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.

25. dubna 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 25. dubna 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, nazýváme *host*.

Poslední vrstva, která zbývá popsat je samotný virtualizační SW. V případě systémových virtuálních strojů se typicky nazývá virtualizační monitor

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

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 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

Druhým typem virtuálních strojů jsou tzv. *systémové virtuální stroje*, které kompletně virtuálizují celou HW platformu. Virtualizační software je většinou umístěn hned na HW počítače a jeho hlavním úkolem je virtualizovat ISA. Tímto způsobem *systémové virtuální stroje* vytvářejí virtuální prostředí pro běh operačního systému a jeho aplikací a dokonce umožňují současný běh různých operačních systémů na jednom HW stroji.

Virtuální počítače

Nejznámějším a pravděpodobně nejpoužívanějším zástupcem z této kategorie virtuálních strojů jsou tzv. *virtuální počítače*. Tento typ virtuálního stroje umožňuje současný běh více operačních systému na jednom fyzickém počítači.

Všechny operační systémy musí využívat instrukce stejné instrukční sady, kterou vykonává host. Jinými slovy tento typ virtuálního stroje žádným způsobem nepřekládá instrukce do jiné instrukční sady.

Dle umístnění virtualizačního monitoru v architektuře počítače, můžeme rozdělit tento typ virtuálního stroje do dvou hlavních kategorií.

- Nativní VMM
- Hosted VMM

Hlavní rozdíl mezi těmito dvěma typy spočívá v umístění virtualizačního monitoru v architektuře počítače. Zatímco *nativní VMM* má přímou kontrolu nad HW počítače, *hosted VMM* běží uvnitř operačního systému, který spravuje HW prostředky. Oba tyto typy VMM jsou detailně popsány v kapitole 1.6.

Virtualizace celého systému

V případě, kdy operační systém a jeho aplikace používají stejnou instrukční sadu jako nižší HW vrstvy, má virtuálizační monitor za úkol spravovat přístup k HW a sdílet ho mezi virtuální počítače. Pokud bychom chtěli ve virtuální stroji spouštět operační systém používající odlišnou instrukční sadu, musíme zajistit emulaci prostředí. Jinými slovy VMM musí vytvořit virtuální prostředí pro konkrétní OS tak, aby si OS myslel, že běží na odpovídající HW platformě. Tento typ virtuálního stroje nazýváme *virtualizace celého systému* a jak je z názvu patrné, mimo instrukční sady můžou být virtualizovány i některá I/O zařízení.

Virtualizace celého systému se hojně využívá případech, kdy je nutné udržet v chodu systémy, který běží na speciálních HW platformách. V tomto případě se vytvoří emulace celého prostředí a systém se přesune do takto vytvořeného prostředí. Systém i jeho aplikace nepoznají žádný rozdíl, jelikož je celé prostředí virtualizováno. Tento typ virtuálního stroje je v mnoha ohledech podobný virtuálnímu stroji popsanému v 1.4.1.

Resource partitioning

Frekvence dnešních procesorů již není hlavním měřítkem jejich výkonosti, jak tomu kdysi bylo. Moderní procesory škálují svůj výkon s počtem výpočetních jader a díky tomu můžeme nezávisle spouštět více výpočetních úloh najednou. Tyto jádra jsou propojeny sdílenou pamětí, která zajišťuje prostor pro ukládání dat a komunikaci. Tohoto faktu využívá technika zvaná *resource partitioning*, která prostředky vícejádrového systému spojuje do částí.

Hard partitioning je technika, kdy jsou fyzické prostředky počítače rozděleny do disjunktních částí. Každá tato část se chová jako nezávislý systém a většinou má vlastní CPU (jádro), paměť, I/O zařízení i adaptér pro připojení

Obrázek 1.3: Konsolidace serverů

k síti. Jednotlivé fyzické komponenty nebo jejich části jsou tedy přímo přiřazeny konkrétní části (partition) systému. V takto rozděleném systému pak můžeme současně provozovat několik instancí operačních systémů. Tato technika zajišťuje velkou míru izolace pro běžící operační systémy, neboť každý OS má k dispozici vlastní disjunktní prostředí, ve kterém může operovat. Pokud nastane SW nebo HW chyby v jedné části (partition) systému, aplikace a OS běžící v ostatních částech nejsou nijak omezeny.

Druhým přístupem k rozdělování zdrojů je technika zvaná *logical partitioning*. V tomto případě fyzické prostředky nejsou exkluzivně přiděleny jednotlivým operačním systémům, ale dochází k jejich sdílení na SW úrovni. Ke sdílení procesoru a jeho jader se například může používat časový multiplexu, který umožňuje střídání OS ve využívání procesoru. Tato technika umožňuje lepší využití fyzických prostředků než *hard partitioning*, kde některé prostředky mohou kvůli malé zátěži zůstat nevyužity.

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.

Obrázek 1.4: Izolace aplikací

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 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

Obrázek 1.5: Migrace virtuálního stroje

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.

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.

Obrázek 1.6: Migrace fyzického na virtuální stroj

1.6 Virtualizační monitor

Jak již bylo zmíněno v definici z kapitoly **??**, 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

Obrázek 1.7: Nativní (Bare-Metal) VMM

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 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

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

Obrázek 1.8: Hostovaný VMM

Obrázek 1.9: VMM se servisním OS

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.

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 způsobem 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 ob-

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]

sluhu 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 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

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]

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.

- 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

V úvodu následující kapitoly jsou popsány základní principy a struktura virtualizační techniky Solaris Zones od firmy Oracle. Dále se tato kapitola zabývá popisem datových struktur, postupů a nástrojů, které slouží ke správě a manipulaci se zónami. Detailnější popis je věnován především administrátorským rutinám pro konfiguraci, instalaci a zálohování zón.

