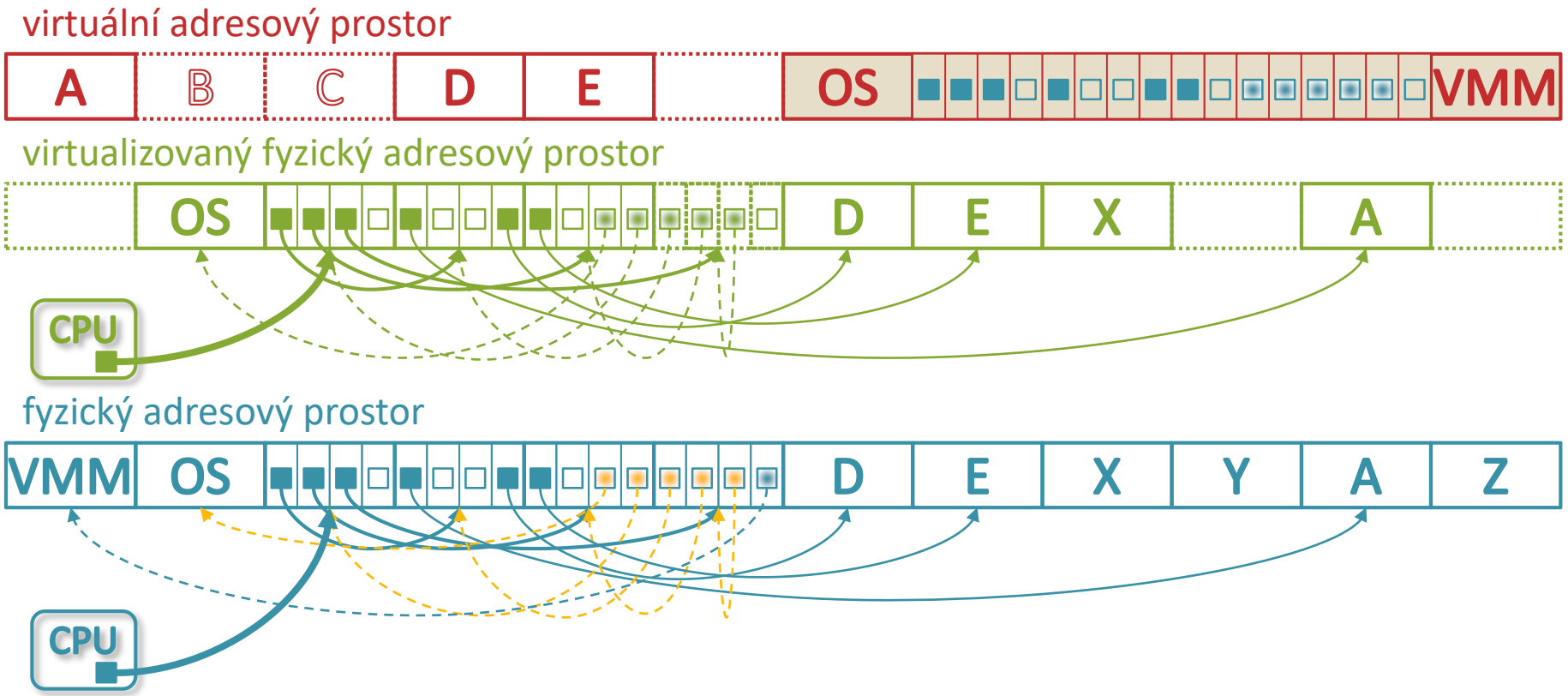
- V systému jsou dvojí stránkovací tabulky
  - Stránkovací tabulky hostitele používané fyzickým CPU
    - Obsahují fyzické adresy v prostoru hostitele
  - Virtualizované stránkovací tabulky hosta používané OS hosta
    - Obsahují virtualizované fyzické adresy v prostoru hosta

