Sem vložte zadání Vaší práce.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Podpora automatické správy virtualizačního kontejneru Solaris Zones na platformě Solaris

Bc. Tomáš Šimáček

Katedra počítačových systémů
Vedoucí práce: Ing. Michal Šoch, Ph.D.

27. března 2018

Poděkování

Doplňte, máte-li komu a za co děkovat. V opačném případě úplně odstraňte tento příkaz.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 27. března 2018 .....

Odkaz na tuto práci

Šimáček, Tomáš. *Podpora automatické správy virtualizačního kontejneru Solaris Zones na platformě Solaris*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Abstract v češtině

Klíčová slova Solaris, Solaris Zones, virtualizace, automatická správa

Abstract

Abstract in english

Keywords Solaris, Solaris Zones, virtualization, automatic management

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek kódů

Úvod

Virtualizace je technika, se kterou se dnes v IT můžeme setkat v mnoha podobách. Jednou z hlavních oblastí využití virtualizace je virtualizace serverů a mimo jiné se objevuje i v oblasti komunikačních sítí a desktopů. Tato technologie umožňuje vytvářet virtuální prostředí nebo prostředky na fyzickým hardware. Speciální softwarová vrstva zvaná virtualizační monitor (VMM) zajišťuje efektivní rozdělování prostředků fyzického systému mezi virtualizované subjekty.

Hlavním tématem této práce je virtualizace serverů, která umožňuje rozdělit jeden fyzický systém na několik nezávislých virtuálních prostředí zvaných virtuální počítač (VM). Možnost vytváření VM značně snižuje náklady na pořizování a provoz fyzických strojů, jelikož už není třeba dedikovaný server pro každou instanci OS. A konečně správným rozdělením VMs na fyzické servery můžeme docílit ideálního rozdělení zátěže a tím efektivně využít dostupné fyzické prostředky.

Rostoucí počet virtualizovaných serverů může mít za následek obtížnější správu. Automatizované nasazování, instalace nebo zálohování VMs může být značným ulehčením vývoje software, testování nebo nasazování aplikací do produkčního prostředí. Tomuto tématu se tato práce věnuje v souvislosti s virtualizačním kontejnerem Solaris Zones.

Cíle práce

Prvním cílem této práce je seznámení se s operačním systémem Solaris a jeho funkcemi. Především jde o popis virtualizační techniky Solaris Zones. Důraz je kladen na popis základních principů, které umožňují běh více zón v rámci jednoho sdíleného jádra OS.

Dalším cílem je detailní popis možností konfigurace zón, jejich instalace, zálohování a v neposlední řadě také integrace Solaris Zones s ostatními službami operačního systému Solaris.

Třetí cíl této práce je porovnat Solaris Zones s ostatními virtualizačními technologemi.

Posledním cílem této práce je implementace nástroje, který umí spravovat větší množství Solaris Zones. Tento nástroj bude umožňovat (dávkovou a interaktivní) instalaci zón na lokální i vzdálené servery, náhradu existujících zón a jejich zálohování. Dále bude umožňovat automatické přidání předem definovaných softwarových balíčků po instalaci zóny.

Struktura práce

TODO

Kapitola 1

Virtualizace

Jak je z názvu kapitoly patrné, hlavním obsahem následující části práce bude virtualizace a to především odvětví, které se věnuje výpočetní technice. Virtualizace je velice komplexní téma, a proto je nutné řádně specifikovat z jakého úhlu pohledu se na toto téma koukáme.

Po představení obecného konceptu virtualizace se proto podíváme na několik oblastí využité této technologie v informačních technologiích. Detailní popis všech oblastí virtualizace není předmětem této práce, a proto se ve zbytku diplomové práce budeme věnovat pouze tématu virtualizace serverů. I takto specifikované téma ale obsahuje mnoho virtualizačních principů a technik, které si u jednotlivých typů virtuálních strojů představíme. Jelikož virtualizace zažívá v dnešní době velký rozvoj, podíváme se také na jednotlivé scénáře nasazení serverů využívající virtualizaci. V poslední části této kapitoly jsou blíže představeny vybrané principy virtualizace a základy virtualizace CPU, paměti a I/O zařízení. TODO (uvidíme jestli bude třeba popisovat vše)

1.1 Obecná definice virtualizace

Než se pustíme do popisu jednotlivých typů virtualizace detailněji, je vhodné definovat tento pojem v obecném slova smyslu. Slovo virtuální je dle [1] definováno následovně.

Definice 1 (Virtual) *Almost or nearly as described, but not completely or according to strict definition.*

Ve výpočetní technice má tento výraz podle stejného zdroje [1] podobnou definici.

Definice 2 (Virtual in computing) *Not physically existing as such but made by software to appear to do so.*

Proces virtualizace ve výpočetní technice tedy můžeme definovat jako vytváření virtuálních prostředků, které skrývají nebo upravují podstatu fyzických prostředků před uživatelem. Tento proces zahrnuje vytváření více virtuálních prostředků z jednoho fyzického. Jako příklad této virtualizace můžeme použít virtuální paměť, kdy se virtuální paměť více procesů mapuje do hlavní (fyzické) paměti počítače. Na druhou stranu může jít i o vytvoření jednoho virtuálního prostředku z více fyzických prostředků. Příkladem pro tento typ virtualizace může být vytvoření jednoho logického disku z několika fyzických a to například v konfiguraci RAID.

Virtualizace si zcela jistě objevuje i v jiných oblastech, ale nás bude zajímat, jak se tento koncept využívá k virtualizaci výpočetní techniky a konkrétně jeho využití v sítích, operačních systémech a také v počítačovém HW.

1.2 Virtualizace ve výpočetní technice

Virtualizace se v dnešní době stala důležitou součástí návrhu počítačových systémů a zdárně se využívá v mnoha oblastech informačních technologií. Velkého rozvoje dosáhla především v oblastech virtualizace operačních systémů, procesorů, sítí ale také programovacích jazyků.

Pokud se podíváme na architekturu dnešních počítačových systémů z pohledu struktury a typů zařízení, které se v ní vyskytují, můžeme najít hned několik oblastí ve kterých se virtualizace využívá.

1.2.1 Virtualizace serverů

V dnešní době je pro většinu společností téměř nutností nějakým způsobem využívat výpočetních prostředků. Důvod pro jejich použití může být potřeba ukládání a zálohy obchodních záznamů, poskytování interních nebo externích služeb či běh výpočetně náročné aplikace. Ať tak či onak, hlavním poskytovatelem výpočetního výkonu v dnešních počítačových systémech je výpočetní server.

Klasický výpočetní server je fyzický počítač (HW), který poskytuje své výpočetní prostředky řídícímu programu. Řídícím programem se klasicky myslí operační systém. Druhy serverů můžeme rozdělit dle typu běžících uživatelských programů v operačním systému. Konkrétně pak hovoříme o aplikačních, souborových nebo výpočetních serverech, které se specializují na poskytování různých druhů služeb, jak je patrné z jejich názvu.

Proces virtualizace serverů spočívá v přenesení operačního systému a jeho služeb do virtuálního prostředí, které napodobuje chování HW ale není závislé na nižších vrstvách. Dochází tedy ke zvýšení přenositelnosti a možnosti současného běhu více instancí OS na jednom fyzickém stroji. Vytváření toho virtuálního prostředí je zajištěno speciální softwarovou vrstvou zvanou virtualizační monitor, hraje roli řídícího programu a pracuje mezi HW a jednotlivými

instancemi OS. Virtualizační monitor nebo také VMM je detailněji popsán v kapitole 1.6, kde jsou představeny jeho hlavní funkce a jednotlivé typy.

Počítačové systémy využívající virtualizace se skládají z dalšího typu serverů tzv. virtualizačních serverů, které slouží jako zdroj fyzických prostředků pro instance OS a jejich uživatelské programy. Tyto servery se vyznačují především velkým množstvím operační paměti a vysokým výpočetním výkonem, který je díky VMM rozdělován mezi hostované operační systémy. Výhody nasazení virtuální infrastruktury jsou dále popsány v kapitole 1.5.