3.1 Virtualizační technika

Oracle Solaris Zones je virtualizační technika, která umožňuje běh více virtuálních strojů na jednom fyzickém stroji. Jak již bylo zmíněno v kapitole 2, tato technika je standardní součástí operačního systému Oracle Solaris od verze systému Solaris 10. Aktuální verze je Solaris 11.3 a přináší novou funkcionalitu v podobě podpory starších verzí operačního systému Solaris.

Pokud bychom chtěli zařadit Solaris Zones do klasifikace virtuálních strojů popsané v kapitole 1.4, pak bychom jí zařadili do sekce *virtualizace na úrovni OS*. Jinak řečeno, tato technika rozděluje zdroje hostitelského operačního systému jako CPU, paměť nebo I/O zařízení mezi běžící virtuální stroje a zajišťuje izolaci na úrovni procesů, souborového systému a sítě. Zónu pak můžeme definovat jako virtuální kontejner běžící v hostitelském operačním systému, který využívá zdrojů hostitelského operačního systému a je izolovaný od ostatních zón.

Standardně se po instalaci operačního systému Solaris nachází v systému jedna zóna. Je to vlastní instance operačního systému a nazývá se **globální** zóna. Jinými slovy, je to zóna, která běží přímo na hardwaru počítače nebo ve virtualizovaném prostředí. Tato zóna má dvě hlavní funkce. Za prvé tato zóna plní funkci hlavního operačního systému a přebírá kontrolu nad fyzickými prostředky po startu systému. Za druhé je hlavním centrálním prvkem pro administraci celého systému a ostatních zón. Globální zóna poskytuje globální pohled na celý systém a má přehled o všech systémových zdrojích a aktivitách ostatních zón. Její role v rámci Solaris Zones je zásadní a její chyba může

zapříčinit pád ostatních zón. Z tohoto důvodu je doporučené používat globální zónu pouze pro účely administrace systému a managementu ostatních zón.

Zóny, které jsou spouštěny v rámci globální zóny nazýváme **neglobální** zóny. Tyto zóny jsou navzájem izolované na několika úrovních. První úroveň izolace je izolace na úrovni sítě. Každá neglobální zóna může mít svůj vlastní logický síťový adaptér, který je vytvořený nad nějakým fyzickým síťovým rozhraním a je dostupný pouze pro konkrétní zónu. Z jiné neglobální zóny tento adaptér není přístupný. Takto vytvořený síťový adaptér může být spravován pouze z globální zóny a ze zóny, ke které byl přiřazen v průběhu jejího vytváření.

Druhou úrovní izolace zón je souborový systém. Globální zóna spravuje svůj vlastní souborový systém ve kterém se nachází standardní adresářová struktura operačních systému typu UNIX. Můžeme v něm nalézt adresář */etc* sloužící pro globální systémovou konfiguraci, adresář */bin* obsahující uživatelsky spustitelné programy nebo například adresář */sbin*, který uchovává programy spustitelné pod privilegovaným uživatelem. Každá neglobální zóna potřebuje pro svůj běh velmi podobné prostředí, a proto má svůj vlastní souborový systém, který se nachází někde v hierarchii souborového systému globální zóny. Podle typu zón popsaných v kapitole 3.1.1 může být tento souborový systém částečně sdílený se souborovým systémem globální zóny a nebo může obsahovat kompletně nezávislý obraz zóny. Při spouštění zóny pak dojde pomocí příkazu `chroot(1)` k přepnutí kořenu souborového systému a zóna pracuje pouze se svojí částí souborového systému. V případě sdílené části souborového systému jsou tyto části připojeny v režimu read-only [5]. Tím je zajištěno, že souborové systémy jednotlivých zón jsou vzájemně izolované a nemohou se vzájemně ovlivnit. Správu těchto souborových systémů je opět možné provádět pouze z konkrétní zóny a nebo ze zóny globální.

Mimo izolace na úrovni sítě a souborového systému Solaris Zones implementuje ještě izolaci na úrovni procesů. Každá neglobální zóna má svůj plánovač a může spouštět svoje vlastní procesy. Procesy běžící v jedné neglobální zóně nejsou žádným způsobem viditelné ani přímo ovlivnitelné z jiných neglobálních zón. Pokud chce proces z jedné zóny komunikovat s procesem z jiné zóny, nemůže k tomu využít mezi procesovou komunikaci, ale musí použít počítačové sítě. Naopak procesy, které běží v rámci jedné zóny spolu mohou komunikovat pomocí signálů, sdílené paměti a tak podobně. Všechny procesy běžící v systému mohou být spravovány z globální zóny. Globální zóna tedy nabízí globální přehled všech procesů, které jsou spuštěny ve všech běžících zónách v systému. Výstup příkazu `ps(1)` v globální zóně zobrazí všechny procesy, zatímco v neglobální zóně budou zobrazeny pouze procesy příslušící dané zóně.