Virtuální paměť ve virtuálním počítači - bez HW podpory



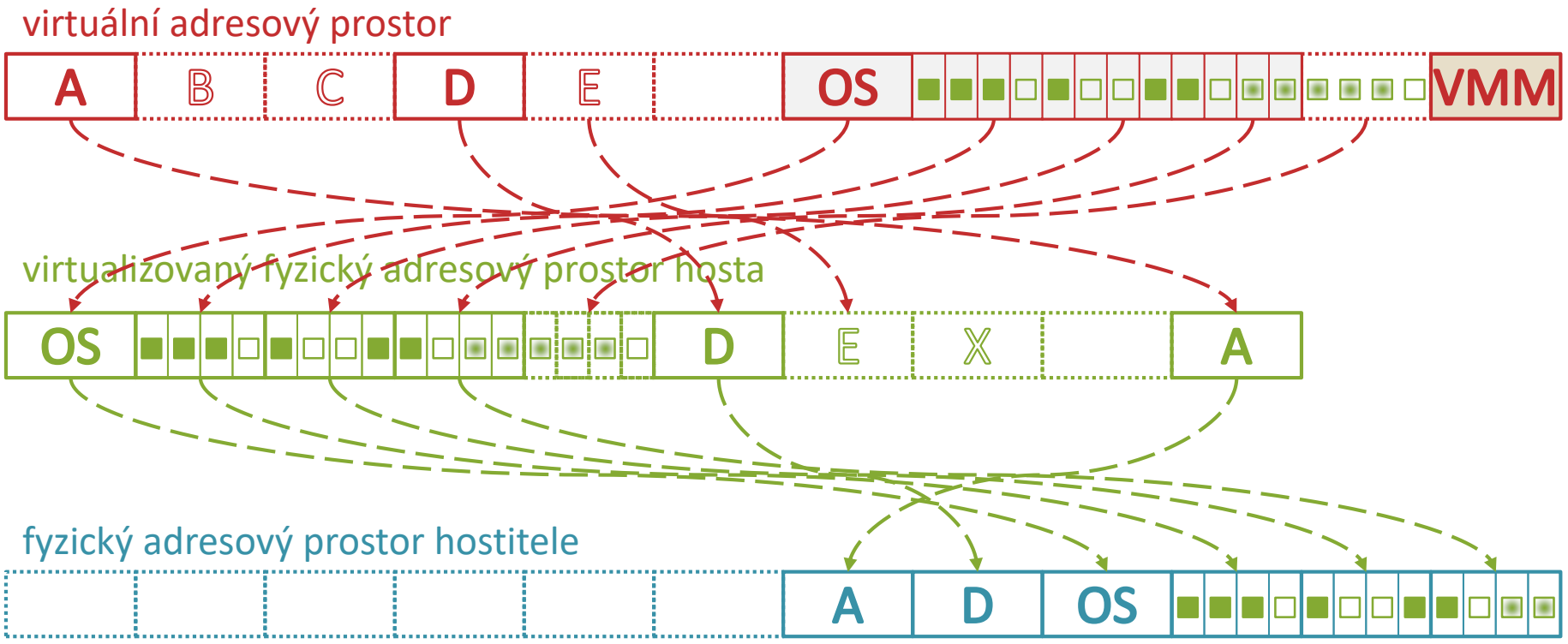
- VMM počítá výsledné mapování ze stránkovacích tabulek hosta
  - OS hosta zapisuje do svých tabulek neprivilegovanými instrukcemi
  - VMM musí zajistit přiměřenou koherenci fyzických tabulek a tabulek hosta
    - Virtualizované tabulky hosta mohou být mapovány read-only a zápisy emulovány
    - Koherenci lze udržovat v rámci emulace privilegované instrukce "TLB flush"

Virtuální paměť ve virtuálním počítači - bez druhé virtualizace



- Slabší VMM neumí odkládání virtualizované paměti na disk
  - Mapování virtualizovaných fyzických adres na fyzické je identita
    - VMM pouze kontroluje, zda OS hosta nemapuje nežádoucí fyzické adresy
  - OS hosta se musí vyrovnat s dírami ve fyzickém adresovém prostoru
    - Používáno převážně při paravirtualizaci (Xen)

Virtuální paměť ve virtuálním počítači - EPT / NPT[RVI]



- Extended (Intel) / Nested (AMD) Page Tables
  - Skládání dvojího mapování provádí procesor sám
    - Týká se mechanismu hardware page-walk
      - TLB obsahuje složené mapování
    - Řeší i chybné naplnění stránkovacích tabulek