Tato práce se zaměřuje právě na techniky virtualizace, které jsou v dnešní době aktuální a využívají se k virtualizaci serverů. Práce podrobně představuje virtualizační techniku Solaris Zones of firmy Oracle, která slouží pro vyváření virtuálních strojů (zón), které sdílejí jedno jádro OS.

1.2.2 Využití virtualizace v sítích

Oblast komunikačních sítí je další neméně významnou oblastí pro využití virtualizačních technik. Bez síťové infrastruktury by mezi sebou počítače nemohli komunikovat, a tudíž by jejich využití nemělo takový potenciál. V dnešní době je tato infrastruktura značně rozsáhlá a to v některých případech ztěžuje její správu. S virtualizací přichází do sítí možnost dynamické konfigurace sítě a to i její topologie. To vše lze uskutečnit z jednoho místa a bez nutnosti zasahovat do fyzických zařízení sítě.

Virtualizace sítí je koncept, který se v mnoha ohledech podobná virtualizaci serverů. V případě serverů, se VMM stará o reprodukci vlastností fyzických prostředků v SW. Podobně je to tomu i v případě virtualizace sítí, kde existuje funkční ekvivalent VMM, který reprodukuje síťové komponenty v SW. Administrátor má tak možnost za chodu vytvářet virtuální síťové komponenty jako je switch, router, firewall nebo load balancer a to vše v rámci desítek sekund. Tento síťový VMM také umožňuje spravovat nové virtuální sítě, které zahrnují všechny standardní síťové služby a kvalitu služeb.[2]

1.2.3 Virtualizace desktopu

Společně s virtualizací serverů a sítí je virtualizace desktopu posledním typem virtualizace, která stojí za zmínku. Pod pojmem desktop si představme klasický stolní počítač, který má obrazovku, myš a klávesnici.

S desktopem je klasicky spojeno grafické uživatelské prostředí, pomocí kterého uživatel ovládá počítač, instaluje aplikace nebo přizpůsobuje prostředí. Bez využití virtualizace nebo další podpůrných systémů jsou všechny informace o uživatelském nastavení uloženy na desktopu a uživatel se k nim dostane pouze z toho samého stroje. Virtualizací desktopu je rozuměno oddělení uživatelského prostředí a nastavení od fyzického stroje. Jednou z možností je přesunutí tohoto prostředí do virtuálního stroje, který je centrálně spravován a spouštěn, když uživatel potřebuje. Tento koncept umožňuje uživateli pří-

stup ke svému prostředí téměř bez ohledu na lokalitu nebo platformu. Mezi další benefity zavedení virtualizovaného desktopu patří zvýšení bezpečnosti a zjednodušení správy celého systému. Tyto výhody pramení především z centralizaci tohoto řešení.

1.3 Virtuální stroj

V části 1.1 jsme si definovali virtualizaci obecně jako virtualizaci fyzických (HW) prostředků. Virtualizace systému nebo komponenty jako je procesor, paměť nebo I/O zařízení na určité vrstvě architektury počítače znamená mapování jeho rozhraní na rozhraní nižší vrstvy způsobem, který může reprezentovat v jiném smyslu než fyzicky existuje.

Tento koncept virtualizace nemusí být aplikován pouze na jednotlivé subsystémy jako například disky, ale může být zobecněn na celý systém. Pro tento účel je zavedena speciální SW vrstva, která operuje mezi konkrétními vrstvami počítačového systému, aby bylo dosaženo požadované architektury. Tato vrstva poskytuje vyšším vrstvám rozhraní a všechny prostředky nižší vrstvy tak, že vyšší vrstvy nemají o existenci této vrstvy ponětí a přitom dochází k virtualizaci celého systému. Tímto způsobem může virtuální stroj obejít kompatibilitu některých komponent fyzického stroje nebo omezení HW prostředků.

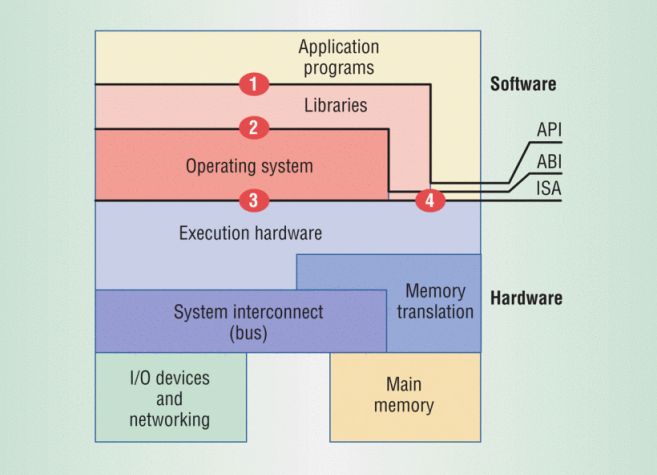
Než budeme pokračovat s klasifikací virtuálních strojů, představíme si architekturu klasického počítačového systému.

1.3.1 Architektura počítačového systému

Jelikož implementace virtuální strojů operují na rozhraních jednotlivých vrstev architektury počítačového systému, je nutné se řádně s těmito vrstvami seznámit. Tyto vrstvy reprezentují několik úrovní abstrakce v počítačovém systému, které mají za úkol odstínit složité implementační detaily některých rozhraní. Čím výše se v této hierarchii nacházíme, tím abstraktnější a jednodušší funkce máme k dispozici.

Každá z vrstev má dobře definované rozhraní, což umožňuje vývoj vyšších vrstev nezávisle na implementaci nižších vrstev, pokud tyto vrstvy budou dodržovat toto rozhraní. Jako příklad si můžeme vzít výrobce procesorů Intel a AMD vyrábějící mikroprocesory, které implementují instrukční sadu IA-32 (x86) [3]. Zatímco nezávisle na vývoji těchto procesorů můžou vývojáři softwaru vyvíjet aplikace, které se kompilují do této instrukční sady. Takto zkompilovaný program pak může být bez problému spuštěn na každém počítači s procesorem architektury IA-32.

Na druhou stranu komponenty navržené pro jeden typ rozhraní nebudou fungovat s rozhraním jiného typu. Jednoduše řečeno program sestavený pomocí instrukcí x86 se nebude dát spustit na počítači s procesorem architektury



Obrázek 1.1: Architektura počítačového systému [3]

SPARC. Nicméně díky některým technikám virtualizace se tohoto dá dosáhnout.

Obrázek 1.2 ukazuje hierarchii počítačového systému a některé jeho SW i HW vrstvy. Dále jsou na obrázku vyznačeny následující rozhraní.

- Instruction set architecture - ISA
- Application binary interface - ABI
- Application programming interface - API

Tyto tři rozhraní jasně definují rozmezí mezi HW a SW a určují architekturu počítačového systému. Uživatelské programy jsou zcela odkázány na funkcionalitu, která je jim poskytnuta kombinací těchto rozhraní.

Instruction set architecture

Instrukční sada neboli ISA definuje rozhraní mezi HW a SW. Rozdělit jí můžeme na dvě části a to na systémovou a uživatelskou instrukční sadu. Rozhraní s číslem 4 na obrázku 1.2 reprezentuje uživatelskou instrukční sadu, která obsahuje instrukce dostupné pro všechny uživatelské programy i knihovny. Rozhraní s číslem 3 na stejném obrázku pak reprezentuje systémovou instrukční sadu a zahrnuje instrukce dostupné pouze operačnímu systému. Tyto instrukce je možné vykonávat pouze v privilegovaném režimu procesoru a jsou zodpovědné za správu HW prostředků.