3.1.1 Typy zón

Jak bylo zmíněno výše, virtualizační technika Solaris Zones umožňuje v rámci jedné globální zóny spouštět mnoho neglobálních zón. Každá neglobální zóna má vlastnost zvanou *brand*, která určuje typ neglobální zóny. Tato vlastnost se specifikuje při konfiguraci zóny v globální zóny a dle přehledu [10] může typ zóny mít následující hodnoty:

- *solaris*
- *solaris-kz*
- *solaris10*

Brand neboli typ zóny určuje jakým způsobem se zóna bude po spuštění chovat. Implicitním typem zóny v Solaris Zones je *solaris*, kterému se také jinak přezdívá nativní zóna nebo také tenká zóna. Dalším typem zóny je *solaris-kz*, kde zkratka za pomlčkou v názvu odpovídá slovnímu spojením kernel zone. Jak název napovídá, tato zóna má vlastní jádro operačního systému a někdy se jí také přezdívá plná nebo tlustá zóna. Posledním typem zón, kterou Solaris Zones umí vytvářet je *solaris10*. Hlavním úkolem této zóny je zajišťovat zpětnou kompatibilitu s operačním systémem Solaris 10 a umožňuje uvnitř této zóny spouštět aplikace určené pro tento systém.

V následujících podkapitolách jsou podrobněji popsány výše zmíněné typy zón.

3.1.1.1 Nativní zóna

Nativní neboli tenká zóna umožňuje administrátorovi vytvořit zónu, která má sdílené jádro operačního systému s globální zónou. Jinými slovy verze jádra operačního systému musí být stejná jako v globální zóně. Tento typ zóny je izolovaný pouze nad svým souborovým systémem a nemá standardně nemá k dispozici informace o žádném fyzickém zařízení systému. Souborové systémy ostatních zón jsou nedostupné a konkrétní neglobální zóna o nich nemá žádné informace. Z jejího pohledu existuje pouze její kořenový souborový systém. Jak již bylo popsáno výše, kořenový sytém nativní zóny může být sdílený se souborovým systémem globální zóny a sdílet tak základní systémové nástroje. Pokud chceme této zóně delegovat nějaký typ zařízení, musíme tak učinit při konfiguraci zóny v globální zóně. Tímto způsobem můžeme nativní zóně zpřístupnit souborové systémy, ZFS pool nebo ZFS dataset. Takto definované prostředky pak po instalaci zóny můžeme z této zóny využívat.

Dále tento typ zóny má svoji vlastní databázi produktů, která obsahuje informace o všech nainstalovaných softwarových komponentech v konkrétní globální zóně. Opět platí, že konkrétní neglobální zóna vidí pouze své balíčky. Díky tomu je možné instalovat dodatečné softwarové balíčky do neglobálních zón, které nemusí být nainstalované v globální zóně [10]. Některé softwarové

balíčky jsou ale společné s globální zónou (kernel) a nelze tedy provádět kompletní aktualizaci bez zásahu do globální zóny.

Nativní zóna podporuje dva typy síťových rozhraní, které mohou být zóně při konfiguraci přiřazeny. Prvním typem je sdílená adresa neboli *shared-ip*. Tento typ síťového rozhraní sdílí ip adresu s nějakým fyzickým rozhraním globální zóny. Pokud chce neglobální zóna komunikovat s okolím, bude v hlavičce paketu ip adresa globální zóny a při obdržení odpovědi globální zóna přesměruje paket na virtuální síťové rozhraní konkrétní globální zóny. Zde můžeme pozorovat podobnost s technikou NAT v sítích. Jako druhý typ rozhraní můžeme použít exkluzivní rozhraní nebo také *exclusive-ip*, které nesdílí ip adresu s globální zónou, ale má svoji vlastní. V tomto případě veškerý síťový provoz generovaný touto zónou bude mít v hlavičce jinou ip adresu než zóna globální.

Tento typ zóny nepodporuje vytváření další neglobálních zón. Jinými slovy se nativní neglobální zóna nemůže chovat jako globální zóna a vytvářet nové zóny uvnitř sebe. Stejně tak z nativní zóny nemůžeme vytvářet ani spravovat jiné neglobální zóny.

Nativní zóna je implicitní typ zóny v Solaris Zones a pokud administrátor nespecifikuje jinak při vytváření zóny, bude nově vytvořená zóna právě typu *solaris*. Tento typ zóny může být provozován na všech systémech, které podporují operační systém Oracle Solaris 11.3 [10].

3.1.1.2 Kernel zóna

Druhým typem zóny, který virtualizační technika Solaris Zones umožňuje vytvářet, je kernel zóna nebo také tlustá zóna. Tento typ zóny obsahuje vlastní jádro operačního systému a na rozdíl od nativní zóny ho nesdílí s globální zónou. Kernel zóna tedy může být provozována na jiné verzi jádra než globální zóna. V důsledku toho kernel zóna podporuje funkcionalitu, které nelze pomocí nativní zóny dosáhnout.

Stejně jako v případě nativní zóny i kernel zóna obsahuje vlastní databázi instalovaných softwarových balíčků. Jelikož i kernel zóna je neglobální, nelze z ní žádným způsobem vidět balíčky ostatních zón. Na rozdíl od nativní zóny administrátor může provádět aktualizaci všech balíčků, protože kernel zóna nesdílí nic s globální zónou. Žádné balíčky tedy nejsou závislé na balíčkách globální zóny.

V případě síťových rozhraní kernel zóny podporují pouze rozhraní typu *exclusive-ip* a neumožňuje sdílet síťovou adresu s globální zónou.

Na rozdíl od nativní zóny se kernel zóny mohou chovat jako globální zóny uvnitř hostitelské globální zóny. Jinými slovy je možné uvnitř kernel zóny vytvářet další neglobální zóny a vytvářet tak hierarchickou strukturu virtuální strojů. Je na zvážení administrátora, jestli daný scénář vyžaduje tuto strukturu.

Požadavky

Jelikož provoz kernel zóny se liší od provozu standardní nativní zóny, liší se i požadavky na hostitelský systéme. Požadavky se liší v závislosti na platformě. Pro jednoduchost si uvedeme pouze požadavky pro systémy s architekturou x86 a procesorem intel. Podle specifikace [11] se na hostitelský systém kladou následující požadavky.

- Procesor musí být typu Nehalem nebo novější
- Virtualizace CPU (VT-x)
- Podpora virtualizace paměti (RVI, EPT)
- Ochrana paměti

Výše zmíněné požadavky kladou nároky na HW vybavení hostitelského systému. Spolu s těmito požadavky musí být v globální zóně nainstalovaný balíček *brand/brand-solaris-kz*, který umožňuje vytváření kernel zón. Administrátor může pomocí příkazu `virtinfo(1)` zjistit, jaký typ virtualizace je v globální zóně podporován. Výpis programu na virtualizované platformě VMvare, kde jsou splněné výše zmíněné požadavky může vypadat následovně.

```
zadmin@shost:~$ virtinfo
NAME            CLASS
vmware          current
non-global-zone supported
kernel-zone     supported
```

3.1.1.3 Branded zóna

Branded zóny byly vytvořeny pro zpětné zajištění kompatibility se staršími verzemi operačního sytému Solaris. Díky technologii *BrandZ* [10] umožňují spouštění aplikací určených pro operační systém Solaris 10 na systému s OS Solaris 11. Aplikace mohou běžet v nezměněné formě v bezpečném prostředí, které je zajištěno neglobální zónou.

Z pohledu administrátora se tento typ zóny chová stejně jako nativní zóna a má stejné vlastnosti, které jsou popsané v podkapitole 3.1.1.1.

3.1.1.4 Shrnutí

Virtualizační technologie Solaris Zones umožňuje vytvářet neglobální zóny uvnitř primární globální zóny. Neglobální zóny poskytují izolované prostředí pro nezávislý a bezpečný běh aplikací. Zóny jsou izolovány na úrovni počítačové sítě, souborového systému a běžících procesů, čímž je zajištěno, že se vzájemně nemohou přímo ovlivňovat. Jediný způsob komunikace procesů z jiných zón je pomocí počítačové sítě.

Tabulka 3.1: Porovnání typů zón a jejich vlastností

Neglobální zóna může být několika druhů, které poskytují různé vlastnosti. Tabulka 3.1 poskytuje stručný přehled základních vlastností jednotlivých druhů zón.

3.2 Administrace

Globální zóna slouží hlavně pro účely správy hostitelského systému a všech neglobálních zón. Poskytuje nainstalovaným zónám prostředky pro jejich běh a spravuje informace o jejich stavu. Je to klíčové místo pro správu celého systému a správných chod neglobálních zón vyžaduje bezproblémový chod globální zóny. Administrátor takového systému by tento fakt vzít v úvahu a nepoužívat globální zónu jako zdroj pro spouštění uživatelských aplikací.

Proces vytváření zón se skládá ze dvou částí. První částí je konfigurace neglobální zóny, kdy administrátor specifikuje jaké parametry má zóna mít a jaké prostředky bude moci využívat. Tento proces zavede zónu do databáze globální zóny a od této chvíle je registrovaná v systému. K tomuto účelu nabízí Solaris Zones nástroj `zonecfg(1)`. Více o tomto nástroji a možnostech konfigurace zón je popsáno v kapitole 3.3.

Z předchozího procesu vznikne jakýsi recept na to, jak danou neglobální zónu vytvořit. Nyní je třeba vytvořit souborový systém zóny se všemi balíčky, které zóna bude potřebovat ke svému běhu. Této fázi se říká instalace zóny a v jejím průběhu se do kořenového souborového systému nahrávají určené balíčky a nastavuje se konfigurační profil zón. Pro tento účel slouží nástroj `zoneadm(1)`. Výstupem tohoto procesu je nainstalovaná zóna připravená ke spuštění. Detailněji nástroj pro instalaci zón a proces instalace popisuje kapitola 3.4.

Čerstvě nainstalovaná zóna může být opět pomocí nástroje `zoneadm(1)` spuštěna a následně se do ní může privilegovaný uživatel přihlásit pomocí nástroje `zlogin(1)`.

3.2.1 Administrátor

Jelikož neglobálních zón může být v systému provozováno velké množství, může být pro administrátora globální zóny spravovat celou globální zónu a přitom se starat o všechny neglobální zóny. Pro tento účel může být vytvořen uživatel, který se bude starat výhradně jenom o neglobální zóny. Tento uživatel bude mít práva na správu všech neglobálních zón.

Pokud by i tak bylo neglobálních zón mnoho, je možné jednotlivým neglobálním zónám přiřadit vlastního administrátora. Ten se bude moc do zóny přihlásit pomocí příkazu `zlogin(1)` a provádět údržbu a správu systému.

3.2.2 Stavový model zón

V Solaris Zones má neglobální zóna definovaný stavový model. Jsou to stavy, ve kterých se zóna během jejího životního cyklu může nacházet. Zónu můžeme převést z jednoho stavu do druhého pouze nějakou kombinací nástrojů `zonecfg(1)` a `zoneadm(1)`. Podle specifikace [12] se neglobální zóna může nacházet v jednom z následujících sedmi stavů.

- *Configured*
- *Incomplete*
- *Unavailable*
- *Installed*
- *Ready*
- *Running*
- *Shutting down/Down*

V každém z těchto stavů může administrátor používat pouze nějakou podmnožinu příkazů, které zónu ovládají. Pro příklad můžeme uvést, že nelze zónu spustit pokud se nachází například ve stavu *configured*. Pro spuštění se zóna musí nacházet ve stavu *installed*. V následujících odstavcích je stručně shrnut význam jednotlivých stavů.