Application binary interface

Fyzické prostředky a zařízení dostupné v fyzickém systému spravuje operační systém, který k nim poskytuje přístup ostatním programům skrze svoje rozhraní. Toto rozhraní (číslo 2) se nazývá systémové a společně s uživatelskou částí (číslo 4) tvoří tzv. *application binary interface* neboli ABI. Toto rozhraní tedy neposkytuje aplikačním programům přímý přístup k HW prostředkům, ale zprostředkovává je skrze systémová volání. Operační systém tak

Application programming interface

Důležitou vrstvou softwarového vybavení počítače jsou uživatelské knihovny. Tyto knihovny skrývají implementační detaily systémových volání a poskytují rozhraní (číslo 1) pro vyšší programovací jazyky jako je C nebo C++. Společně s uživatelskou částí instrukční sady (číslo 4) tvoří rozhraní nazývané *application programming interface* neboli API. Uživatelské programy pak mohou využívat tohoto rozhraní, což přináší výhody v přenositelnosti na systémy, které nabízejí stejné API .

1.4 Klasifikace virtuálních strojů

Abychom mohli rozlišit jednotlivé typy virtuálních strojů, musíme se podívat v jaké části počítačové architektury operují a tedy jakou vrstvu virtualizují. V části 1.3.1 jsme definovali tři dobře definované rozhraní počítačového systému a je logické, že virtualizační software bude virtualizovat nějakou z nich. Dle rozdělení v [3] můžeme virtuální stroje obecně rozdělit na následující dva druhy v závislosti na tom, které rozhraní počítačové architektury virtualizují.

- *Systémové virtuální stroje*
- *Virtuální stroje v procesech*

Jak může být z názvu patrné, *systémové virtuální stroje* poskytují kompletní systémové prostředí, které podporuje operační systém a jeho aplikace. Operační systém využívá ke svému běhu ISA systému. Systémový virtuální stroj tedy musí poskytovat operačnímu systému stejné rozhraní jako OS očekává. Nabízí tak operačnímu systému přístup k HW prostředkům fyzického stroje, které mohou být virtualizovány.

Na druhou stranu procesy využívají ke svému běhu mimo uživatelské části ISA také ABI nebo v případě vyšších programovacích jazyků API. Virtuální stroje, které se specializují na virtualizaci těchto dvou systémových rozhraní, budeme nazývat *virtuální stroje v procesech*. Hlavním účelem tohoto typu virtuálního stroje je podpora jednoho procesu. Její činnost začíná v okamžiku vytvoření procesu a končí v okamžiku jeho ukončení.

V následujících podkapitolách jsou popsané jednotlivé typy virtuálních strojů. Tato klasifikace byla převzata z [3] a upravena podle požadavků práce.

Terminologie

Pro účely dalších částí diplomové práce si definujeme některé pojmy, které se v architektuře počítačového systému s virtuálním strojem vyskytují.

Po přidání virtualizačního SW mezi nějaké vrstvy počítačového systému nám vzniknou tři části. Tuto skutečnost popisuje obrázek 1.2, který zároveň ukazuje na jaké místo zaujímá virtualizační software v případě systémových virtuální strojů (b) a virtuálních strojů v procesech (a).

Softwarová vrstvy, které se v hierarchii počítačového systému nachází nad virtualizačním SW (je jim poskytováno rozhraní), se souhrnně nazývají *guest*. V případě systémových virtuálních strojů se dá také mluvit o *guest OS*, což je operační systém běžící ve virtuální prostředí.

Na druhou stranu vrstvy poskytující rozhraní virtualizačnímu SW neboli se nacházejí níže v hierarchii, nazývame *host*.

Poslední vrstva, která zbývá popsat je samotný virtualizační SW. V případě systémových virtuálních strojů se typicky nazývá virtualizační monitor neboli VMM. Pro virtualizační SW v druhém typu virtuálních strojů se používá název *runtime*.

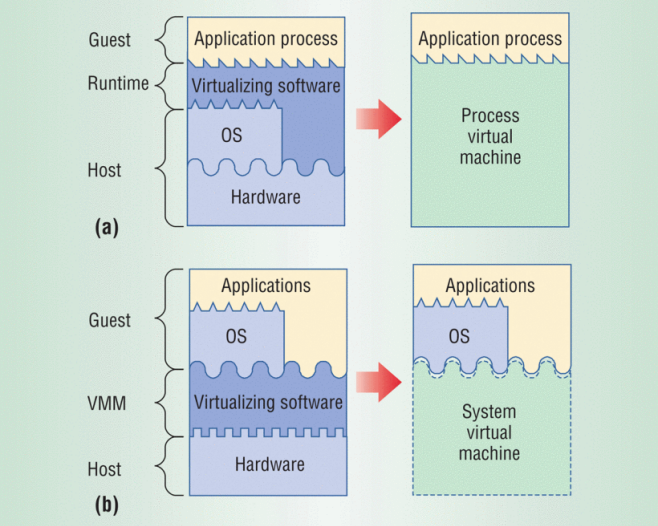
Jak je naznačeno na obrázku 1.2, typ virtuálního stroje je jasně určen strukturou hosta a virtualizačního SW. Systémové virtuální stroje se tedy skládají z HW vrstvy a virtualizačního monitoru. V případě virtuálních strojů v procesech se vrstva hosta skládá z HW a hostitelského operačního systému, ve kterém je spouštěn *runtime*.

1.4.1 Virtuální stroje v procesech

Jak již bylo zmíněno tento typ virtuálního stroje poskytuje uživatelským programům virtuální ABI nebo API. Různé implementace těchto virtuálních strojů si kladou za cíl splnění různých kritérií. Některé se snaží zajistit přenositelnost mezi různými počítačovými platformami a jiné se snaží optimalizovat instrukce uživatelského programu. Následovat bude stručný výčet jednotlivých typů virtuálních strojů, které spadají do této kategorie.

Virtualizace na úrovni OS

Pro představení prvního zástupce z této rodiny virtuálních strojů nemusíme chodit daleko. Jako virtuální stroj můžeme totiž považovat operační systémy, které umožňují současný běh více procesů najednou. Operační systém poskytuje každému procesu iluzi, že na systému běží sám. Z tohoto důvodu je nutné sdílet fyzickou paměť, CPU a jiná HW zařízení mezi běžícími procesy. Operační systém poskytne každému procesu stejně velký izolovaný virtuální adresní prostor, který je mapován do fyzické paměti počítače. Procesor se sdílí



Obrázek 1.2: Virtuální stroje v procesech (a) versus systémové virtuální stroje (b) [3]

mezi procesy tak, že dochází k tzv .přepínání kontextu, což znamená uložení všech registrů a načtení registrů procesu, který má přidělený procesor. Přístup k ostatním HW zařízením systému je řízen skrze systémové volání operačního systému. Konečně můžeme říct, že OS poskytuje virtuální prostředí pro každý proces v systému.

TODO partišnování zdrojů operačního systému.

Emulátory a překladače

Jak bylo zmíněno výše, některé implementace těchto virtuální strojů jsou zaměřeny na přenositelnost programů mezi jednotlivými počítačovými architekturami. Tyto virtuální stroje zpracovávají instrukce jiné instrukční sady než vykonává systém hosta a poté je dynamicky překládají do instrukční sady hosta. Konkrétní implementace by mohla například umožňovat vykonávat programy zkompilované pro architekturu IA-32 na systému s architekturou SPARC. Těmto virtuálním strojům říkáme překladače nebo emulátory, jelikož emulují prostředí dostupné na jiných architekturách a mapují ho do architektury hosta.

Techniky překladu instrukcí jsou detailněji popsaný v kapitole 1.7.1.4. Jen pro ukázku si zmíníme techniku interpretace, která jednotlivé instrukce nejprve načte, dekóduje a poté vykoná ekvivalentní instrukci nebo sadu instrukcí pomocí instrukční sady hosta.