3.2.2.1 Configured

Stav *configured* značí, že konfigurace zóny je hotová a uložená na perzistentním úložišti. V tuto chvíli se zónou ještě nebyl asociován žádný diskový obraz a tedy nemá připojený kořenový souborový systém. Tento stav je v pořadí první stav, ve kterém se zóna může od svého vzniku nacházet. Nachází se v něm buď bezprostředně po vytvoření konfigurace pomocí `zonecfg(1)` nebo když je zóna odinstalována nebo odpojena.

3.2.2.2 Incomplete

Zóna se nachází ve stavu *incomplete* během procesu instalace a odinstalace. Je to přechodný stav, ale v případě poškození nainstalované zóny může být v tomto stavu stále. V případě úspěchu procesu instalace pomocí nástroje `zonecfg(1)` je stav zóny změněn na stav *installed*. Pokud uspěje proces odinstalování zóny pomocí stejného nástroje, je stav změněn na stav *configured*.

3.2.2.3 Unavailable

Ve stavu *unavailable* se zóna nachází v případě, kdy zóna byla v minulosti nainstalována, ale momentálně nemůže být spuštěna, přesunuta nebo její validace vrací chybu. Tento stav může mít několik příčin. Jednou z nich může být nedostupnost zdrojového souborového systému zóny. Souborový systém může být nedostupný chybou administrátora nebo například chybou diskového zařízení. Další příčinou může být nekompatibilita softwarového vybavení globální zóny a neglobální zóny. To se může stát například ve chvíli kdy migrujeme neglobální nativní zónu z jednoho systému na druhý a tyto dva systémy mají odlišnou verzi jádra.

3.2.2.4 Installed

Stav *installed* signalizuje, že zóna s danou konfigurací je nainstalovaná ve svém kořenovém souborovém systému, ale nemá alokovanou žádnou virtuální platformu pro svůj běh. Jinými slovy ještě nemůže být přímo spuštěna. Zóna ve stavu *installed* již může být zálohována nebo migrována mezi různými hosty.

3.2.2.5 Ready

Zóna se nachází ve stavu *ready*, právě když je pro ni alokována virtuální platforma pro její běh. To znamená, že jádro hostitelského operačního systému Solaris vytvořilo proces `zsched`, vytvořilo virtuální síťové rozhraní a zpřístupnilo je neglobální zóně. Obecně jádro inicializovalo všechny prostředky specifikované v konfiguraci zóny a zpřístupnilo je dané neglobální zóně. V tomto stavu ještě nebyl spuštěn žádný uživatelský proces asociovaný s konkrétní zónou [12]. Tento stav je tranzitní a nastává v okamžiku kdy zahajujeme boot zóny pomocí příkazu `zoneadm(1)`.

3.2.2.6 Running

Ve stavu *running* se zóna nachází když je spuštěn první uživatelský proces. Většinou se jedná o proces `init(1)`, který inicializuje celou zónu a umožňuje spouštění procesů uvnitř dané neglobální zóny. Zóna ve stavu *running* má tedy alokovanou virtuální platformu v jádru hostitelského operačního systému, inicializované všechny zařízení a spuštěné uživatelské procesy.

3.2.2.7 Shutting down/Down

Posledním stavem respektive dvojicí stavů, ve kterých se může neglobální zóna nacházet jsou stavy *shutting down* resp. *down*. Tyto stavy jsou tranzitní a nastávají ve chvíli, kdy daná zóna zastavuje svůj běh. V případě, kdy nelze zónu z nějakého důvodu zastavit, může daná zóna setrvat v některém z těchto dvou stavů [12].

3.2.2.8 Doplňkové stavy kernel zón

Výše zmíněné stavy jsou společné pro všechny typy zón. Nastávají tedy u nativních zón, kernel zón i branded zón. Pro kernel zóny existují ještě další stavy, které zmíníme jenom v rychlosti, jelikož nejsou tolik podstatné. Jedná se o stavy *suspended*, *debugging*, *panicked*, *migrating-out* a *migrating-in*. Jména stavů jsou samovysvětlující, a proto nemá smysl je podrobněji popisovat.

3.3 Konfigurace

Prvním krokem k vytvoření zóny je zaregistrování její konfigurace do systému Solaris Zones. K tomuto účelu se používá nástroj `zonecfg(1)`, který umožňuje vytváře, měnit nebo mazat konfigurace jednotlivých neglobálních zón. Dle manuálových stránek [13] Tento nástroj umožňuje administrátorovi zadávat konfiguraci ve třech následujících režimech.

- Interaktivně
- Dávkově
- Pomocí souboru s příkazy

První režim zadávání konfigurace vyžaduje aktivní účast uživatele a umožňuje interaktivně zadávat příkazy, které definují konfiguraci dané zón. Další dva režimy načítají příkazy k definici konfigurace z jiného zdroje než je interaktivní vstup uživatele. V případě dávkového režimu se jedná o vstup příkazů na příkazové řádce, kdy jsou příkazy zřetězeny za sebe a předány nástroji `zonecfg(1)` jako parametr. V druhém případě jsou příkazy načteny ze souboru, kde je každý příkaz na samostatné řádce.