Virtuální stroje v HLL

Virtuální stroje napsané ve vyšších programovacích jazycích přímo navazují na výše popsané téma přenositelnosti mezi platformami. Nevýhoda emulátorů spočívá ve faktu, že se specializují na překlad jedné instrukční sady do druhé. Pokud bychom tedy chtěli docílit přenositelnosti mezi všemi platformami, museli bychom vytvořit emulátory pro každou kombinaci instrukčních sad. Jelikož je toto řešení poněkud složité a náročné na implementaci, můžeme k vytvoření virtuálního stroje použít právě vyšší programovací jazyky. Virtuální stroj napsaný v nějakém HLL se tedy neopírá o konkrétní architekturu, ale o samotný programovací jazyk a jeho výhody. Takto vytvořený virtuální stroj přijímá vlastní jazyk a využívá konkrétního HLL k jeho provádění. Takto je zajištěna přenositelnost programů, které jsou napsané v jazyce virtuálního stroje, mezi systémy, nakterých jsou dostupné potřebné knihovny a samotná implementace virtuálního stroje.

Nejznámějším zástupcem této kategorie virtuálních strojů je Java virtual machine od společnosti Sun Microsystems. Tento virtuální stroj provádí instrukce vyššího programovacího jazyka zvaného Java a dnes existuje mnoho implementací v nejrůznějších programovacích jazycích. Implementace HotSpot, která je napsaná v jazyce C++ a kterou v dnešní době vyvíjí a udržuje společnost Oracle, je pravděpodobně neznámější a nejvíce využívanou implementací JVM.

1.4.2 Systémové virtuální stroje

Virtuální počítače

Virtualizace celého systému

1.4.3 Definice pojmů

Pro potřeby popisu virtualizačních technik si definujeme základní pojmy a entity, které se ve virtuální infrastruktuře vyskytují.

Fyzické prostředky počítače jsou dostupné prostředky a služby poskytované HW počítače. Tyto prostředky určují celkový výpočetní výkon daného stroje.

Virtualizační monitor, *Virtual Machine Monitor - VMM* nebo také *Hypervisor* je softwarová vrstva, která virtualizuje HW prostředky fyzického počítače a přiděluje je virtuálním strojům.

Virtuální prostředky jsou virtuální reprezentace fyzických prostředků. Tyto prostředky využívají virtuální stroje skrze VMM. Často tyto prostředky reprezentují pouze nějakou část jejich fyzických ekvivalentů. Například část adresního prostoru fyzické paměti nebo určité procento výpočetního času CPU.

Host je HW a SW platforma, která poskytuje virtuálním strojům své fyzické prostředky. Softwarové vybavení hosta obsahuje virtualizační monitor a

Obrázek 1.3: Konsolidace serverů

v některých případech může obsahovat i operační systém hosta tzv. *Host OS*

Virtuální stroj nebo *Virtual Machine - VM* je virtualizované prostředí vytvořené virtualizačním monitorem, ve kterém běží operační systém virtuálního počítače nazývaný *Guest OS* společně se svými aplikacemi.

1.5 Nasazení virtuální infrastruktury

Pro přechod k virtuální infrastruktuře serverů existuje v dnešní době několik dobrých důvodů. Jedním z hlavních benefitů virtualizace pro dnešní firmy a organizace je značná finanční úspora. Tato úspora se projevuje především ve snížení nákladů organizace na pořizování a provoz fyzických zařízení.

Mezi další benefity virtualizace patří především efektivní využití výpočetních zdrojů, vysoká dostupnost běžících aplikací nebo vytvoření oddělených a nezávislých prostředí pro vývoj, testování a nasazení software.

Výhody zavedení virtuální infrastruktury jsou podrobněji popsány v následujících podkapitolách, které se zabývají základními scénáři pro nasazení virtuální infrastruktury.

1.5.1 Konsolidace

Konsolidace serverů je proces sjednocování systémů z více fyzických serverů na jeden fyzický server, který pro tyto systémy poskytne virtuální prostředí pro jejich běh. Vstupem tohoto procesu je tedy několik systémů na fyzických serverech, na kterých běží různé aplikace. Vstup procesu je naznačen na obrázku 1.3 vlevo. Výstupem konsolidace je jeden fyzický server s dostatečnými prostředky, na kterém konsolidované systémy běží jako virtuální počítače. Výstup můžeme vidět na obrázku 1.3 vpravo.

Využití scénáře

Dnes je zcela běžnou praktikou provozovat jednu aplikaci na jednom dedikovaném serveru. Pokud aplikace využívá jen malé procento výpočetních zdrojů daného serveru, může administrátor sjednotit více takovýchto serverů do jednoho. Pro organizaci, která vlastní tisíce takovýchto serverů může konsolidace výrazně zmenšit požadavky na prostor, spotřebu energie a provoz fyzických serverů. Správnou konsolidací serverů může společnost docílit efektivního využití dostupných prostředků a tím výrazně snížit vynaložené finanční prostředky [**?**].

Rychlý vývoj technologií v oblasti hadware zapříčiňuje rychlé stárnutí některých systémů a přechod ze staršího na nový může být složitý. Obzvláště v případě, kdy systém potřebuje ke svému běhu speciální hadware. Aby bylo

Obrázek 1.4: Izolace aplikací

možné provozovat služby poskytované těmito zastaralými systémy, můžeme je spustit jako virtuální počítač na modernějším hadware. Systém se bude chovat stejně jako kdyby běžel na zastaralém hadware, zatímco výkonost služby může těžit z novější a výkonnější hadwarové vrstvy [**?**].

1.5.2 Izolace

Dalším ze scénářů využití virtualizované infrastruktury je izolace aplikací. Proces izolace aplikací spočívá v oddělení dvou a více kritických aplikací běžících na jednom systému do nezávislých virtuálních prostředí. Vstupem je jeden systém s aplikacemi, které se mohou negativně ovlivňovat. Vstup izolačního scénáře je naznačen na obrázku 1.4 vlevo. Výstupem je několik nezávislých virtuálních počítačů, ve kterých běží jednotlivé aplikace. Výstup izolace aplikací je ukázán na obrázku 1.4 vpravo.

Využití scénáře

V dnešní době jsou útoky na aplikace vystavené do internetu běžnou záležitostí. Pokud útočník využije nějaké zranitelnosti aplikace, může v některých případech získat kontrolu nad celým systémem. V takovém případě jsou ohroženy všechny data a aplikace, které na daném systému běží. Vhodným krokem v tomto případě je proto využití virtualizace a rozdělení aplikací do nezávislých prostředí.

Jedním z příkladů ohrožení systému může být útok na výpočetní zdroje. Podstatou útoku je vyčerpání fyzických zdrojů systému, což má za následek nedostupnost jeho služeb a v některých případech i pád celého systému. Ve virtualizovaném prostředí lze přidělit každému VM pouze určitou část prostředků a tím chránit celý systém před jejich vyčerpáním. V případě napadení jednoho VM sice dojde k jeho vyřazení, ale ostatní VM a jejich služby mohou dále pokračovat v běhu.

1.5.3 Migrace

Posledním diskutovaným scénářem nasazení virtualizované infrastruktury je migrace. Jedná se o proces přesunutí systému z jednoho počítače na druhý. V rámci virtualizace se budeme bavit o přesouvání systému na počítač s běžícím VMM, který zprostředkovává virtuální prostředí. Výstupem procesu je systém, který do něj zároveň vstupuje. Rozdíl je v tom, že daný systém na konci procesu běží ve virtuálním prostředí nějakého VMM. Dle typu migrovaného systému můžeme rozdělit scénář na následující typy.

Obrázek 1.5: Migrace virtuálního stroje

Obrázek 1.6: Migrace fyzického na virtuální stroj

Migrace VM

Migrací virtuálního stroje se rozumí přesun VM mezi dvěma různými fyzickými stroji s VMM. Tento přesun byl dříve možný pouze v případě když oba stroje měli stejný HW, operační systém a procesor [**?**]. Tato možnost administrátorovi umožňuje přesouvat virtualizované systémy na výkonnější hosty a tím umožňuje dynamicky regulovat využití fyzických prostředků v závislosti na aktuální zátěži systému.

Další výhodou zavedení virtualizované architektury je zajištění vysoké dostupnosti služeb. Virtualizace umožňuje zajistit redundanci ve smyslu spuštění služby na více serverech najednou. Ve virtualizované architektuře můžou nastat dva typy selhání. Prvním typem je selhání VM uvnitř VMM. Pokud dojde k selhání některé VM, jiná VM převezme obsluhu požadavků a v minimálním čase dojde k obnovení služby. Druhým typem je selhání celého VMM nebo hosta. V tomto případě je nutné provozovat více redundantních hostů pro VM, které v případě HW chyby převezmou obsluhu služby.

Proces migrace VM je představen na obrázku 1.5, kde můžeme vidět konfiguraci před migrací (vlevo) a po provedení migrace VM (vpravo).

Migrace fyzického stoje na VM

Migrace fyzického stroje na virtuální stroj je proces, kdy dochází k přesunu a virtualizaci systému z fyzického stroje. Vstupem je tedy systém běžící na stroji bez VMM, jak je naznačeno na obrázku 1.6 vlevo. Výstupem tohoto procesu je opět virtuální stroj běžící ve virtuálním prostředí VMM.

Virtualizací serverů dochází k uvolňování fyzického HW a stejně jako v případě konsolidace tak společnost může značně ušetřit na nákladech nutných k provozu a správě fyzických serverů. Obecně lze říct, že tento scénář přináší podobné benefity jako v případě konsolidace popsané v kapitole 1.5.1.

1.6 Virtualizační monitor

Jak již bylo zmíněno v definici z kapitoly 1.4.3, virtualizační monitor je softwarová starající se o virtualizaci fyzických prostředků hosta. Vytváří tedy virtuální prostředí pro běh virtuální počítačů. Jinak řečeno VMM vytváří iluzi pro každý virtuální počítač, který si myslí že přímo ovládá HW fyzického systému.

V klasickém nevirtualizovaném prostředí se OS stává po zavedení do hlavní paměti hlavním řídícím programem, který spravuje běh aplikací a vyřizuje požadavky aplikací na HW zařízení pomocí ovladačů. Operační systém tedy

pracuje v nejvíce privilegovaném režimu procesoru známého jako *ring 0* a může využívat všechny instrukce instrukční sady.

Druhým případem je prostředí s VMM nebo také virtualizované prostředí. V tomto prostředí přebírá roli hlavního řídícího programu VMM, který je po startu systému zaveden do hlavní paměti. Od té chvíle pracuje VMM v nejvíce privilegovaném režimu CPU a spravuje všechny hostované OS (guest OS). Ovladače k jednotlivým HW zařízení nyní nejsou součástí těchto OS, ale jsou součástí samotného VMM. Každý OS si sám spravuje vlastní aplikace a chová se jako kdyby nebyl izolován virtualizačním monitorem ve virtuálním prostředí.[**?**]

1.6.1 Požadavky na VMM

Z úlohy virtualizačního monitoru při virtualizaci systémů plyne několik základních požadavků, které by měl VMM splňovat. Specifikace těchto požadavků je převzata z [4].

Transparentnost

Operační systém běžící ve virtuálním prostředí virtualizačího monitoru by se neměl dozvědět o existenci VMM ani jiných VM, se kterými ve skutečnosti sdílí prostředky hosta.

VMM tedy zachytává systémové volání operačních systémů na HW zařízení nebo na čtení z paměti a transparentně je vyřizuje. Operační systém virtuálního počítače si tak myslí, že komunikuje přímo s HW. V této fázi VMM využívá některých technik virtualizace CPU, paměti nebo I/O zařízení, aby mohl sdílet prostředky mezi virtuální stroje. Některé základní techniky virtualizace jsou popsány v kapitole 1.7.

Izolace

Virtualizační monitor vytváří virtuální prostředí, které by mělo izolovat jednotlivé instance OS od sebe. Každý virtuální stroj má svůj kontext procesoru i svoji virtuální paměť mapovanou do jiných oblastí fyzické paměti. Jinak řečeno jednotlivé VM se nemohou vzájemně ovlivňovat svoji činnost a pád jedné VM by neměl ovlivnit činnost ostatních.

Ochrana

Virtualizační monitor by měl být chráněn proti všem vyšším vrstvám virtuální počítačů. Operační systém virtuálního stroje běží v privilegovaném režimu úrovně 1 (ring 1) a VMM běží v nejvíce privilegovaném režimu [1] úrovně 0. Pro operační systém virtuálního počítače to znamená, že nemůže přistupovat

[1] Současné procesory podporují plnou virtualizaci v HW. Mají speciální režim ochrany úrovně -1 speciálně pro VMM [4]

Obrázek 1.7: Nativní (Bare-Metal) VMM

do paměti mapované pro VMM a privilegované instrukce vyvolají volání do virtualizačního monitoru, který emuluje jejich chování.

1.6.2 Typy VMM

V následujících třech podkapitolách jsou představeny tři typické architektury VMM, které se liší ve způsobu přístupu k fyzickým prostředkům systému. Obecně lze tyto způsoby rozdělit na přímý a nepřímý přístup.

Nativní VMM

Virtualizační monitor se nazývá nativní nebo také Bare-Metal, pokud má přímý přístup k HW pomocí vlastních ovladačů. Obvykle se nativní VMM implementuje ve firmwaru počítače nebo jako hlavní program, který se po startu systému zavede do hlavní paměti a je mu předáno řízení. Tento přístup poskytuje nejvíce kontroly nad systémem a je také nejefektivnější, protože VMM přístup k HW zařizuje přímo. Architektura nativního VMM je ukázána na obrázku 1.7

Pokud hypervisor nepodporuje určité typy procesorů nebo periferních zařízení některých HW plaforem, může softwarové vybavení VMM limitovat přenositelnost na tyto platformy.[**?**]

Hostovaný VMM

Druhým typem architektury VMM se nazývá hostovaný, protože využívá ke svému běhu existující a běžící operační systém, který mu poskytuje rozhraní s HW. Virtualizační monitor v tomto případě neobsahuje ovladače k HW, protože jsou součástí operačního systému hosta. Aby mohl hostovaný VMM správně pracovat, běží některé jeho části (kernel) v privilegovaném režimu procesoru společně s operačním systémem hosta. Požadavky na HW zařízení jsou jádrem VMM přesměrovány do komponenty VMM, která neběží v privilegovaném režimu. Tato komponenta dále zavolá nativní rozhraní OS a požadavek je vyřízen operačním systémem hosta [**?**]. Jednotlivé komponenty hostovaného VMM jsou naznačeny na obrázku 1.8.

Tento typ virtualizace je v dnešní době velice populární, protože umožňuje používat virtualizaci na systémech s běžícím operačním systémem a to především desktopech. Přenositelnost hostovaného VMM je zcela určena přenositelností dané implementace mezi jednotlivými typy OS. Na druhou stranu tento typ VMM není vhodný pro nasazení do prostředí s vysokými výpočetními nároky, jelikož díky další vrstvě mezi HW a VMM není tak efektivní jako nativní VMM.

Obrázek 1.8: Hostovaný VMM

Obrázek 1.9: VMM se servisním OS

Servisní VM

Některé typy nativních VMM vyžadují pro svoji plnou funkčnost jednu nebo více speciálních instancí VM. Tato instance většinou slouží pro správu VMM a ostatních instancí VM. Nicméně existují i hypervisory, které využívají výhody existujících OS a především jejich podpory velké škály HW ovladačů. V tomto případě je OS spuštěn ve speciální instanci VM a je společně s jeho ovladači využíván jako komponenta VMM [**?**]. Tento typ architektury je naznačen na obrázku 1.9.

1.7 Techniky virtualizace

O pár odstavců výše jsme si popsali podstatu virtualizačního monitoru a také jsme si představili jeho základní typy. Nyní se pokusíme popsat techniky a principy, které VMM používá pro práci s HW, aby docílil vytvoření virtuálního prostředí.