Všechny režimy mají jednu věc společnou. Tím jsou příkazy, kterými předávají nástroji informace o definici konfigurace zóny. Nástroj tedy očekává přesně definovanou syntaxi příkazů, které umí zpracovávat. Konfigurace zóny se skládá z konfigurace globálních atributů a prostředků. Prostředky mohou reprezentovat síťové rozhraní nebo jiný typ zařízení a mají svoje vlastní lokální atributy. Popis všech příkazů, které nástroj `zonecfg(1)` umí zpracovávat je zbytečný, a proto si syntaxi ukážeme na následující výpisu 3.1 konfiguračního souboru zóny. V ukázce 3.1 je patrné, že každá řádka začíná příkazem. Příkaz `create` vytvoří v paměti reprezentaci konfigurace, která se po dokončení procesu konfigurace uloží do souboru. Následuje sekvence příkazů `set`, které nastavují nějaké globální atributy zóny. Více o nich v následující podkapitole 3.3.1. Dále můžeme v konfiguračním souboru vidět příkaz `add`, který reprezentuje přidání zdroje k zóně. V tomto případě se jedná o přidání síťového rozhraní s automatickou konfigurací. Základní typy jednotlivých prostředků popisuje podkapitola 3.3.1. Následuje opět sekvence příkazů `set`, která se ale nyní váže k předchozímu příkazu `add` a nastavuje tak lokální atributy daného

Kód 3.1: Ukázka konfigurace zóny

```
zadmin@shost:~$ cat /var/tmp/rzone-test_hu67.zonecfg
create -b
set brand=solaris
set zonepath=/system/zones/rzone-test
set autoboot=false
set autoshutdown=shutdown
set ip-type=exclusive
add anet
set linkname=net0
set lower-link=auto
set configure-allowed-address=true
set link-protection=mac-nospoof
set mac-address=auto
end
```

síťového rozhraní. Celý konfigurační soubor je ukončený příkazem `end`, který reprezentuje výstup z konfigurace prostředku.

Nástroj `zonecfg(1)` umožňuje dva módy editace konfigurace zóny. Prvním mód je editace konfigurace uložené v souborovém systému. Změna konfigurace v tomto módu, žádným způsobem neovlivní běžící zónu. Druhý způsobem je editace konfigurace v takzvaném živém módu, která umí ovlivňovat nastavení zóny ve stavu *running*. V tomto případě je například možné dočasně přidat běžící zóně síťové rozhraní.

3.3.1 Globální atributy

Globální atributy popisují globální vlastnosti zóny jako celku. Neváží se tedy ke konkrétnímu prostředku ale k zóně jako takové. Podle manuálových stránek [13] umožňuje příkaz `zonecfg(1)` konfigurovat třináct globálních atributů zóny. V této kapitole nebudeme popisovat všechny, ale zaměříme se na nejdůležitější z nich. Důraz bude kladen především na atributy, které jsou nezbytné pro vytvoření zóny.

3.3.1.1 Jméno zóny

Jedním z hlavních atributů zóny je její jméno. Tento atribut je hlavním identifikátorem zóny v rámci systému a používá se pro její specifikaci v rámci nástrojů `zonecfg(1)` a `zoneadm(1)`. Nastavuje se pomocí vlastnosti *zonename*, nemá žádnou implicitní hodnotu a je povinným atributem pro vytvoření neglobální zóny.

3.3.1.2 Cesta k souborovému systému zóny

Klíčovým atributem zóny je cesta k adresáři, kde je připojený kořenový souborový systém neglobální zóny. Tento adresář obsahuje všechny nezbytné soft-

warové balíčky pro běh zóny. V operačním systému Solaris 11 se pro kořenové souborové systémy zón používá souborový systém ZFS. Tato skutečnost umožňuje využívat pokročilé funkce ZFS jako například snapshot nebo klonování při správě a instalaci neglobálních zón. Tento atribut se nastavuje pomocí vlastnosti *zonepath* a je povinným atributem pro vytvoření zóny. V jeho definici je možné používat proměnou *zonename* a jeho implicitní hodnota je nastavena na `/system/zones/%{zonename}`.

3.3.1.3 Typ zóny

Jak již bylo zmíněno v kapitole 3.1.1, typ neglobální zóny určuje jak s ní bude globální zóna zacházet. Konfigurace typu zóny se provádí pomocí atributu *brand*, který je povinným atributem pro vytvoření zóny. Může nabývat hodnot *solaris*, *solaris-kz* nebo *solaris10* a jeho implicitní hodnota je *solaris*.

3.3.1.4 Typ IP adresy

Stejně jako předchozí atribut byl i atribut určující typ síťové adresy již zmíněn v kapitole 3.1.1. V krátkosti pouze naznačíme, že tento atribut určuje jestli bude síťová adresa sdílená s adresou globální zóny či nikoli. Tento atribut se nastavuje pomocí atributu *ip-type* a může mít hodnoty *shared* a *exclusive*, což je zároveň implicitní hodnota.

3.3.1.5 Automatické spouštění a vypínání

Jako poslední zmíníme dva atributy, které souvisí se spouštěním a vypínáním neglobálních zón. Atribut *autoboot* vyjadřuje jestli se má daná neglobální zóna spustit při startu zóny globální. Tento atribut může nabývat dvou hodnot. Jestliže chceme aby se zóna spustila při startu globální zóny, musíme nastavit hodnotu tohoto atributu na *true*. V opačném případě nastavíme hodnotu *false*, která je implicitní hodnotou tohoto atributu. O automatické spouštění neglobálních zón se stará systémová služba `svc:/system/zones:default`, a proto je nutné, aby byla spuštěna [13].

Druhým atributem je atribut *autoshutdown*, který se uplatňuje při vypínání globální zóny a naznačuje co se má stát s danou neglobální zónou. Jeho hodnoty mohou být *shutdown* pro korektní vypnutí zóny, *halt* a nebo *suspend*. Implicitně se při vypínání globální zóny používá korektní vypnutí neglobální zóny.

3.3.2 Zdroje

Mimo generických atributů umožňuje nástroj `zonecfg(1)` přidávat do konfigurace zóny takzvané zdroje. Zdroj je objekt nějakého typu, který má svoje lokální atributy a do konfigurace zóny se přidává pomocí příkazu `add`. Zdroj

Kód 3.2: Konfigurace zařízení kernel zóny

```
add device
set storage=/dev/zvol/dsk/rpool/VARSHARE/zones/z1/disk
set bootpri=0
set id=0
end
```

reprezentuje většinou nějaké zařízení, souborový systém, prostředky pro přidělování zdrojů zónám nebo například specifikuje uživatele, který může zónu administrovat. Některé zdroje mohou být do konfigurace zóny přidány vícekrát. V takovém případě jim `zonecfg(1)` automaticky přidělí číselný identifikátor, který daný zdroj jednoznačně určuje. Podle manuálových stránek [13] umožňuje konfigurace zón přidávat až jednadvacet různých typů zdrojů. Stejně jako v případě globálních atributu si zde popíšeme jenom zdroje, které jsou nejdůležitější pro správný běh neglobální zóny.

3.3.2.1 Zařízení

Prvním typem zdroje, který může být delegován neglobální zóně, je obecné zařízení. Takovým zařízením může být například disk nebo disková partition. V případě operačního systému se jedná o takzvané *slice*, které jsou alternativou k diskovým partition.

Použití tohoto zdroje se liší v závislosti na typu neglobální zóny, ke které ho chceme přiřadit. V případě nativní zóny má tento zdroj následující atributy.

- *match*
- *allow-partition*
- *allow-raw-io*

Atribut *match* odpovídá jménu zařízení, které chceme delegovat zóně. Hodnotou může být absolutní cesta k zařízení nebo regulární výraz, který může specifikovat více zařízení najednou. Druhý atribut *allow-partition* určuje, jestli zóna bude moci používat nástroj `format(1)`, který umožňuje rozdělování disku na jednotlivé partition. Přímý přístup na disk může být povolen pomocí atributu *allow-raw-io*.

V případě kernel zóny je podle manuálových stránek [14] povinné přidat alespoň jedno diskové zařízení, které bude sloužit jako hlavní disk. Implicitně je pro kernel zónu vytvořen souborový systém ZFS, který je vyexportovaný jako zařízení. Toto zařízení se přidá v průběhu konfigurace a nastaví se mu atribut *bootpri* na hodnotu 0 (primární disk). Část konfigurace specifikující úložné zařízení kernel zóny je naznačena v kódu 3.2 Podle doporučení v [13] může být nebezpečné delegovat zónám obecné zařízení a povolit na ně přímý přístup.

Kód 3.3: Konfigurace síťového rozhraní

```
add anet
set lower-link=auto
set configure-allowed-address=true
set link-protection=mac-nospoof
set mac-address=auto
end
```

Tento krok může vést k nestabilitě a ohrožení globální zóny, jelikož uživatelské procesy neglobálních zón mohou přímo ovlivňovat delegovaná zařízení.

3.3.2.2 Síťové rozhraní

Jelikož spolu neglobální zóny nemohou komunikovat jinak než pomocí počítačové sítě, nástroj `zonecfg(1)` umožňuje delegovat neglobálním zónám následující dva typy síťových rozhraní.

- Fyzické síťové rozhraní - *net*
- Automatické síťové rozhraní - *anet*

Fyzické síťové rozhraní neboli zdroj *net* umožňuje delegovat do neglobální zóny existující fyzické síťové rozhraní z globální zóny. Tento typ zdroje má několik lokálních atributů, které definují jeho zdroj a chování. Hlavním atributem je *physical*, který reprezentuje jméno fyzického rozhraní v globální zóně. Jedno fyzické rozhraní nemůže být sdíleno napříč více neglobálními zónami. Pomocí další atributů je možné specifikovat například ip adresy, které se mohou k danému rozhraní připojovat, gateway nebo ip adresu rozhraní.

Automatické síťové rozhraní neboli *anet* je druhým síťového zdroje, který může být neglobální zóně přiřazen. Rozdíl oproti předchozímu typu je v tom, že tento typ rozhraní nemusí v globální síti existovat a je vytvořeno jako virtuální síťové rozhraní. Jelikož je toto zařízení virtuální, umožňuje administrátorovi nastavovat mnohem větší škálu atributů. Těchto atributů je podle manuálových stránek [13] opravdu velké množství a opět si popíšeme jenom některé z nich. Nejdůležitějšími atributy jsou *linkname* a *lower-link*. Atribut *linkname* určuje jméno síťového rozhraní tak, jak se bude jevit v neglobální zóně. Pomocí tohoto jména je možné toto rozhraní konfigurovat. Atribut *lower-link* je v podstatě jméno fyzického nebo virtuálního síťového rozhraní v globální zóně. Jinými slovy přes toto zařízení bude proudit provoz generovaný vytvořeným síťovým rozhraním. Další atributy již nebudeme podrobně zmiňovat. Ostatní parametry slouží například k nastavování MAC adresy, ochrany linkové vrstvy, VLAN a mnohého dalšího. V ukázce 3.3 je naznačeno jak by mohla vypadat konfigurace automatického síťového rozhraní pomocí nástroje `zonecfg(1)`.

3.3.2.3 Řízení zdrojů

Jak již bylo zmíněno výše, neglobální zóny využívají fyzických prostředků hostitelského systému (globální zóny). Aby bylo možné zajistit kontrolu na přidělováním prostředků jednotlivým neglobálním zónám, umožňuje konfigurace specifikovat některé omezení na využívání zdrojů. Tyto omezení se do konfigurace přidávají jako kterékoli jiné zdroje. Podle manuálových stránek [13] existují následující typy zdrojů, které umožňují řízení přidělování fyzických zdrojů.

- Exkluzivní CPU - *dedicated-cpu*
- Omezení CPU - *capped-cpu*
- Omezení paměti - *capped-memory*

První ze zdrojů s názvem *dedicated-cpu* slouží pro alokování určitého počtu procesorů a procesorových jader exkluzivně pro použití danou neglobální zónou. V systému dojde k vytvoření množiny procesorů, která v době běhu dané neglobální zóny může být využívána pouze danou neglobální zónou. Dokonce ani globální zóna nemůže tyto procesory a jádra využívat. Tento typ zdroje umožňuje specifikovat výpočetní zdroje s různou granularitou. Administrátor může zóně přiřadit prostředky na úrovni procesorových jader, procesorů nebo dokonce celých soketů s několika procesory.

Zdroj *capped-cpu* reprezentuje množství procesorového času, které může být danou zónou využíváno. Tento zdroj má pouze jeden numerický atribut, kterým je desetinné číslo určující možné procentuální využití jednoho procesoru. Hodnotou 1 se tedy myslí, že daná zóna může využít 100% procesorového času.

Posledním zdrojem, který si zmíníme v rámci řízení zdrojů zón je *capped-memory*. Tento zdroj se podobá *capped-cpu* ale řídí využití paměti. Tento zdroj má tři atributy *physical*, *swap* a *locked*, které se týkají určitého druhu paměti. Atribut *physical* souvisí s hlavní operační pamětí počítače a umožňuje administrátorovi specifikovat, kolik hlavní paměti může daná zóna využít. Ostatní atributy mají stejný význam, ale týkají se jiné části paměti. Všechny atributy je možné specifikovat v jednotkách kilobyte (K), megabyte (M), gigabyte (G) nebo terabyte (T).

3.3.2.4 ZFS Dataset

Standardně mají neglobální zóny ponětí pouze o svém kořenovém souborovém systému a nevidí žádný jiný. Pokud chce zóna využívat nějaké další úložiště, musíme jí delegovat buď celý disk nebo můžeme použít zdroj *dataset*. Tento zdroj reprezentuje existující souborový systém ZFS v globální zóně, který je delegován do neglobální zóny jako virtuální ZFS pool. Tento zdroj má dva

Kód 3.4: Delegace administrace jinému uživateli

```
add admin
set user=zadmin
set auths=login,manage
end
```

atributy specifikující jméno souborového systému v globální zóně a alias pod kterým bude vytvořen virtuální ZFS pool v neglobální zóně.

3.3.2.5 Administrátor zóny

Jak již bylo zmíněno v kapitole 3.2.1, administrátor globální zóny může delegovat administraci neglobální zóny na jiného uživatele. K tomuto účelu se požívá zdroj *admin*, který reprezentuje administrátora dané neglobální zóny. Tomuto administrátorovi lze přiřadit různé privilegia, které mu umožňují nějakým způsobem spravovat zónu. Podle manuálových stránek [13] jsou možná privilegia následující.

- *login*
- *manage*
- *copyfrom*
- *config*
- *liveconfig*

Názvy jednotlivých privilegií odpovídají akcím, které může daný uživatel se zónou provádět. Za zmínění snad stojí jenom privilegium *copyfrom*, které umožňuje administrátorovi vytvářet nové zóny jako klony dané neglobální zóny. V kódu 3.4 je ukázána konfigurace která umožňuje uživateli *zadmin* přihlašování a základní správu dané zóny.

3.3.3 Vytvoření konfigurace

3.3.4 Změna konfigurace

3.3.5 Smazání konfigurace

3.4 Instalace

3.5 Zálohování

Kapitola 4

Návrh aplikace

Kapitola 5

Implementace

5.1 Návrh

5.2 Řešení

5.2.1 Šablonovací systém

5.2.2 Šablonovací systém

Kapitola 6

Testování

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.: The architecture of virtual machines. *Computer*, ročník 38, č. 5, Květen 2005: s. 32–38, ISSN 0018-9162, doi: 10.1109/MC.2005.173.

[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 tech-

nické 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

[10] Oracle: *Introduction to Oracle® Solaris 11. Zone Brand Overview* [online]. [cit. 2018-04-24]. Dostupné z: https://docs.oracle.com/cd/E53394_01/html/E54762/gitrc.html

[11] Oracle: *Hardware and Software Requirements for Oracle Solaris Kernel Zones* [online]. [cit. 2018-04-24]. Dostupné z: https://docs.oracle.com/cd/E53394_01/html/E54751/gnwoi.html

[12] Oracle: *Zone Administration Overview* [online]. [cit. 2018-04-24]. Dostupné z: https://docs.oracle.com/cd/E53394_01/html/E54762/gqhar.html

[13] Oracle: *Man pages section 1M: System Administration Commands. zonecfg(1M)* [online]. [cit. 2018-04-24]. Dostupné z: https://docs.oracle.com/cd/E36784_01/html/E36871/zonecfg-1m.html

[14] Oracle: *Man pages section 7: Standards, Environments, Macros, Character Sets and Miscellany. zones_solaris-kz(7)* [online]. [cit. 2018-04-25]. Dostupné z: https://docs.oracle.com/cd/E88353_01/html/E37853/zones-solaris-kz-7.html

Příloha A

Seznam použitých zkratek

BIOS Basic Input Output System

DNS Domain Name System

EPT Extended Page Tables

HW Hardware

ISA Instruction Set Architecture

IT Information Technology

NAT Network Address Transaltion

OS Operating System

PC Program Counter/Personal Computer

RVI Rapid Virtualization Indexing

SW Software

SMF Solaris Management Facility

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
```