Na chvilku se oprostíme od VMM a podíváme se na HW počítače z pohledu OS. Přesněji chceme vědět, co potřebuje OS ke svému běhu, abychom byli schopni vytvořit virtuální prostředí pro jeho běh. Důležité je zmínit, že chceme na jednom fyzickém systému provozovat více VM, tedy více instancí OS.

Operační systém je v konečném součtu jenom soubor instrukcí, kterým procesor rozumí a umí je vykonávat. Abychom mohli na fyzickém systému provozovat operační systém nebo jakýkoli jiný program, je nutné aby využíval jen instrukce z dostupné instrukční sady (ISA) procesoru. Pokud je tento požadavek splněn, může být OS na tomto systému provozován bez dalších opatření. Na druhou stranu pokud požadavek splněn není a instrukce programu jsou napsané v jiné instrukční sadě, je nutné tuto sadu emulovat. Jinými slovy musíme zajistit překlad z jedné instrukční sady do druhé. Tomuto tématu se více věnuje kapitola **??**. Vedle překladu instrukcí je nutné zajistit sdílení procesoru mezi jednotlivými virtuálními stroji.

Vedle vykonávání instrukcí potřebuje OS někde uchovávat svoje datové struktury a svůj kód. To samozřejmě vede k hlavní paměti počítače a nutnosti sdílení této paměti mezi virtuálními počítači.

Nemusíme chodit daleko a podobný princip sdílení prostředků mezi více entit můžeme najít v dnešních operačních systémech. V roli virtuálních počítačů si můžeme představit klasické procesy a roli VMM převezme OS, který má za úkol přidělovat prostředky procesům [5]. Stejně jako v případě virtuálních strojů spolu procesy musí sdílet procesorový čas a hlavní paměť, a proto

jsou techniky virtualizace počítačů podobné těm, které se používají v OS pro souběžný běh více procesů.

V neposlední řadě OS potřebuje ovládat zařízení a periferie připojené k fyzickému systému. K čemu by nám jinak byl počítač, kdybychom nevěděli co nám vlastně říká nebo ho dokonce nemohli ovládat.

Všechny výše zmíněné problémy musí VMM řešit, pokud chce vytvořit prostředí pro běh více OS na jednom fyzickém stroji. Ve zkratce můžeme říct, že virtualizační monitor musí implementovat multiplexování CPU, virtualizaci paměti, virtualizaci I/O zařízení a v některých případech i emulaci ISA.

Následujících podkapitoly představují základní techniky, které by měl VMM v nějaké podobě implementovat. Pro lepší pochopení této problematiky si definujeme několik pojmů.

Definice 3 *Kernel mód*

Definice 4 *User mod*

Definice 5 *PC*

1.7.1 Virtualizace CPU

Nyní se opět podíváme na fyzický stroj z pohledu virtualizačního monitoru a na jeho práci s procesorem. Spustit operační systém ve virtuální prostředí nutně musí znamenat, že se nějakým způsobem začnou vykonávat jeho instrukce. Jelikož OS virtuálního počítače neví o existenci VMM, předpokládá, že je v systému sám a má přímý přístup k HW. Na rozdíl od klasické nevirtualizované architektury není OS hlavním řídícím programem, a proto je nutné, aby VMM obsloužil volání privilegovaných instrukcí. Dále je třeba zajistit ochranu paměti alokované VMM před neprivilegovaným přístupem. Neprivilegovaný přístup znamená, že se OS virtuálního počítače nebo procesy v něm běžící pokusí přistoupit k paměti, která je alokovaná virtualizačním monitorem.

1.7.1.1 Režimy ochrany CPU

Moderní procesory mají 4 režimy ochrany paměti, které se značí čísly od 0 do 3. Nejvyšší režim ochrany má číslo 0 a je označován jako režim kernel. Procesor běžící v tomto režimu má přístup ke všem instrukcím instrukční sady a může přistupovat k jakékoli paměti. Číslo 3 naopak znamená režim nejnižší ochrany a může využívat jen část instrukční sady, které se někdy přezdívá uživatelská ISA. Tomuto režimu se říká user. Většina operačních systémů používá právě dva výše zmíněné režimy ochrany a zbylé dva jsou nevyužité.

Operační systémy v nevirtualizované architektuře hrají roli řídícího programu, a proto běží v režimu kernel. Ve virtualizované architektuře tuto roli přebírá VMM, a proto je potřeba, aby běžel v nejvíce privilegovaném režimu.

Řešením může být posunutí OS virtuálního stroje do méně privilegovaného režimu. Například na úroveň 1, zatímco VMM poběží na úrovni 0. Druhým řešením je vytvoření nového režimu ochrany s číslem -1, který bude speciálně pro VMM. Současné procesory řeší ochranu právě tímto způsobem a podporují plnou virtualizaci v HW [4].

Výsledkem je tedy znemožnění přístupu programů s nižší úrovní ochrany k paměti alokované programem z vyšší úrovně ochrany. Paměť VMM je ve virtualizované architektuře chráněna proti přístupu z OS virtuálního počítače a stejně tak je chráněna paměť OS virtuálního počítače před přístupem z jeho procesů.

1.7.1.2 Multiplexování CPU

Zjednodušeně můžeme říct, že spuštění virtuálního počítače v prostředí virtualizačního monitoru zařídíme skokem na adresu první instrukce OS.

Pro jednoduchost předpokládejme, že máme k dispozici jedno výpočetní jádro CPU a chceme ho sdílet dvěma virtuálními stroji. Oba virtuální stroje mají svojí instanci OS a běží na nich jejich aplikace. Princip střídání virtuálních strojů na CPU je velmi podobný s principem střídání běžících procesů v OS. Operační systém provádí tzv. změnu kontextu (context switch), kdy je přerušen běh aktuálního procesu a procesor je přidělen jinému. V případě VMM je nutné provést tzv. změnu stroje (machine switch). Při tomto procesu dojde nejprve k uložení celého stavu virtuálního stroje, což zahrnuje uložení registrů, PC a na rozdíl od změny kontextu je nutné uložit i aktuální stav HW. Dále je obnoven stav virtuálního stroje, kterému plánovač přidělil procesor a poté je proveden skok na adresu instrukce uvedené v PC [6]. Tímto krokem je proces změny virtuálního stroje na CPU dokončen.

V případě systému s více procesorovými jádry je situace velmi podobná. Jelikož je k dispozici více jader CPU, může v jednom okamžiku běžet více VM. Z tohoto důvodu je nutné synchronizovat přístup těchto virtuálních strojů k HW systému.

1.7.1.3 Virtualizace privilegovaných instrukcí

Instrukční sadu procesoru můžeme rozdělit na dvě části. V první části jsou neprivilegované instrukce, které mohou být prováděny v každém režimu ochrany CPU. Tyto instrukce jsou využívány především uživatelskými programy a ve virtualizovaném prostředí je můžeme přímo provádět na CPU bez nutnosti nějakých opatření. Pokud aplikace nebo OS virtuálního počítače potřebují vykonávat tento typ instrukcí, dojde k jejich provedení na CPU bez nutnosti zásahu VMM.

Druhou částí ISA, jsou tzv. privilegované instrukce, které mohou být prováděny pouze v privilegovaných režimech CPU. Některé privilegované instrukce mohou přímo skryté paměti cache, TLB, vyvolávat přerušení nebo jiným způ-

sobem ovlivňovat běh systému. Operační systém ve virtualizovaném prostředí nemůže mít možnost vykonávat privilegované instrukce, protože by pak ovládal fyzický systém a od toho je na systému přítomen VMM. Z tohoto důvodu musí virtualizační monitor nějakým způsobem zachytávat pokusy o vykonávání privilegovaných operací a zajistit jejich správné provedení.

Jedním z případů, kdy VMM musí reagovat na volání privilegované instrukce, je situace, kdy se proces operačního systému snaží provést systémové volání. Příkladem systémového volání může být volání funkce `open()` pro otevření souboru na disku, volání funkce `write()` pro zápis dat do souboru nebo třeba volání funkce `exec()` pro vytvoření dalšího procesu v systému.

Nejprve se podíváme na to, jak funguje systémové volání v klasických systémech bez virtualizace. Konkrétně se podíváme na architekturu x86, kde se pro dosažení systémového volání používá speciální instrukce přerušení `int` s argumentem `0x80`. V ukázce 1.1 je zobrazen kód assembleru ze systému FreeBSD [**?**], který spouští systémové volání funkce `open()` pro otevření souboru. Z kódu 1.1 je vidět, že volání funkce `open()` potřebuje tři parametry a to konkrétně mód otevření souboru, příznaky a cestu k souboru na disku. Typ systémového volání je určen díky poslednímu parametru na zásobníku, kterým je v tomto číslo 5 [6]. A konečně na posledním řádku ukázky se volá zmíněná instrukce `int`, která vyvolá přerušení.

Kód 1.1: Systémové volání na FreeBSD

```
open:
        push    dword mode
        push    dword flags
        push    dword path
        mov     eax, 5
        push    eax
        int     80h
```

Z pohledu virtualizačního monitoru je důležité, co se děje po vyvolání přerušení. Jelikož se jedná o privilegovanou instrukci, dojde k přepnutí procesoru z uživatelského režimu (user) do privilegovaného režimu (kernel) a kontrolu dostane operační systém. Konkrétně je řízení předáno rutině starající se o obsluhu přerušení, která byla zaregistrována v HW operačním systémem při jeho prvním startu [6]. Po obsloužení systémového požadavku je řízení vráceno zpět volajícímu procesu. Posloupnost volání je naznačena v tabulce 1.1 společně s přepínáním režimu ochrany procesoru.

Je logické, že případě virtualizace bude nutné přidat několik kroků do posloupnosti volání. Rozdílem oproti předchozímu případu je fakt, že OS nemá nad systémem plnou kontrolu a neběží v plně privilegovaném režimu. Tuto funkci nyní plní VMM, který má v HW zaregistrovanou rutinu pro obsluhu přerušení. Pokud tedy nějaký uživatelský proces chce provádět systémové volání, bude postupovat stejně jako v předchozím případě. Nicméně zpráva o přerušení se nedostane k OS ale k VMM, jak je naznačeno v druhém kroku

Proces	Hardware	Operační systém
1. *Systémové volání:* Vyvolání přerušení;		
	2. *Kernel mód:* Skok na obsluhu přerušení;	
		3. *Obsluha přerušení:* Dekódování přerušení a spuštění odpovídající rutiny OS; Návrat z OS;
	4. *User mód:* Návrat z obsluhy;	
5. *Obnova běhu;*		

Tabulka 1.1: Sytémové volání bez virtualizace [6]

v tabulce 1.2. Virtualizační monitor nemůže přímo vyřídit systémové volání, protože mu nerozumí a neví co má dělat (není OS). Tuto funkci plnil operační systém a jeho rutina pro obsluhu přerušení. Jak již bylo zmíněno, při prvním startu OS se systém pokusí registrovat v HW rutinu pro obsluhu přerušení, což je ale privilegovaná operace [6]. Tento pokus VMM odchytí a ke každému běžícímu OS si zaznamená adresu těchto rutin. Virtualizačnímu monitoru tedy stačí spustit rutinu korespondující OS v méně privilegovaném režimu a počkat na její dokončení. V okamžiku ukončení obsluhy systémového OS opět provede privilegovanou instrukci a řízení je předá zpět VMM, jak je ukázáno ve čtvrtém kroku tabluky 1.2. Konečně VMM přepne procesor do uživatelského režimu a vrátí řízení zpět uživatelskému programu, čímž je systémové volání ukončeno. V tabulce 1.2 jsou z důvodu úspory místa vynechány operace HW, které přepínají režimy procesoru.

Z obou tabulek je patrné, že při virtualizaci se musí vykonat více kroků, čímž se zvyšuje doba vykonávání privilegovaných instrukcí a to může mít za následek zhoršení výkonu systému [6].

1.7.1.4 Emulace instrukční sady

Doposud jsme se bavili o situaci, kdy stroje běžící ve virtualizovaném prostředí zpracovávaly instrukce stejné architektury jako host. Nyní se krátce podíváme na situaci, kdy virtuální stroje zpracovávají instrukce jiné architektury než fyzický stroj.

V tomto případě procesor nerozumí instrukcím VM a je nutné zajistit překlad z jedné ISA do druhé. Tato technika se nazývá emulace instrukční sady a spočívá v navození iluze, že virtuální stroj běží na HW podporující jeho instrukční sadu.

Proces	Operační systém	Virtualizační monitor
1. *Systémové volání:* Vyvolání přerušení;		
		2. *Obsluha přerušení:* Skok na obsluhu přerušení odpovídajícího OS (snížení privilegií)
	3. *Obsluha přerušení:* Dekódování přerušení a obsloužení systémového volání; Návrat z OS;	
		4. *Obsluha ukončena:* Provedení reálného návratu;
5. *Obnova běhu;*		

Tabulka 1.2: Sytémové volání s virtualizací [6]

- Přímá interpretace
- Binární překlad
- Inkrementální binární překlad za běhu

[7]

1.7.2 Virtualizace Paměti

1.7.3 Virtualizace I/O zařízení

Kapitola 2

Solaris

Solaris nebo dříve SunOS je operační systém původně vytvořený firmou Sun Microsystems, ale v současné době vyvíjený a podporovaný firmou Oracle. Je to komplexní unixový operační systém, který v sobě integruje pokročilé technologie pro virtualizaci, moderní souborový systém ZFS, vlastní systém pro instalaci a správu SW a v neposlední řadě také podporu cloudu. Díky integraci těchto technologií poskytuje Solaris stabilní a rychlé prostředí pro různé scénáře nasazení aplikací a navíc tato integrace vytváří pohodlné rozhraní pro správu tohoto OS.

2.1 Verze Solarisu

V době psaní této diplomové práce je nejaktuálnější stabilní verze operačního systému Solaris verze s označením 11.3, avšak ke dni 30. ledna 2018 byla do světa vypuštěna beta verze 11.4 [**?**]. Pro účely popisu operačního systému Solaris a jeho služeb, zejména služby Solaris Zones, bude použita stabilní verze 11.3. Existují i starší verze 11.2 a 11.1, které ale nebudou předmětem zkoumání.

2.2 Podporované architektury

Operační systém Solaris v současné době podporuje následující dvě HW architektury počítačových systémů.

- x86
- SPARC

Jelikož pořízení architektury SPARC by bylo složitě, bude pro účely této diplomové práce použita architektura x86 přesněji její 64 bitová verze.

2.2.1 x86

První počítačovou architekturou podporovanou operačním systémem Solaris je x86. Tato architektura je v dnešní době velmi rozšířená především v oblasti osobních počítačů a je podporována nejznámějšími OS jako Windows, Linux a Mac. Solaris tuto architekturu podporuje jak v 32 bitové verzi x86-32 tak i v 64 bitové verzi x86-64.

2.2.2 SPARC

Scalable Processor Architecture neboli SPARC je z pohledu Solarisu domovská architektura. Architektura SPARC byla stejně jako Soalris původně navržena společností Sun Microsystems a nyní ji spravuje společnost Oracle. Tato architektura je tedy od začátku své existence úzce spojena s operačním systémem Solaris, který se snaží využívat všechny její výhody. Uplatnění této architektury je především v komerčním sektoru, který klade vysoké nároky na přizpůsobivost a škálovatelnost řešení.

2.3 Služby

Hlavní předností operačního systému je kvalita ale i kvantita jeho služeb. Tyto služby umožňují nasazení toho OS i ve scénářích, kdy by ostatní OS selhaly nebo by nemohly být vůbec použity.

2.3.1 Service Management Facility

Service Management Facility neboli SMF je systém, který v operačním systému Solaris spravuje systémové služby. Nahrazuje tím tradiční způsob spravování služeb pomocí tzv. *init* skriptů, který byl běžný v ostatních unixových operačních systémech a dokonce i v dřívějších verzích OS Solaris. Hlavním rozdílem oproti staršímu způsobu je možnost u služby definovat závislosti na ostatních službách. Na rozdíl od sériového spouštění *init* skriptů z adresáře je díky tomuto zlepšení možné při startu systému paralelně spouštět více nezávislých služeb najednou, a tím urychlit start systému [8]. Pro účel startu jsou v systému definovány speciální služby tzv. *milestone*, které ve skutečnosti nic nedělají. Mají definovaný pouze seznam závislostí, který určuje jaké služby se mají spustit. Při startu se určí do kterého *milestone* má systém nastartovat a tím je přesně určeno, které služby se mají spustit.

2.3.2 Souborový systém ZettaByte

Pro ukládání na disk používá Solaris souborový systém ZettaByte neboli ZFS. Je to pokročilý systém, který byl vyvinut společností Sun Microsystems a integrován do operačního systému Solaris. ZFS dokáže spravovat velké množství

dat díky své 128-bitové architektuře [**?**]. Mezi hlavní funkce ZFS patří ověřování integrity dat, vlastní softwarový RAID nebo šifrování dat. Díky principu *copy on write* dokáže udržet data neustále konzistentní, což některé souborové systémy nedokážou nebo tento problém řeší složitě. Architektura tohoto souborového systému umožňuje klonování jednotlivých svazků nebo rychlou a elegantní tvorbu obrazů disku tzv. *snapshot*, které z počátku nezabírají téměř žádné místo na disku. Datové bloky jsou totiž zduplikovány až v okamžiku, kdy se zdrojový blok nebo jeho klon změní. Tento způsob uchovávání dat společně s možností *deduplikace* stejných datových bloků značně snižuje nároky na diskové místo.

Principů a funkcí ZFS hojně využívají další služby operačního systému Solaris. Jako příklad můžeme uvést virtualizační techniku Solaris Zones, která je hlavním tématem této diplomové práce.

2.3.3 Virtualizace

Dle specifikace [9] nabízí operační systém Solaris ve verzi 11.3 následující techniky virtualizace.

- Virtualizace na úrovni OS
- Virtuální počítače
- Hardware partitions

Tyto modely se liší zejména ve způsobu izolace virtualizovaných prostředí a ve flexibilitě přidělování prostředků těmto prostředím. Čím více model izoluje prostředí od sebe, tím nabízí menší flexibilitu v přidělování prostředků.

2.3.3.1 Solaris Zones

Jedním z modelů virtualizace nabízené operačním systémem Solaris je *virtualizace na úrovni OS*. Tento model umožňuje vytvořit jedno nebo více izolovaných prostředí (zón) pro běh programů v rámci jedné instance OS. Takto vytvořené prostředí jsou izolovány na úrovni procesů, souborového systému a síťových rozhraní. Každá zóna má vlastní lokální pohled na systémové prostředky, které mohou být dále virtualizované operačním systémem. Virtualizace na úrovni operačního systému poskytuje vysoký výkon a flexibilitu, protože nezanechává tak velkou stopu na diku, paměti nebo CPU na rozdíl od zbylých dvou modelů.

Operační systém Solaris poskytuje tento model virtualizace skrz službu Solaris Zones, která je přímo integrována to jádra OS.

2.3.3.2 Virtuální počítače

Model *virtuálních počítačů* popsaný v kapitole **??** umožňuje souběžný běh více instancí operačního systému na jednom fyzickém stroji. Konkrétně každý virtuální počítač má svojí instanci operačního systému, který nemusí být stejný

ve všech virtuálních strojích. Tento typ virtualizace je umožněn díky virtualizačnímu monitoru, který vytváří pro operační systémy iluzi, že jsou na fyzickém stroji samy. Virtuální počítače poskytují na rozdíl od virtualizace na úrovni OS menší flexibilitu rozdělování prostředků, ale naopak poskytuje větší úroveň izolace.

Tento typ virtualizace je v OS Solaris 11.3 podporován produkty Oracle VM Server for x86, Oracle VM Server for SPARC a Oracle VM VirtualBox. Každá z těchto implementací se zaměřuje na jinou architekturu nebo používá jiný typ virtualizačního monitoru.

2.3.3.3 Hardware partitions

Posledním modelem, který je ne přímo podporovaný operačním systémem Solaris, jsou tzv. *hardware partitions*. Je to technika, která fyzicky odděluje běh OS na oddělených částech fyzických prostředků. Tohoto způsobu virtualizace je dosaženo bez pomocí hypervisoru, a tudíž tato technika poskytuje reálný výkon systému. Hardware partitions je technika poskytující běžícím operačním systémům největší izolaci, ale na druhou stranu není tak flexibilní v konfiguraci prostředků jako výše zmíněné modely.

Tato technika není z logických důvodů poskytována OS Solaris, jelikož se jedná o virtualizaci na HW úrovni. Pro účely nasazení tohoto OS s touto virtualizační technikou používá Oracle speciální servery SPARC M-Series.

Kapitola 3

Solaris Zones

Závěr

Virtualizace se stala běžnou a možná i nezbytnou součástí dnešního počítačového světa.

Literatura

[1] University, O.: *Definition of virtual in English* [online]. [cit. 2018-03-26]. Dostupné z: https://en.oxforddictionaries.com/definition/virtual

[2] staff, V.: *The Virtues of Network Virtualization* [online]. [cit. 2018-03-26]. Dostupné z: https://www.vmware.com/ciovantage/article/the-virtues-of-network-virtualization

[3] Smith, J. E.; Nair, R.: .

[4] Kašpar, J.; Tvrdík, P.: *Techniky virtualizace II.* [online]. Praha, České vysoké učení technické v Praze, Fakulta informačních technologií [cit. 2018-03-12]. Dostupné z: https://edux.fit.cvut.cz/courses/MI-POA/_media/lectures/mi-poa09.pdf

[5] Kašpar, J.; Tvrdík, P.: *Techniky virtualizace I.* [online]. Praha, České vysoké učení technické v Praze, Fakulta informačních technologií [cit. 2018-03-12]. Dostupné z: https://edux.fit.cvut.cz/courses/MI-POA/_media/lectures/mi-poa08.pdf

[6] Arpaci-Dusseau, R. H.; Arpaci-Dusseau, A. C.: *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, 0 vydání, Květen 2015. Dostupné z: http://www.ostep.org

[7] Kašpar, J.; Tvrdík, P.: *Techniky virtualizace III.* [online]. Praha, České vysoké učení technické v Praze, Fakulta informačních technologií [cit. 2018-03-12]. Dostupné z: https://edux.fit.cvut.cz/courses/MI-POA/_media/lectures/mi-poa10.pdf

[8] Muzikář, Z.; Žďárek, J.: *Start systému, proces init, Solaris Service Management Facility* [online]. Praha, České vysoké učení technické v Praze, Fakulta informačních technologií [cit. 2018-03-23]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-ADU/_media/lectures/07/biadu_p07_start.pdf

[9] Oracle: *Introduction to Oracle® Solaris 11 Virtual Environment* [online]. [cit. 2018-03-23]. Dostupné z: `https://docs.oracle.com/cd/E53394_01/pdf/E54760.pdf`

Příloha A

Seznam použitých zkratek

DNS Domain Name System

HLL High Level Language

HW Hardware

ISA Instruction Set Architecture

JVM Java Virtual Machine

IT Information Technology

OS Operating System

PC Program Counter/Personal Computer

RAID Redundant Array of Independent Disks

SMF Service Management Facility

SPARC Scalable Processor Architecture

SW Software

TLB Translation Lookaside Buffer

VM Virtual Machine

VMM Virtual Machine Monitor

ZFS Zettabyte File System

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS