Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informačních technologií

Zjišťování efektivity virtualizovaných Linuxových distribucí pod hypervisory XEN a KVM

(úvod do virtualizace)

Bakalářská práce

Autor: Daniel Čulík
Studijní obor: Informační management

Vedoucí práce: Mgr. Josef Horálek, Ph.D.

Hradec Králové Srpen 2016

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 12.8.2016 Daniel Čulík

Poděkování:

Rád bych poděkoval Mgr. Josefu Horálkovi, Ph.D. za vedení mé práce a za cenné náměty pro její vypracování.

Anotace

Tato bakalářská práce se zaměřuje na oblast virtualizací, speciálně na sledování efektivity linuxových distribucí založených na Debian jádře a Red Hat jádře, v různých testech a jejich následovné možné uplatnění na základě výsledků výše zmíněných testů.

V dnešní době se v IT prostředí setkáváme se slovem virtualizace velmi často, širší veřejnost si, ale stále neuvědomuje, o co se jedná a má proto velmi často problém s rozhodováním zda se jim v jejich podnikání nebo soukromé sféře vyplatí virtualizace nasazovat, popřípadě jaký typ zvolit aby nejlépe plnil jimi požadovaný účel.

Tato práce se proto bude snažit přiblížit prostředí virtualizací, virtualizačních nástrojů a poskytne výsledky několika testů provedených ve virtualizačním módu, který je veřejnosti nejdostupnější.

Anotation

This bachelor work will be taking a look on virtualization, it will be a specially focused on compering a efficiency of two linux distributions, one based on Debian and a second based on Red Hat, true a list of several test. Based on which we will later decide which of those linux distribution and virtualization methods is suited better for task in question.

Nowadays we hear term virtualization in IT environment all the time. But broad population still doesn't have a clue what the virtualization is and what it's full potential. That very often leads to situation where people can't decide or don't know if virtualization can help them with their busyness or personal endeavors, or are not sure which type of hypervisor and operating systems is best performing given tasks.

This paper will try to get reader more familiar with virtualization. Virtualization software and it will provide results of tests which were performed on the most common type of virtualization mode.

Univerzita Hradec Králové
Faculty of Informatics and Management
Akademický rok: 2016/2017

Studijní program: Systems Engineering and Informatics
Forma: Full-time
Obor/komb.: Informační management (im3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Čulík Daniel	Revoluční 67, Jablonec nad Nisou	I1100543

TÉMA ČESKY:

Zjišťování efektivity HWA virtualizace Linuxových distribucí pod hypervisory XEN a KVM

TÉMA ANGLICKY:

Evaluating efficiency of HWA virtualization of Linux distributions hypervisors Xen and KVM

VEDOUCÍ PRÁCE:

Mgr. Josef Horálek, Ph.D. - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl.
Cílem práce je provést analýzu nástrojů KVM a XEN pro virtualizaci linuxového stroje. Na základě analýzy provést reálné nasazení stroje a jeho zátěžové testování.

Osnova.
Úvod
Úvod do virtualizace - Popis co virtualizace je a jak funguje.
Virtualizace KVM - historie, princip fungování.
Virtualizace XEN - historie, princip fungování.
Popis nasazení vybraného virtualizačního nástroje
Testování a vyhodnocení
Závěr

SEZNAM DOPORUČENÉ LITERATURY:

Podpis studenta: ..... **Datum:** .....

Podpis vedoucího práce: ..... **Datum:** .....

Obsah

Seznam obrázků

Úvod

Virtualizace je jedním z nejvíce skloňovaných IT odvětví poslední doby. Poskytuje řadu výhod, které jsou uvedeny v první kapitole. V této kapitole jsou pak následně rozepsány a přiblíženy. Je zde uvedeno jak konsolidace serverů, (koncentrace mnoha málo využívaných serverů do jednoho konstantně využívaného)která může ušetřit mnoho finančních prostředků, tak jsou zde uvedeny i další výhody z oblasti zálohování dat, bezpečnosti a zjednodušení softwarového testování.

Druhá kapitola nám umožňuje náhled na to, co se děje s jednotlivými fyzickými komponenty hardwaru během interakce s hypervizorem. Jak reaguje RAM, jak se nakládá s využitím procesu, jak virtuální stroje zacházejí se síťovou kartou a jaké různé typy virtuálních pevných disků můžeme na fyzických discích vytvořit.

Třetí kapitola rozplétá a přibližuje jednotlivé typy virtualizace a vyrtualizační principy. Dozvíme se zde, jak jednotlivé typy virtualizací zachází s fyzickým hardwarem a jaký je rozdíl úrovně abstrakce mezi jednotlivými typy. Dále se zde budeme snažit vysvětlit poněkud zmatečnou terminologii, která je způsobena marketingovými tahy jednotlivých společností pracujících na vývoji hypervizorů.

Čtvrtá kapitola se už detailně věnuje jednotlivým virtualizačním nástrojům a jejich hypervisorům. Jsou tu zastoupeni všichni přední hráči virtualizačního průmyslu. Hyper-V od Microsoftu je veřejnosti ne příliš známý, ale v průmyslu se s sním, jde často setkat. VMware a jeho balíček vSphere je zatím nejznámější virtualizční nástroj současnosti. Od komerčního Citrixe a RHEV jsou zde uvedeny jejich freeware verze Xen a KVM.

Pátá kapitola se zabývá hardwarovými zátěžovými testy. Každý test je jasně popsán a je uvedeno, co je jeho podsatatou. Dále jsou specifikovány podmínky, za kterých testy probíhaly. Jsou zde též popsány hardwarové komponenty, na kterých byly testy prováděny. Jsou zde uvedeny i verze operačních systémů a vyrtualizačních nástrojů. Následně je u každého testu i jeho vyhodnocení s přiloženým grafem a tabulkou výsledků.

V závěru jsou uvedeny konečné poznatky a zhodnocení zda byly naplněny cíle práce. Jako další je zde i jisté shrnutí situace na poli virtualizací a doporučení pro jednotlivé uživatele.

1. Úvod do virtualizace

První kapitola je věnována bližšímu seznámení s principy virtualizace a termíny, které jsou nezbytné pro pochopení fungování virtualizace a bez kterých by hlubší porozumění této problematice nebylo možné.

1.1 Virtualizace

Matthew Portnoy ve své knize [1] přibližuje virtualizaci jako abstrakci fyzické komponenty do logického objektu. Virtualizací celých soustav komponent, v našem případě celých počítačů, lze dosáhnout většího zužitkování dostupných zdrojů, a tím dalších energetických a ekonomických výhod, které si uvedeme v dalších částech práce.

První virtualizace na podnikové úrovni byla poprvé využita již v šedesátých letech dvacátého století, a to na mainframech společnosti IBM. Nicméně první, jasně systemizovanou definici, požadavků na počítač podporující virtualizaci, byla sepsána dvojicí Gerald J. Popek a Robert P. Goldberg. V jejich článku ,,*Formální požadavky pro virtualizovatelnost architektur třetí generace*“[1]. V Článku jsou popsány role a vlastnosti virtuálních strojů, stejně tak jako role VMM (Virtual Machne Monitor). Díky takto definovaným požadavkům může dnes VM (Virtual Machine) virtualizovat všechny nezbytné hardwarové prostředky, jako jsou procesory, operační paměti, úložný prostor a síťové prvky.

Dle Popka a Goldberga potřebuje VMM naplňovat tři základní vlastnosti aby správně uspokojil jejich definici. [1]:

- **Věrohodnost**: Prostředí, které Virtual machnine monitor, tzv. hypervizor, vytváří pro fungování virtuálního stroje, musí být přesným obrazem hardwaru fyzického počítače.
- **Izolace**: VMM jakožto kontrolní entita musí mít kompletní kontrolu nad systémem, takže má zabránit tomu, aby si jednotlivé virtuální stroje vzájemně zasahovaly do zdrojů.
- **Výkon**: Rozdíl na výkonu virtuálního a fyzického stroje by měl být minimální nebo by neměl být vůbec.

Podle autorů Kennetha Hesse a Amy Newmanové je formální definice virtualize jednoduchá, je to fyzická abstrakce výpočetních zdrojů [2].

Pro další výklad bude žádoucí uvést některé pojmy týkající se tématu, a to pojmy **HOSTITELSKÝ** a **HOSTOVANÝ** operační systém, dále pak pojmy **VM** a **VMM**.

Hostovaný operační systém:

Hostovaný operační systém využívá pro svůj běh systémové prostředky hostitelského operačního systému. Velmi často je také nazýván jako VM (Virtual Machine) tedy virtuální stroj.

Hostitelský operační systém:

Hostitelský operační systém funguje přímo na fyzickém hardwaru a vytváří prostředí pro hostovaný operační systém.

[1] Dostupné k zakoupení či půjčení na *http://dl.acm.org/citation.cfm?doid=361011.361073*

VMM:

VMM (Virtual Machne Monitor) v dnešní době také nazýván jako Hipervisor je speciální software který zprostředkovává Hostovanému operačnímu systém přístup ke zdrojům fyzického stroje.

VM:

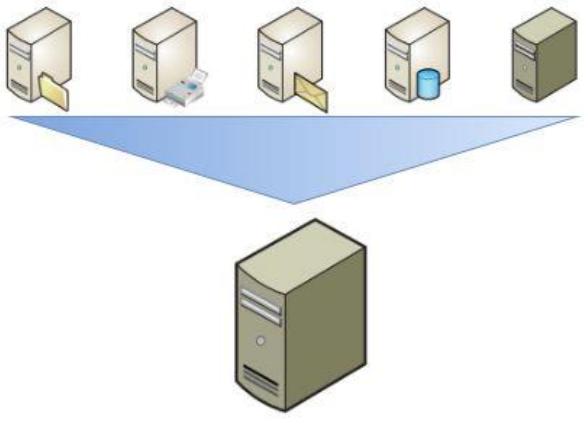
VM (Virtual Machnine) Virtuální stroj je operační systém nebo aplikační prostředí které je instalováno na softwaru který imituje hardware, tedy VMM.

1.2 Přínosy virtualizace

1.2.1 Konsolidace

Konsolidace serverů je jedním z největších, ne-li největší výhodou virtualizace. Virtualizace umožňuje vytvořit větší množství virtuálních strojů na jednom fyzickém stroji, reálný počet závisí na výkonosti daného hardwaru. Můžeme tak redukovat velké množství redundantního hardwaru. Jedním z nejlepších kandidátů na virtualizaci jsou podle K. Hesse [2] web servery, mail servery a jiné síťové služby (DNS, DHCP, NTP). Tyto služby většinu času svého provozu plně nevytěžují zdroje, které jim fyzický hardware nabízí, podle M. Portnoye procesory nevykonávají žádnou činnost po 95% svého času [1]. Virtualizovnané stroje nebudou nutně efektivnější, ale budou vyžadovat menší množství elektrické energie nebo prostředků používaných na jejich chlazení. Výhody, které konsolidací získáme, tedy jsou:

- Efektivnější využívání hardwaru.
- Nižší náklady na chlazení serverů.
- Nižší spotřeba elektrické energie
- Nižší finanční náklady na technickou podporu serverů
- Přehlednější správa serverů.



Obrázek 1 - Konsolidace serverů [1]

1.2.2 Zálohování a přístupnost

Podle H. Kennetha a A. Newman [2] je virtualizace nejméně ekonomicky a časově náročná na tzv. MTTR (Mean Time To Recovery – Nejkratší čas na obnovu) v případech katastrofálního selhání. Po dlouhá léta platilo, že si systémoví administrátoři mohli vybrat dvě volby z tří možných: RYCHLOST, NÍZKÁ CENA, SPOLEHLIVOST. Díky virtualizaci je toto pravidlo již zapomenuté [2]. Virtuální stroje jsou totiž zapouzdřené systémy, v podstatě jen svazek souborů, který lze jednoduše zálohovat či různě přesouvat [1]. Obnova je proto stejně rychlá jako běžné spouštění systému. Může nastat situace, kdy bude nutné obnovit nějaká data ze záloh, virtuální stroje však zůstanou po celou dobu operace funkční a server tak nepotřebuje odstávku z důvodu hardwarové poruchy nebo reinstalace stěžejního softwaru. Stěžejní položkou v problematice zálohování jsou tzv. **snapshoty**. Jedná se o zachycení stavu virtuálního stroje ve specifickém bodě v čase. Jsou to záchytné body, ke kterým se můžete jednoduše a kdykoli vrátit v případě, že si přejete upravit nebo odstranit vámi provedené změny v systému nebo jeho konfiguraci[1]. Díky všem těmto prvkům je možné přesouvat Virtuální stroje bez nutné odstávky nebo zálohovat servery z jiných geografických lokalit v případě přírodních katastrof.

1.2.3 Softwarové testování

Využití virtuálních strojů pro testování byla jedna z prvních aplikací virtualizace x86 architektury. Virtuální stroj je vytvořen, následně opatřen Operačním systémem, nakonfigurován a případně dovybaven nutným softwarem a patchem. Dále se vytvoří snapshot takového systému, ten může být dále použit k instalaci, uninstalaci a modifikaci softwarových balíčků. Použitím virtualizovaného stroje tak dostáváme šanci předejít různým konfliktům s rozdílnými konfiguracemi nebo kolizím různých programů při produkčním nasazení serveru. Pokud by došlo během testování k nějakým problémům, je možné se rozhodnou zda je chceme řešit nebo zda si jen obnovíme snapshot stabilního systému. Tento typ softwarového testování nám dává možnost důsledně a opakovaně provádět riskantní testy bez nutnosti reinstalace systému a následné instalace aplikací a patchů [2].

1.2.4 Centralizovaná správa

Veškeré hlavní virtualizační produkty mají konzoli nebo centralizované uživatelské rozhraní, které umožňuje prohlížení a zprávu všech virtuálních strojů spadající pod toto rozhraní. Toto centralizované řídící rozhraní je přijatelné řešení pro všechny správce serverů. Odpadá tak nutnost separátně připojovat jednotlivé prvky interface pro jednotlivé stroje. Toto řídící rozhraní dává dále systémovému administrátorovi nástroj, který umožňuje zprávu jakéhokoli systému, nehledě na operační systém který na daném virtuálním stroji funguje. Dále pak můžeme vytvářet jednotlivé skupiny virtuálních strojů a skupiny uživatelů s různými úrovněmi přístupu k těmto strojům [2]

1.2.5 Spotřeba

Spotřeba elektrické energie je velmi oblíbené téma každé debaty která se týká virtualizací na servere nebo blade serverech. Virtuální stroje spotřebovávají elektrickou energii, stejně tak jako užívají jiných zdrojů fyzického stroje jako je operační paměť, procesor, místo na disku nebo síťový provoz. Virtuální stroj jak už bylo řečeno je svazek souborů, můžeme o něm tedy přemýšlet, jako o jakékoli jiné aplikaci která na systému běží. Nevytížený systém má spotřebu energie nižší, nežli vytížený. Nicméně několik plně vytížených systémů s větším počtem virtuálních strojů nemá celkový odběr elektrické energie a požadavků na chlazení tak velký jako stejný počet fyzických systémů, které nejsou plně vytížené [2].

1.3 Vztahy operačního systému s architekturou CPU

Podle, Rogiera Dittnera a Davida Rule [4] jsou ideální hardwarové architektury ty, v kterých je operační systém a CPU navržen a postaven jeden pro druhého, a jsou velmi úzce spárováni. Jedním takovým případem byl MULTICS, co dělalo MULTICS. To v jeho době speciálním byl jeho přístup rozdělování softwarových operací, aby eliminoval riziko nebo možnost ohrožení nebo destabilizace, poruchovými komponentami, od toho aby ovlivnily ostatní fungující komponenty. Zavedl tak formální mechanismus nazývaný **ochranné kruhy**. Ten umožňuje oddělovat důvěryhodné operace Operačního systému od nedůvěryhodných operací uživatelských programů. MULTICS zavedl osm těchto ochranných kruhů, umožňoval tak různé úrovně isolace a abstrakce od jádra ničím nevázaných interakcí s hardwarem. Bohužel tento design se ukázal jako příliš finančně nákladný a uzavřený pro praktické tržní nasazení [4].

Dnes nejvíce využívaná architektura v moderních počítačích je IA-32, nebo x86. Začínajíce s chipsetem 80286, nám rodina x86 představila dvě hlavní metody adresování paměti:

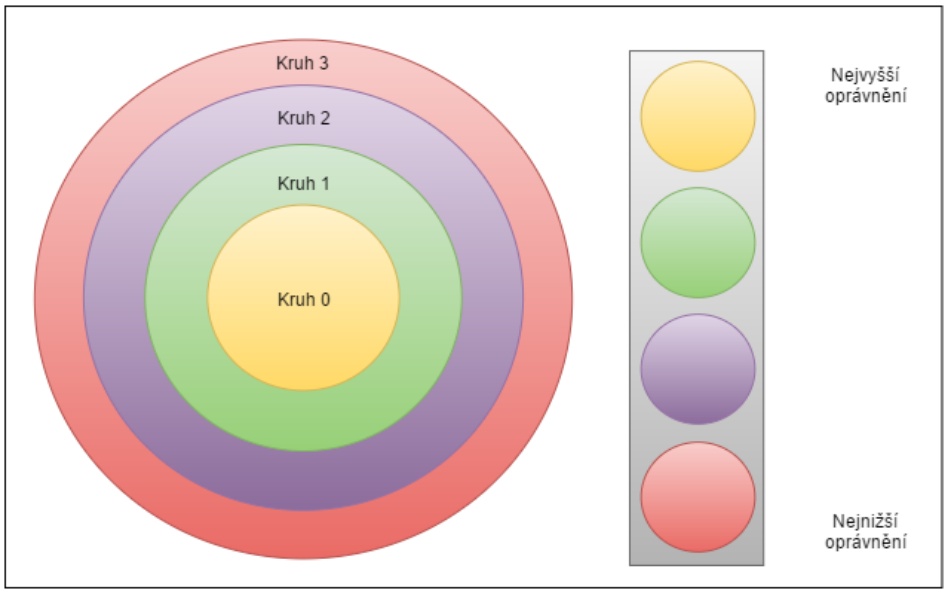
- Reálný režim
- Chráněný režim

Po uvedení chipsetu 80368 byla přidána další metoda:

- Virtual 8086 nebo VM86

Reálný mód, který je limitován na pouhý jeden megabyte paměti se stal velmi rychle zastaralým. Virtuální mód jehož strop byl nastaven na 16 MiB a s příchodem 32-bitových operačních systému široce dostupných pro x86 architektury se též stal velmi rychle zastaralým. Záchranou pro x86 se tedy stal chráněný režim, který byl dovybaven několika novými vylepšeními pro podporu multitaskingu a které obsahovaly členění procesů tak, aby jim již nebylo umožněno zapisovat mimo jim vyhraněný adresní prostor, dále pak tyto výdobytky obsahovaly hardwarovou podporu pro virtuální paměť a umožňovaly přepínání mezi jednotlivými úlohami [4].

V architektuře x86 používá chráněný režim čtyř privilegovaných úrovní, nebo kruhů, číslovaných od 0 do 3. Systémová paměť je rozdělena do segmentů, a každý segment je přiřazen jednomu kruhu. Procesor používá privilegované úrovně k tomu, aby určil co může a nemůže být provedeno s kódem nebo daty v daném segmentu. Pojem kruhů je převzatý ze systému MULTICS, kde byl kruh 0 považován za kruh s největší kontrolou procesoru [4]. Viz obr.2.



Obrázek 2 - Stupně oprávnění architektury [4]

Moderní operační systémy poupravily tento model a zredukovaly počet kruhů ze čtyř na dva:

- Kruh 0, je znám jako režim jádra (ang. Kernel mode). V tomto režimu může procesor provést jakoukoli operaci, kterou architektura umožňuje. Má tedy neomezený přístup k systému.

- Kruh 3, je znám jako uživatelský režim (ang. User mode). Tento režim nemá schopnost přímého přístupu k hardware, ale může užívat tzv. systémových volání. Většina procesu v systému je vykonána v tomto režimu.

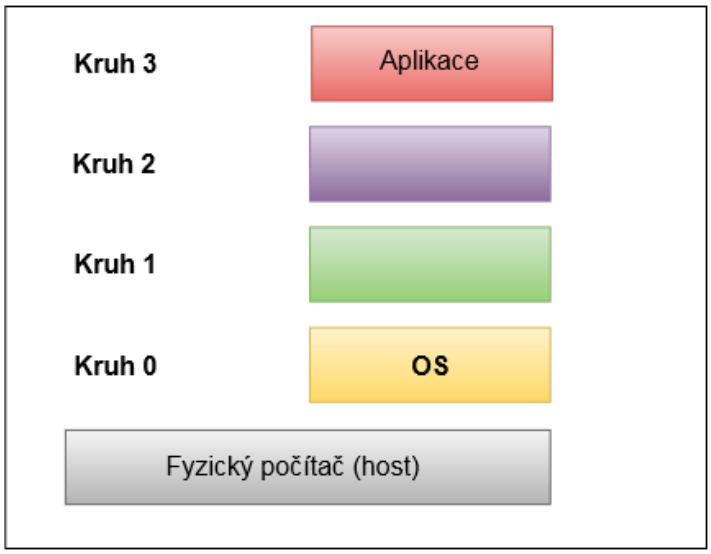
Bezpečnostní mechanizmy zajišťují, že instrukce procesoru, které přímo pracují s hardwarem (tzv. privilegované) jsou prováděny pouze operačním systémem v režimu jádra. Pokud se program běžící v kruhu 3 pokusí adresovat prostor mimo svůj segment, dojde k hardwarovému přerušení, dále můžeme proces ukončit, nebo nasimulovat provedení operace a pokračovat dál.

1.3.1 Komplikace s virtualizací architektury x86

X86 operační systémy jsou navrženy tak aby fungovaly přímo na „Železe“ hardwaru a tak přirozeně předpokládají, že plně vlastní veškerý hardware na daném stroji. Jak jsme již zmiňovali x86 nabízí čtyři privilegované úrovně známé jako kruh 0 až kruh 3, s tím že v kruhu 0 se provádí veškeré privilegované instrukce, které pracují přímo s hardwarem, a v kruhu 3 pak běží veškeré uživatelské procesy. Virtualizace architektury x86 vyžaduje umístění virtualizační vrstvy pod operační systém (u kterého se předpokládá, že je umístěn v kruhu 0) která vytvoří a řídí virtuální stroje. Komplikující situací dále představují citlivé instrukce, které nemohou být efektivně virtualizovány, jelikož mají odlišnou sémantiku pokud nejsou vykonány v kruhu 0 [5].

Virtualizace tyto specifické vlastnosti operačních systémů x86 obchází různými způsoby, [5] jsou to:

- Plná virtualizace za pomoci binárního překladu
- Operačním systémem asistovaná virtualizace nebo paravirtualizace
- Hardwarově asistovaná virtualizace (první generace)



Obrázek 3 - Architektura x86 bez virtualizace [5]

1.3.2 64. bitové rozšíření x86 (x86-64)

Podle autorů [6] je rozšíření x86 architektury o 64 bitové adresování jen otázkou přidání nového režimu který se nazývá **Long mode**, avšak i nadále architektura podporuje předešlé režimy, které jsou součástí skupiny, která se označuje jako **Legaci mode**.

Long mode se skládá ze dvou pod-režimů:

- 64-bit režim – který má schopnost adresovat 64-bitový virtuální adresní prostor.
- Režim kompatibility – ten zajišťuje binární kompatibilitu s již existujícími 16 a 32 bitovými aplikacemi pokud běží na 64 bitovém operačním systému.

1.4 Hypervizor

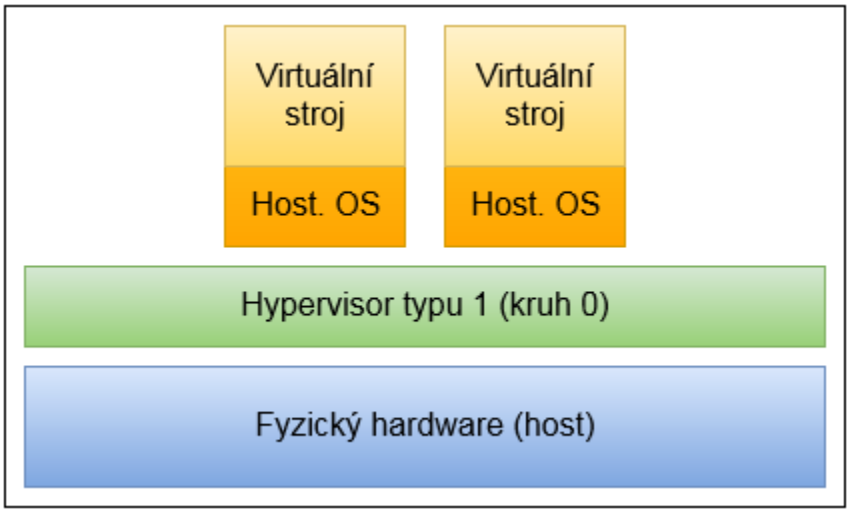
Podle M. Portnoye [1] je hypevizor (v dřívějších dobách označován jako VMM) definován jako softwarová vrstva která spočívá mezi virtuálním strojem a hardwarem fyzického počítače. Bez hypervizoru, komunikuje operační systém přímo s hardwarem který je pod ním. Operace disku jdou přímo do subsystému disku a strojová volání jsou brána přímo z fyzické paměti. Bez hyperivisoru by více jak jeden operační systém z více virtuálních strojů chtělo přistoupit k hardwarovým zdrojům jako je například procesor, naráz, což by vyústilo v absolutní chaos. Hypervizor řídí veškeré interakce mezi všemi virtuálními stroji a hardwarem, který všechny hostované virtuální stroje sdílí.

Dle autorů H. Kennetha a A. Newman [2] je definice Hypervizoru lehce odlišná. Hypervizor je software který běží přímo na „železe“ hostujícího stroje, nejlepší cesta jak popsat technologii hypervizoru je udělat si porovnání mezi hypervizorem a hostovaným virtuálním strojem. Na první pohled se hypervizor jeví podobně jako virtuální stroj, přitom jsou však velmi rozdílné.

Dále je dle [2] Hypervizor virtualizační software který funguje na operačním systému. Naopak, hostovaná virtualizace využívá operační systém a spouští virtualizační software jako aplikaci. Hypervizor je instalován přímo na hardware stroje a následně je instalován operační systém, což je sám o sobě paravirtualizovaný virtuální stroj. Hostovaný operační systém má pak označení jako Virtuální stoj 0 nebo také Dom0.

1.4.1 Hypervizor typ 1

Tento typ funguje přímo na „železe“ fyzického stroje, bez operačního systému, který by stál mezi ním a hardwarem. Protože tedy není přítomna žádná rušivá mezivrstva mezi hypervizorem a hardwarem je tento typ hypervizoru označován jako tzv. **bare-metal** implementace. Bez jakéhokoli prostředníka může hypervizor typu 1 přímo komunikovat s hardwarovými zdroji které jsou součástí fyzického stroje na kterém hypervizor typu 1 funguje. Tím je mnohem efektivnější než hypervizor typu 2. [4] Díky tomu že hypervizor typu 1 funguje přímo na hostujícím hardwaru, je tak na skutečném kruhu 0 a hostované operační systémy potom běží na vyšších kruzích a dovolují tak skutečnou izolaci jednotlivých virtuálních strojů.



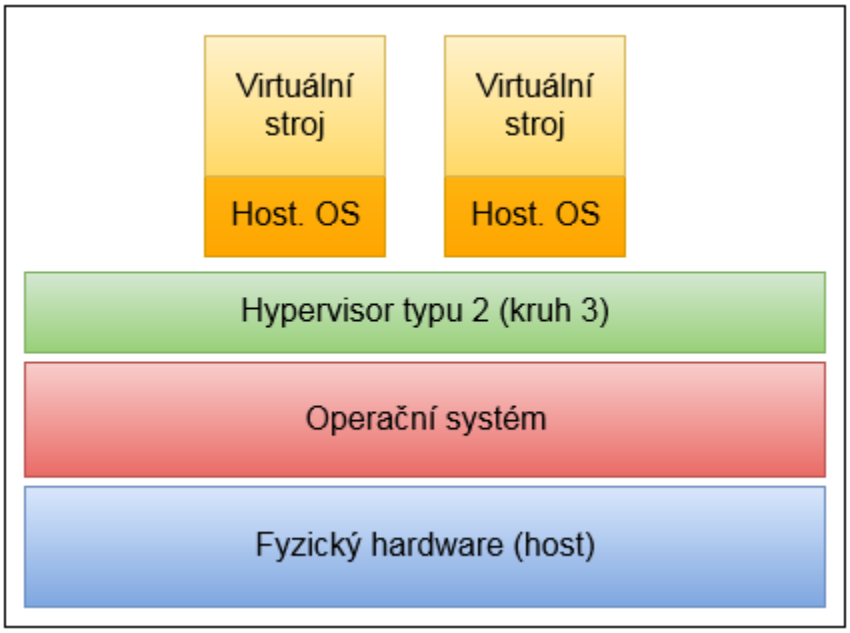
Obrázek 4 - Schéma virtualizace typu 1 [1]

1.4.2 Hypervizor typ 2

Hypervizor typu 2 je [1] definován takto. Je to aplikace, která funguje v rámci tradičního operačního systému, který je instalován na fyzickém hardwaru. Je to tedy proces, který funguje většinou v kruhu 3. První vyrtualizace architektury x86 byly plněny pomocí hypervizorů typu 2, protože to byla nejrychlejší cesta jak dostat virtualizace na trh. Tradiční operační systém, který běžel na fyzickém hardware, řešil veškeré záležitosti týkající se hardwarových zdrojů a hypervizor této schopnosti využíval. Jednou z velkých výhod kterou hypervizor typu 2 má je, že podporuje velký rozsah dostupného hardwaru, tím že může převzít jeho řízení od operačního systému, který virtuální stroj používá. Dále jsou pak hypervizory typu 2 mnohem jednoduší na instalaci a nasazení do produkce, jelikož většina hardwaru, což jsou různá úložiště nebo síťové prvky, už fungují díky tomu, že ovladače jsou přebrány z operačního systému.

Hypervizory typu 2 nejsou tak výkonné jako hypervizory typu 1 a to z důvodu, již [1] zmíněné mezivrstvy operačního systému, která je mezi hypervizorem a samotným hardwarem fyzického počítače. Pokaždé když virtuální stroj provede jakoukoli operaci týkající se hardwaru, jako je čtení z disku nebo posílání dat skrze síťový interface, musí předat tento požadavek hypervizoru stejně tak jako je tomu u hypervizoru typu 1. Rozdíl je v tom, že hypervizor typu 2 musí tento požadavek nadále předat hostujícímu operačnímu systému, který řídí veškeré I/O operace.

Další nevýhoda je v menší spolehlivosti, což je zapříčiněno tím že je více míst v kterých může hypervizor selhat. Dále pak to že vše co negativně ovlivní hostující operační systém na kterém hypervizor běží, může ovlivnit stabilitu nebo činnost virtuálních strojů, které v rámci tohoto hypervizoru fungují. Jako příklad mohou být uvedeny systémové aktualizace, které často vyžadují restart systému. Pokud by k aktualizacím došlo na hostujícím operačním systému, restartovaly by se společně s ním i všechny hypervizorem hostované virtuální stroje.



Obrázek 5 - Schéma virtualizace typu 2 [1]

2. Hardware

V této kapitole krátce představíme, jednotlivé fyzické komponenty hardwaru, které jsou používány během virtualizace. Dále se dozvíme informace na co hledět a jak řešit případné problémy s výkonem u systému které jsou primárně určené k virtualizování. Tyto informace nám usnadní pochopené toho že virtuální stroje jsou jenom svazek souborů [7].

2.1 Procesor

Virtuální stroje nasazují virtuální procesor, který je identický s procesorem hostujícího fyzického stroje, a který dociluje virtualizace tím, že nechává projít ne-privilegované instrukce (kruh 3) přímo na fyzický procesor. Privilegované instrukce jsou bezpečně zpracovány skrze hypervizor. Tím, že hypervizor umožní vykonávat veškeré procesy uživatelské úrovně přímo, umožňuje virtualizovanému stroji pracovat rychlostí která se blíží rychlosti hostujícího procesoru. Každý hostovaný virtuální stroj bude mít svůj vlastní virtuální procesor, který je izolovaný od ostatních virtuálních strojů. Navíc každý procesor virtuálního stroje, má svoje vlastní registry, buffer a kontrolní strukturu [7].

2.2 Operační paměť (RAM)

Stejně jako u virtualizace procesoru i u operační paměti, jsou přístupy k ní řízeny skrze hypervizor nebo skrze virtualizační vrstvu. Hypervizor je zodpovědný za zprostředkování přístupu k přiléhající operační paměti pro virtuální stroj. Hostující stroj na oplátku mapuje operační paměť zpět k fyzickému zdroji. Řízení disponibilních zdrojů operační paměti je zcela transparentní jak pro hostovaný virtuální stroj, tak pro subsystémy tohoto virtuálního stroje, které spravují operační paměť [7].

Z pohledu výkonosti a možné rozšiřitelnosti je důležité si položit otázku, jak virtuální stroj nakládá se stránkovací a nestránkovací pamětí. Oba dva typy paměti jsou vytvořeny během startu systému. Nestránkovací paměť se skládá z rozsahu virtuálních adres, které zůstávají v RAM po celou dobu běhu systému. Stránkovací paměť může být prohozena do pomalejších systémových komponent, jako je například pevný disk. Možnost použít stránkování nám dovoluje vytvořit a spouštět, větší číslo virtuálních strojů na daném fyzickém systému. Toto řešení prohazování paměti z pevného disku do RAM, má ovšem velmi negativní dopad na výkonost[7].

2.3 Pevný disk

Na rozdíl od RAM je pevný disk považován za hlavní permanentní paměť pro virtuální stroje. Během instalace tradičního operačního systému, se provede mapování bloků pevného disku do souborového systému a připraví se disk pro uchovávání dat. Souborový systém je řešení jakým jsou soubory na pevném disku organizovány, ukládány a pojmenovávány. Úložiště souborů je reprezentováno na plotnách pevného disku jako magnetický vzor nesousedících bloků. Pokud by bloky byly sousedící, nebo propojené odkazy do jednoho uskupení, potom by sekvenční bloky dat mohly reprezentovat jeden vnořený soubor v jiném větším souborovém systému. Virtualizační vrstva konstruuje virtuální pevný disk výše uvedeným způsobem, zapouzdřením virtuálního disku do jednoho souboru, který je pak uložen na hostujícím pevném disku. Toto je striktně virtuální pevný disk a je to tak abstrakce pevného disku [7].

Virtuální disky jsou tak robustní a mobilní proto, že abstraktní proces, který zapouzdřuje disk, z něj zároveň vytvoří soubor. Přesouvání disku virtuálního stroje je jednoduché jako přesouvání jakéhokoli souboru dat. Virtuální disk ať už je to SCSI (Small Computer System Interface) nebo IDE (Integrated Drive Electronics) je hostujícímu operačnímu systému představen jako typický diskový řadič a rozhraní. Vývojáři virtualizačního softwaru užívají jen malé množství ovladačů pro hostované virtuální stroje když vytváří abstraktní pevný disk. Tím, že používají malé množství prozkoušených a testovaných ovladačů, nemají virtuální stroje tolik problémů jako ovladače tradičních operačních systémů [7].

Disky virtuálních strojů přicházejí v různých podobách jako je fyzický, prostý, dynamický a virtuální. Každý z těchto disků má svá pozitiva a negativa. Nyní si představíme a krátce popíšeme některé z nich.

2.3.1 Virtuální disk

Nejjednodušší způsob podle autorů [7] jak si definovat co virtuální disk je, je nejlepší si nejdříve říci co není. Virtuální disk **není** disk fyzický. Je to všeobecný pojem, který popisuje veškeré typy virtuálních disků a jejich funkce, které jsou používány virtuálními stroji. Platí, že virtuální disk se skládá ze souboru nebo svazku souborů a virtuálnímu stroji se zobrazuje jako fyzická jednotka.

2.3.2 Dynamický disk

Během vytváření virtuálního disku, je nezbytné určit maximální velikost tohoto pevného disku. Tím, že nepřidělíme nově vznikajícímu virtuálnímu disku celý diskový prostor hned na začátku, může vytvořit takzvaný dynamický disk. Dynamický disk je hned po vytvoření velmi malý (zpravidla obsahuje pouze hostovaný operační systém) a roste podle toho jak se do hostovaného operačního systému ukládají data. Výhody užívání dynamického disku jsou v tom, že spotřebovávají daleko méně místa na disku hostujícího stroje a systém se dále jednodušeji zálohuje. Nevýhodou možnosti dynamicky měnit velikost virtuálního disku je úbytek výkonnosti virtuálního stroje, který ho používá [7].

2.3.3 Pre-alokovaný disk

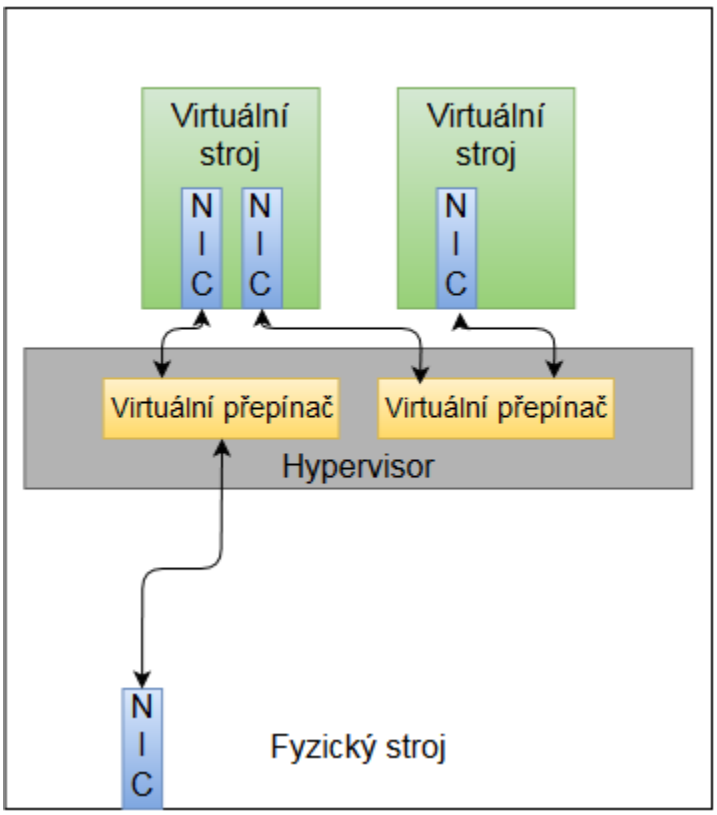
Pre-alokovaný disk má určenou velikost již na začátku jeho tvorby. Pre-alokované disky jsou stále virtuálními disky, zmapují však veškerý prostor, který jim byl hypervizorem určen. Výhoda používání pre-alokováného disku je v tom, že jejich velikost je neměnná a zůstane v rámci, který byl na začátku stanoven. Další výhodou je, že na rozdíl od dynamických disků, výkon virtuálního stroje není tak moc postižen, bohužel fragmentaci disku se ani v tomto případě nevyhneme [7].

2.3.4 Návratový disk

Návrat v tomto případě označuje možnost vrátit stav disku do předešlého stavu. U disků virtuálních strojů to znamená, že hostující operační systém nezapisuje provedené změny ihned na hostův virtuální disk. Veškeré změny, které jsou provedeny během pracovní seance, jsou namísto toho zapsány do dočasného souboru. V momentě kdy je pracovní seance ukončena a virtuální stroj se vypne, jste hypervizorem tázáni, zda se provedené změny mají uložit nebo zrušit. Pokud chcete změny uložit musíte je opětovně potvrdit. Tím dočasné soubory promítnou změnu do originálního obrazu disku. Pokud změny nepotvrdíte je dočasný soubor okamžitě zrušen [7].

2.4 Síťové zdroje

Stejně jako u jejich fyzických protějšků, virtuální síťové zdroje pro virtuální stroje představují cestu jak komunikovat s fyzickým světem. Každý virtuální stroj může být nastaven tak, aby měl jednu nebo více síťových karet, též známých jako NIC (Network Interface Card) které reprezentují připojení k síti. Virtuální síťové karty se však nepřipojují napřímo k síťovým kartám hostitele. Hypervizor podporuje řešení při kterém je vytvořena virtuální síť, která následně propojuje virtuální síťové karty se sítí která je složená z virtuálních přepínačů. Je to právě tato virtuální síť ke které se připojují fyzické síťové karty [1].



Obrázek 6 - Schéma virtuální sítě [1]

Výše uvedená virtuální síť je dále nezbytná z bezpečnostního hlediska, protože vytváří bezpečné prostředí pro virtuální stroje, které sdílejí hosta. Další bezpečnostní výhoda je v tom, že veškerá komunikace, která probíhá mezi virtuálními hosty je směrována skrze přepínače a nikdy tak neopustí fyzického hosta. Pokud je síťová karta druhého virtuálního stroje připojená k přepínači a ten přepínač není spojen s fyzickou síťovou kartou, zůstává pouze jeden způsob jak může druhý virtuální stroj komunikovat s vnější sítí a to je skrze první virtuální stroj, který je připojen do obou přepínačů tak jak je vyobrazeno na obr.6. První virtuální stroj nám tak vlastně vytváří ochrannou vrstvu pro druhý virtuální stroj [1].

3. Typy virtualizace

V této kapitole se budeme věnovat popisu nejpoužívanější a nejdůležitější techniky hardwarové (též označované jako serverové) virtualizace, které se při virtualizaci architektury x86 používají.

3.1 Plná softwarová virtualizace (Hardwarová emulace)

Tato technika virtualizace vytváří vytrualizační vrstvu, která plně simuluje hardware na kterém je následně nainstalován operační systém. Virtualizace hardwaru je kompletní včetně systému BIOS. Operační systém, který je nainstalován si není vědom toho že by byl vyrtualizován, nemusí být proto nijak zvláštně modifikován. Tím plná softwarová virtualizace neklade žádná omezení pro výběr operačního systému, který bude na virtuálním stroji fungovat[4][5].

Výhody:

- Není vyžadována žádná úprava na hardwaru hostujícího stroje nebo na softwaru operačního systému.
- Poskytuje úplnou izolaci jednotlivých virtuálních strojů. Na každém virtuálním stroji tak mohou fungovat různé operační systémy.
- Plná softwarová virtualizace je jediná která nepotřebuje žádnou hardwarovou nebo softwarovou úpravu pro plnění privilegovaných instrukcí.
- Zjednodušená přenositelnost virtuálních strojů, které je docíleno díky striktnímu oddělení virtualizovaného hardwaru od fyzického.

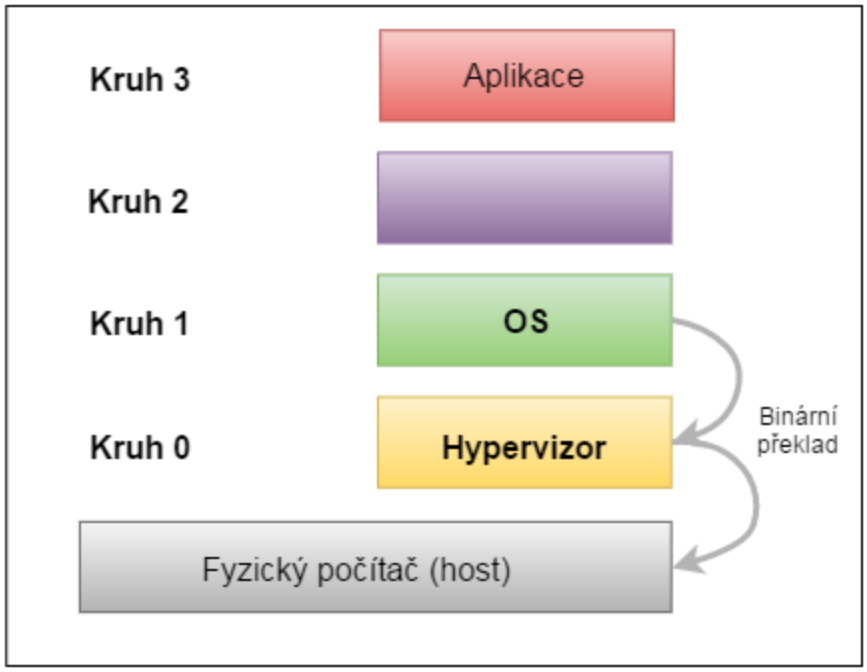
Nevýhody:

- Nízká výkonost během vyřizování požadavků privilegovaných instrukcí virtuálního stroje s přímým přístupem k fyzickému hardwaru. Tato ztráta výkonu je zapříčiněna nutností zachytávání a emulování privilegovaných funkcí.

Autoři a developeři společnosti WMware [5] používají pro řešení problému spojených s emulací privilegovaných instrukcí metodu binárního překladu.

3.1.1 Binární překlad

Binární překlad v kombinaci s technikou přímého přístupu, pomáhá virtualizovat jakýkoli operační systém architektury x86. Metoda binárního překladu zachytává a překládá kód operačního systému aby mohla vyměnit instrukce, které nelze virtualizovat a nahradila je sekvencí instrukcí, které mají požadovaný efekt na virtuální hardware. Mezitím neprivilegované příkazy, které jsou zpravidla vytvářeny uživatelskými procesy v kruhu 3, jsou vykonávány přímo na procesoru, aby se mohla zachovat maximální výkonnost výpočetního výkonu. Operační systém hostovaného virtuálního stroje je přesunut do Kruhu 1, tím má stále vyšší privilegia než uživatelské procesy, ale zároveň má nižší privilegia než hypervizor který zůstává v kruhu 0 [5]. Tak jak je tomu zobrazeno na obr. č. 7.



Obrázek 7 - Metoda binárního překladu u plné softwarové virtualizace (Hardwarové emulace) [5]

3.2 Paravirtualizace

Paravirtualizace je alternativní přístup k virtualizaci. Namísto snahy emulovat hardware x86 architektury, skrze softwarové virtualizační prostředí, paravirtualizační hypervizor koordinuje systémová volání k fyzickým hardwarovým zdrojům. Místo toho, aby se emuloval hardware, paravirtualizační přístup instaluje hostovaný operační systém, který se často označuje jako „DomainU“, přímo na hypervizor. Hypervizor ovšem neobsahuje žádné ovladače pro síťové rozhraní a jiná zařízení, tento problém však řeší skrze instalaci tzv. privilegovaného hosta, který se též označuje jako „Domain0“. Výše uvedený privilegovaný host funguje jako normální virtuální stroj, ale na rozdíl od normálního virtuálního stroje, má oprávnění používat přímá systémová volání k fyzickému hardwaru, který funguje pod ním. Kdykoli pak ostatní neprivilegované virtuální stroje chtějí přistoupit k těmto hardwarovým zdrojům, použijí právě privilegovaného hosta [8].

Proto, aby veškerý výše uvedený popis fungoval, je pro paravirtualizaci nutné, aby jádro operačního systému, který bude na virtuální stroj nainstalovaný bylo upraveno. Tato úprava jádra způsobuje, že veškeré nevirtualizovatelné instrukce jsou nahrazený tak zvanými „hypercally“, které komunikují přímo s virtualizační vrstvou [5].

Výhody:

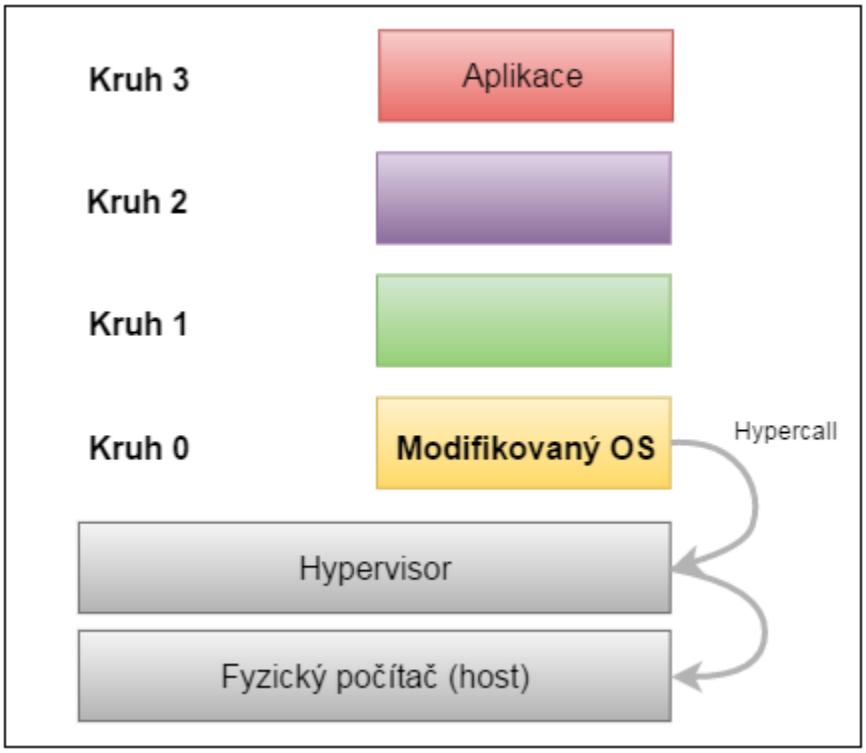
- Pokud není dostupná hardwarově asistovaná virtualizace, je paravirtualizace mnohem efektivnějším řešením než úplná virtualizce [4].

- Vyřizování privilegovaných instrukcí je mnohem méně výkonnostně náročné než v případě hardwarové Emulace [8].

Nevýhody:

- Nutnost modifikace jádra hostovaných operačních systémů, a z toho vyplývající nemožnost instalace proprietárních operačních systémů jako je např. Windows.

- Další nevýhodou vyplývající z modifikace jádra, je snížená přenositelnost a kompatibilita paravirtualizovaných virtuálních strojů.



Obrázek 8 - Technika paravirtualizace [5]

3.3 Hardwarově asistovaná virtualizace

Oba hlavní hráči na poli výroby procesorů, dnes již běžně dodávají procesory pro architekturu x86-64, které umožňují hypervizorům využít možnost hardwarově asistované virtualizace. Tato technologie se nazývá různě u obou výrobců, účel je ovšem totožný. U AMD se jedná o **AMD-V,** u společnosti Intel se jedná o **Intel VT**.

Podle Niela Smythe [9] je princip fungování u obou procesorů stejný. A to, že procesor je vybaven hardwarovým rozšířením, toto rozšíření umožňuje běh nemodifikovaných hostů bez toho aniž by hypervisor měl absolutní přehled, jako je tomu u plné virtualizace (hardwarové emulace). Jednoduše řečeno všechny novější procesory mohou zprostředkovávat dodatečný privilegovaný mód, který je

umístěn nad kruhem 0, v kterém může hypervizor operovat, a tím nechává kruh 0 přístupný pro nemodifikované hostované operační systémy.

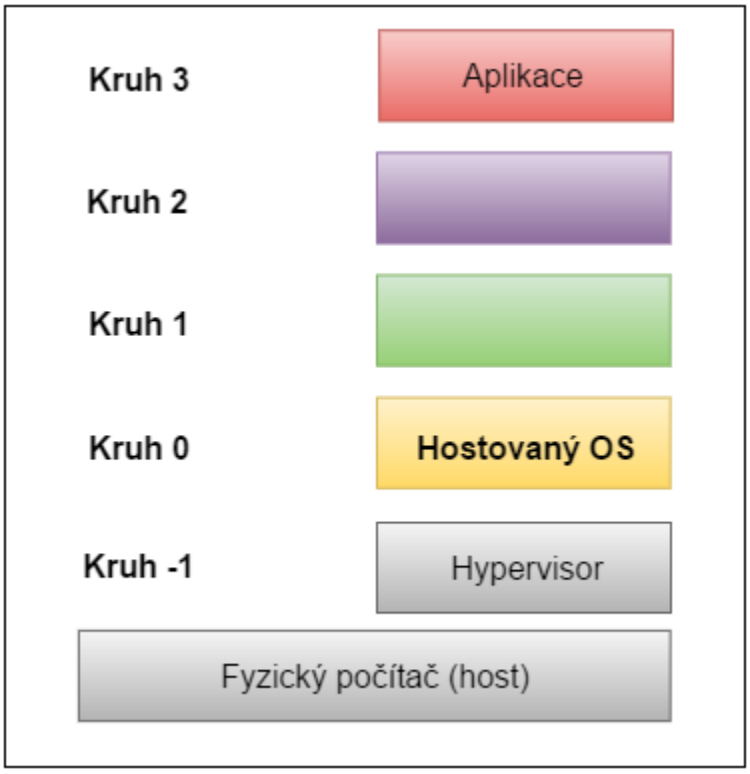
Ovšem autoři [5] společnosti WMware ve své dokumentaci uvádějí, že nový privilegovaný mód není umístěn nad kruhem 0, ale naopak pod ním. Tento kruh -1 (označován též jako root mode) běží jako hypervizor, a hostovaný systém běží v kruhu 0. Hostovaný operační systém poskytuje služby jádra a vykonává další citlivá volání, tak jak je uvedeno na obr. 9.

Výhody:

- Privilegované instrukce jsou vyřizovány na úrovni hardwaru namísto toho, aby byly ošetřovány softwarově jako při plné virtualizaci.
- Není nutná modifikace hostovaného operačního systému.
- S kompatibilním hypervizorem je možné použít paravirtualizační ovladače pro I/O zařízení.

Nevýhody:

- Vyžaduje hardwarovou podporu procesoru.



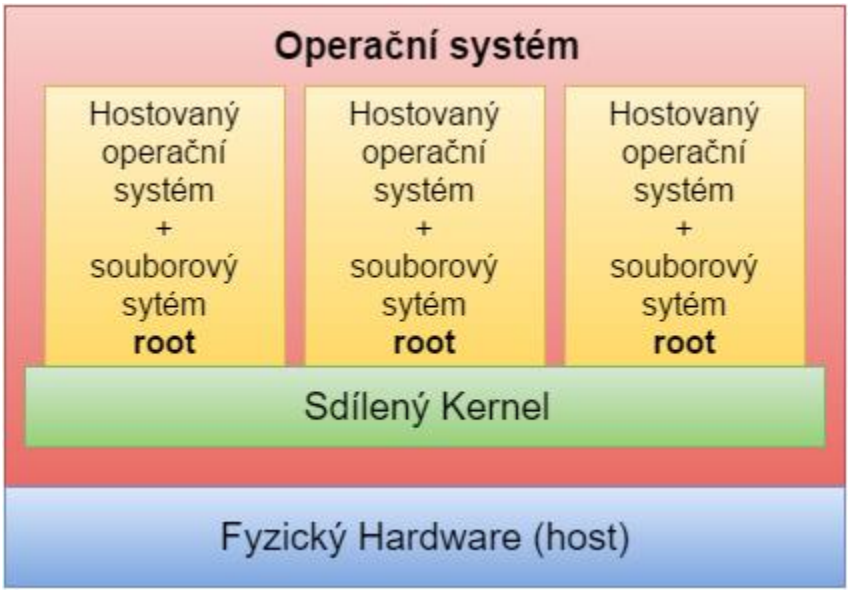
Obrázek 9 - Hardwarově asistovaná virtualizace

3.4 Kontejnerové virtualizace

Tato skupina virtualizačních technik, se od výše zmíněných serverových virtualizací liší v tom, že pro virtualizaci hostovaných systémů nepoužívá hypervisor.

3.4.1 Virtualizace sdílením Kernelu

Virtualizace sdílením Kernelu též známá jako virtualizace na úrovni operačního systému, využívá designu architektury systémů založených na Linuxu a Unixu. Abychom lépe porozuměli tomu, jak virtualizace sdílením Kernelu funguje, musíme nejdříve zmínit dvě hlavní komponenty Linuxových a Unixových systémů. V samotném jádru operačního systému se nachází **Kernel**. **Kernel** je jednoduše řečeno, komponenta, která má na starosti veškeré operace mezi operačním systémem a fyzickým hardwarem. Druhá klíčová komponenta, je souborový systém **root,** který obsahuje veškeré knihovny, soubory a utility nezbytné k tomu, aby mohl operační systém fungovat. V průběhu virtualizace sdílením Kernelu, dochází k tomu, že každý virtuální host má svůj vlastní souborový systém **root**, ale zároveň mají všichni tito hosti stejný Kernel, který sdílí s hostujícím operačním systémem [9].



Obrázek 10 - Zobrazení virtualizace sdílením Kernelu [9]

Tento typ virtualizace je v zásadě možný díky vlastnosti Kernelu dynamicky měnit aktuální souborový systém **root** (koncept také známý jako chroot) za jiný souborový systém **root** aniž by se musel restartovat celý systém [9].

Výhody:

- Vysoká výkonost virtuálních hostů.
- Vysoká flexibilita nasazování a odebírání virtuálních hostů.

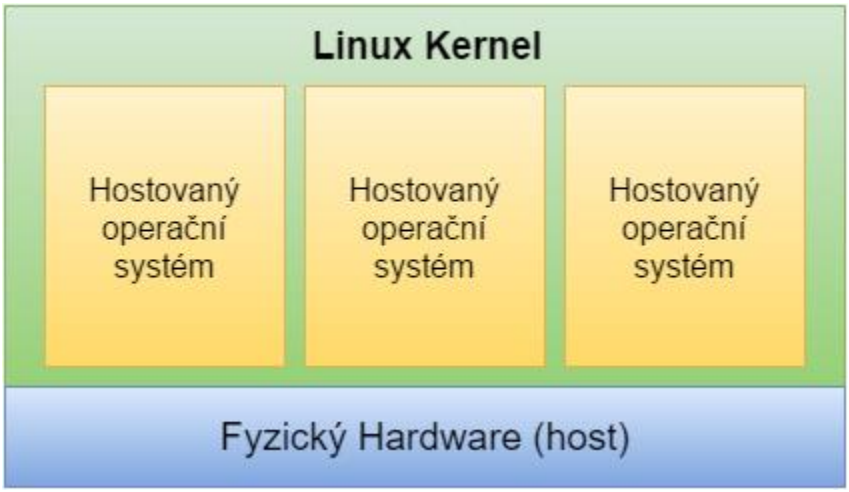
Nevýhody:

- Omezené množství operačních systému k virtualizaci. Není možné virtualizovat Windows, dále není možné virtualizovat na jedné verzi Kernelu, operační systém Linuxu či Unixu, které vyžadují rozdílnou verzi Kernelu [9].

- Nedostatečná úroveň izolace jednotlivých virtualizovaných hostů, veškeré zabezpečení okolo chroot bylo designováno jako opatření proti omylům a ne jako opatření proti případným útokům nebo poruchám.

3.4.2 Virtualizace na úrovni Kernelu

U tohoto druhu virtualizace funguje hostující systém na speciálně modifikovaném Kernelu. Tato modifikace obsahuje několik rozšíření, která následně řídí více virtuálních strojů, kde každý z nich obsahuje operační systém. Na rozdíl od virtualizace sdílením Kernelu, zde každý virtuální stroj obsahuje operační systém, který má svůj vlastní Kernel. Ale i zde jsou podobné restrikce jako u předešlé virtualizační metody, a to hlavně v tom, že každý hostovaný stroj musí obsahovat Kernel, který je kompilovaný pro ten samý hardware, na jakém funguje hostující stroj [9].



Obrázek 11 - Architektura virtualizace na úrovni Kernelu [9]

Výhody:

- Velmi jednoduché nasazení

- Vyšší izolace jednotlivých virtuálních strojů

Nevýhody:

- Není možné virtualizovat nic jiného než systémy architektury Unix nebo Linux

3.5 Stručná tabulka výše zmíněných serverových virtualizací podle autorů [4]

TYP VIRTUALIZACE	KRÁTKÝ POPIS	VÝHODY	NEVÝHODY
PLNÁ SOFTWAROVÁ VIRTUALIZACE	Tato technika plně emuluje veškerý hardware, který je prezentován virtuálnímu stroji.	Poskytuje úplnou izolaci jednotlivých virtuálních strojů. Nevyžaduje modifikaci hostovaného operačního systému.	Nízká výkonost během vyřizování privilegovaných instrukcí s přístupem na procesor.
PARAVIRTUALIZACE	Tato technika poskytuje částečnou emulaci použitého hardwaru, většina ale ne všechny funkce jsou simulovány.	Je jednodušší na implementaci dále je jednou z nejvýkonnějších virtualizačních technik pro síťové a I/O operace.	Hostované operační systémy musí být modifikované. Díky této skutečnosti jsou mnohem hůře přenositelné.
HARDWAROVĚ ASISTOVANÁ VIRTUALIZACE	Tato technika je jedna z nejnovějších pro architekturu x86, je to hybridní přístup plné softwarové virtualizace a procesorové hardwarové akcelerace umožněné diky Intel VT a AMD-V.	Nevyžaduje modifikaci operačního systému hosta ani operačních systémů virtuálních strojů.	Vyžaduje procesor, který obsahuje hardwarové rozšíření, umožňující hardwarovou akceleraci.

3.6 Terminologie

Virtualizace jako taková existuje již mnoho let. Díky objevu nových virtualizačních metod a díky masové dostupnosti a ekonomické nenákladnosti pro architekturu x86, se slovní termíny s virtualizací spojené vyskytují nyní mnohem častěji nežli dřív. Často je ovšem tato terminologie a názvosloví nepřesné, nedostatečné nebo dvojsmyslné. Velmi často se stává, že samotné společnosti, které tento virtualizační software s využitím různých virtualizačních metod vyvíjejí, použijí zavádějící výraz záměrně, ve snaze odlišit novou funkci svého produktu od zbytku jeho ostatních funkcí. Nejlepším příkladem je projekt Xen. Xen se zpravidla nejvíce zabývá metodou paravirtualizace, tedy metodou která vyžaduje modifikaci hostovaného operačního systému. Xen ale také podporuje metodu Hardwarově asistované virtualizace, kdy je zapotřebí aby hostující hardware obsahoval hardwarové rozšíření procesoru VT-x nebo AMD-V. Tuto metodu označuje za plnou virtualizaci.

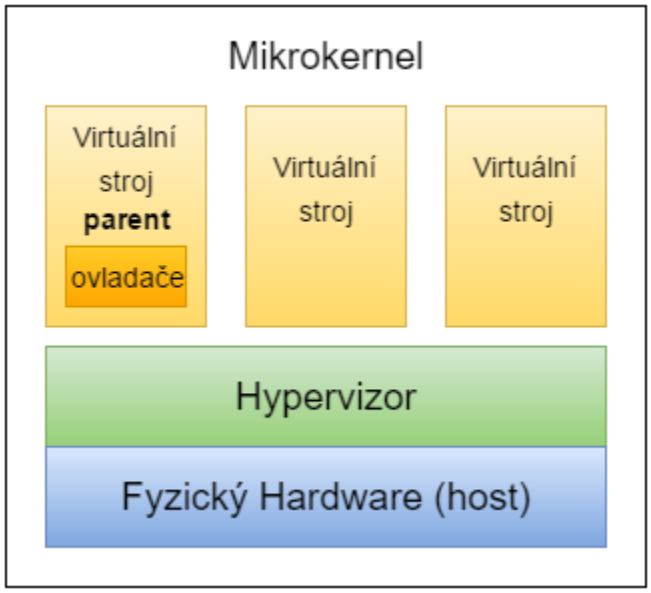
Za mnoho dalších nepřesností může samotná komunita kolem virtualizace. Proto je vždy nejprve výhodnější zjistit, jak daná virtualizační metoda funguje a potom se jí pokusit zařadit do jedné z několika pevněji definovaných skupin, jako jsou právě serverové nebo kontejnerové virtualizace. A následně je na základě funkcí rozřadit mezi paravirtualizaci, hardwarově asistovanou virtualizaci a další.

4 Virtualizační řešení

V této kapitole se zaměříme na jednotlivá softwarová řešení, zprostředkovávající virtualizaci. Jsou to jednotlivé produkty různých společností. Popíšeme si také, jakou metodu virtualizace jsou schopny používat a jaká je jejich architektura.

4.1 Hyper-V

V roce 2008 představil Microsoft svou RTM (release to manufacturing) verzi „Microsoft Windows Server 2008“ ke které byla přidávána první verze Hyper-V. Toto ovšem nebyl první virtualizační produkt od Microsoftu, Virtual PC byla desktopová aplikace, kterou bylo možné nainstalovat na základní operační systém a následně spustit sekundární operační systém. Takto mohl mít uživatel mít simultánně spuštěné dva operační systémy zároveň. S následným vydáním další verze produktu „Microsoft Windows Server 2008“ Microsoft představil svého prvního plnohodnotného 64bitového hypervizora známého též jako Hyper-V.
V této verzi Microsoft dále představil „Microsoft Cluster Service“ (MSCS), MSCS byla utilita která zaručovala rychlou dostupnost virtuálních strojů. Tento způsob rychlé dostupnosti se začal nazývat rychlá migrace. Princip je že všechny virtuální stroje jsou umístěny na sdíleném clusteru a pokud jedna z Hyper-V cluster nod selže, veškeré virtuální stroje, které jsou na ní umístěné, budou rychle migrovány na jinou Hyper-V cluster nodu. Další verze samostatného Hyper-V dostala mnoho vylepšení oproti verzi, která byla distribuovaná s Microsoft server 2008. Za zmínku stojí „Live migration of Virtual machines“ a „Cluster shared volumes“ [10].



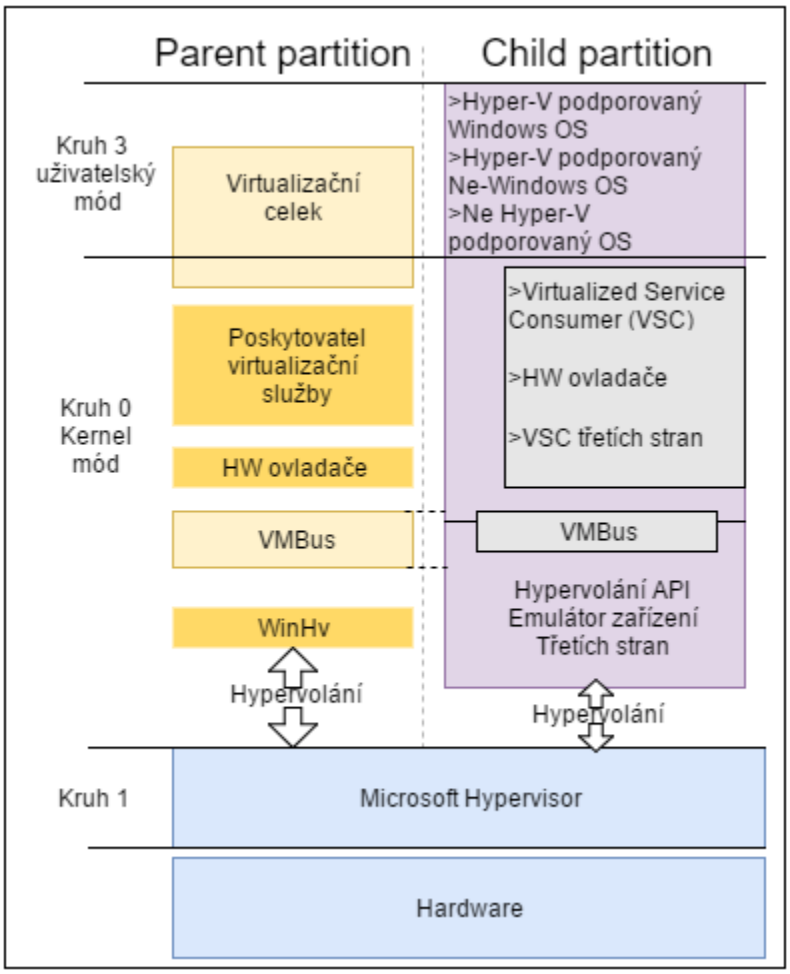
Obrázek 12- Schéma Mikrokernel [10]

Hyper-V patří dále do skupiny Mikrokernelových hypervizorů. V Mikrokernelových hypervizorech mají veškeré hardwarové ovladače oprávnění jak do kruhu 0 tak do uživatelského kruhu 3. Dále je pak veškeré plánování procesů a paměti vykonávané v kruhu 1. Výhoda tohoto druhu hypervizoru spočívá v tom, že mnoho výrobců dodává ovladače ke svému hardwaru alespoň pro jeden operační systém. Není proto problém najít kompatibilní hardware pro Mikrokenelové hypervizory. Mikrokernelový hypervizor totiž

vyžaduje, aby tyto ovladače, které jsou nezbytné pro fyzický hardware, byly instalovány na operační systém, který běží v takzvané „parent partition“. To znamená, že nemusíme instalovat ovladače pro fyzický hardware na každý operační systém nám vytvořeného virtuálního stroje. Další z výhod mikrokernelových hypervizorů je, že nenarušují koncept TCB[2] a pracují tak jen ve předem vyhraněných privilegovaných mezích [10].

„Parent partition“ je logická jednotka, která má veškerý přístup ke všem dostupným hardwarovým zařízením u lokálního virtualizačního celku. Dále má také oprávnění vytvářet „child partitions“ a spravuje veškeré komponenty které pod „child partitions“ spadají [10].

Jak již bylo zmíněno výše, „Child partition“ je logická jednotka která je vytvořena parent jednotkou a veškeré komponenty, které jsou ve správě hostovaného virtuálního stroje, spadají pod child jednotku [10].



Obrázek 13-Hyper-V Architektura [10]

[2] Trusted computing base – více lze nalézt na http://csrc.nist.gov/publications/history/dod85.pdf

Abychom správně porozuměli Hyper-V parent jednotce, musíme si říci, co se děje ve chvíli kdy se dokončuje instalace Hyper-V. V momentě kdy startujeme server, abychom dokončili instalaci Hyper-V, dojde ke vzniku parent jednotky. Hypervizor parent jednotky je pak následně umístěn pod vrstvu operačního systému. Nyní operační systém Windows Server 2012 běží nad vrstvou hypervisoru parent jednotky. Jednoduše řečeno si představme parent jednotku jako mozek celého Hyper-V virtual stack managementu a veškeré ostatní komponent jsou nainstalovány na oddělené child jednotky. Dále pak parent jednotka zajišťuje, že hypervisor má dostatečná oprávnění pro přístup k veškerým hardwarovým zdrojům a zajišťuje, že zatím co jsou tyto zdroje používány bude dodržen koncept TCB. Ke všem těmto úkolům, parent jednotka ještě dále zajišťuje opětovné vytvoření sebe samé při startu Hyper-V systému. Během normálního fungování virtuálních strojů na Hyper-V serveru je to právě parent jednotka která zajišťuje fungování child jednotek na hypervisoru. Poslední věc, kterou parent jednotka obstarává, je fungování jako prostředník mezi child jednotkami a hardwarem ve chvíli kdy child jednotky (virtuální stroje) přistupují k hardwarovým zdrojům [10].

Další z řady služeb, které zajišťují bezchybný chod Hyper-V je „Virtual Machine Management Service“. Tato služba, je řídícím prostředím Hyper-V, a je zodpovědná za stav jednotlivých virtuálních strojů pro child jednotky, a to včetně rozhodovací schopnosti měnit stav virtuálních strojů, dále pak kontrola vytváření snapshotů a řízení přidávání nebo odebírání různých zařízení. Pokud je virtuální stroj v child jednotce spuštěn, VMMS pro tento virtuální stroj vytvoří nový pracovní proces, který pak slouží pro vykonávání různých řídících a kontrolních procesů.

Jako poslední službu k popisu si vybereme „Virtual machine bus“ neboli také VMbus. VMbus je komunikační médium mezi parent jednotkou (hypervisorem) a child jednotkou (virtuálním strojem). VMbus je středobod veškerého managementu a přenosu dat, z parent jednotky na child jednotku a obráceně. Veškeré instrukce jdou z parent jednotky do child jednotky skrze VMbus. Pokud by např. virtuální stroj potřeboval přístup k DVD-ROMu který je na hypervisor serveru, šel by požadavek z child jednotky (virtuálního stroje) do DVD-ROMu skrz VMbus.

4.2 VMware ESXi

VMware ESXi je součástí uceleného balíčku nástrojů pro podnikové virtualizace jménem VMware vSphere. Sada těchto nástrojů dává uživatelům možnost vcelku nenáročně uvést do provozu virtualizaci podnikové úrovně. [11]

Roku 2007 byl na trh uvolněn VMware ESX 3.5. Společně s ním, společnost dále zveřejnila VMware ESXi 3.5. Tím se rozjela velmi zajímavá vývojová linie pro ESX model. VMware pak následně uvolňoval další verze souběžně spolu až do příchodu prvního balíku aplikací vSphere 5. Již od prvního uvolnění ESXi se VMware netajil tím, že v budoucnu ESXi plně nahradí ESX. Při uvedení vSphere 4.1 bylo ohlášeno, že tato verze bude poslední v, které bude ESX dostupný a všechny následující verze budou založeny výhradně na ESXi. Od té doby se ESX začal nazývat jako ESX classic aby tak bylo jednoduší jeho rozeznání od ESXi. Přestože měly tyto dva produkty odlišný design managementu, ve všem ostatním si byly velmi podobné, protože byly napsány na základě stejného kódu. [12]

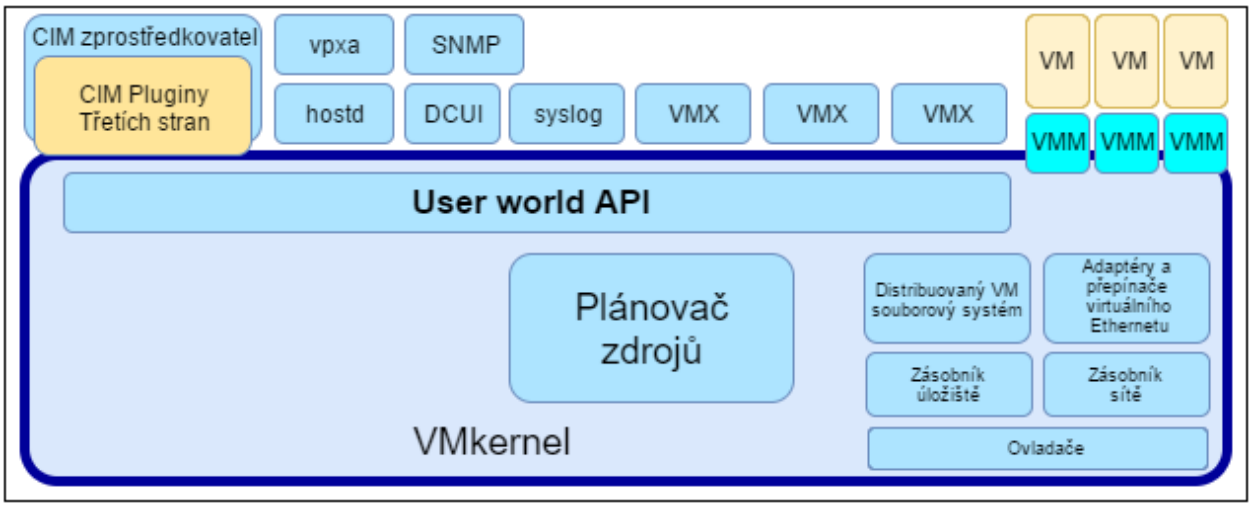
K oběma výše zmíněným hypervisorům se určitě hodí zmínit, že existuje i třetí verze a to tzv. „VMware vSphere hypervisor“ což je marketingové označení pro samostatnou verzi ESXi. Tato verze je volně ke stažení, ale je ochuzena o některé schopnosti plnohodnotné verze. Tento hypervisor nemůže být řízen skrze vCenter, dále pak veškerá vzdálená připojení skrze různá API (Application programming interface) mají povolená oprávnění pouze pro čtení. Dalším omezením je limit 32GB fyzické paměti pro jednoho hosta. Tato verze je proto spíše používaná jako tréninkový nástroj.[12]

Oba komerční produkty (krom samostatné verze ESXi zdarma) byly ceněny a licencovány stejně. ESX classic a ESXi mohou oba koexistovat ve stejném clusteru v jeden moment a sdílet zdroje mezi jednotlivými virtuálními stroji. Pokud nepracujete přímo s hosty samotnými, nebudete mít s největší pravděpodobností šanci poznat rozdíly v klientu, když se budete připojovat k vCenter.[12]

Základními kameny ESX classic jsou tři hlavní elementy, které fungují na fyzickém hardware a zprostředkovávají prostředí pro virtuální stroje. Dva z těchto elementů, VMkernel a VMM(Virtual Machine Monitor) jsou v podobných verzích stále k nalezení i v ESXi, třetí z této skupiny „Servisní konzole“ je jedním z klíčových rozdílů mezi ESX classic a ESXi a to z toho důvodu že se v ESXi již nenalézá. Servisní konzole též známá jako „Console Operating System“ (COS), nebo též jako VMnix, byla příkazová řádka pomocí které se pracovalo s ESX hypervisorem. Jednalo se o modifikovanou distribuci Red Hat Enterprise Linuxu, která umožňovala uživatelům privilegovaný přístup k VMkernelu. Konzole neměla žádný přímý přístup k fyzickému serveru nebo jakékoli z jeho hardwarových komponent. Ovšem je možné doinstalovat různé hardwarové ovladače, které následně konzoly umožňovaly spouštění skriptů na vnitřní infrastruktuře, hardwaru a spouštění agentů třetích stran. [12]

Tím, že v ESXi byla odstraněna servisní konzole od základu, změnilo, co jde s VMware hypervisorem dělat. ESXi je mnohem méně náročnější na výpočetní výkon a má také mnohem menší tzv. footprint, to znamená, že pro svou režii vyžaduje mnohem méně operační paměti stejně tak ESXi spotřebuje mnohem méně místa na úložišti na kterém je zrovna nainstalován ať už je to lokální disk nebo SAN space. Zredukováním kódu jádra se též dosáhlo většího zabezpečení ESXi. ESX classic měl po instalaci kolem 2 GB instalovaných souborů, kdežto ESXi má po instalaci pouhých 125 MB. Menší množství kódu je tak mnohem jednoduší zabezpečit a pak následovně bránit jelikož je mnohem méně míst k útoku. Další úpravou ESXi oproti ESX classic je zpráva procesů, ESX classic provozoval jeden virtuální hostovaný vCPU na kterém veškeré

procesy běžely sériově. V ESXi byla tato funkce přenesena do VMkernelu tím se dosáhlo většího prostoru pro veškeré procesy tak aby mohli být vykonávány paralelně, díky sofistikovanějšímu rozvržení. [12]



Obrázek 14 - VMware ESXi architektura [13]

VMkernel zprostředkovává prostředí v kterém pak mohou fungovat veškeré procesy které na systému běží, a to včetně managementu aplikací, agentů a dokonce i virtuálních strojů. Má kontrolu nad všemi hardwarovými zařízeními v serveru a řídí distribuci výpočetních zdrojů pro aplikace. Hlavní procesy která běží na VMkernelu jsou následující. [13]

- Direct Console User Interface (DCUI) – je nízko úrovňové konfigurační a řídící rozhraní, přístupné skrze serverovou konzoli a používá se většinou pro prvotní konfiguraci. [13]

- The Virtual Machine Monitor (VMM) – Což je prostředí v kterém se uskutečňují veškeré operace vyžadované Virtuálním strojem, dále zde běží proces VMX. Každý běžící virtuální stroj má své separátní procesy VMM a VMX. [13]

- Různí Agenti, kteří se používají k tomu, aby umožnily přístup k managementu VMware infrastruktury pro aplikace spravující vzdálený správu. [13]

- The Common Information Model (CIM) – CIM je rozhraní které umožňuje správu na hardwarové úrovni skrze sadu standartních API třetích stran. [13]

VMkernel dále používá velmi jednoduchý souborový systém, ve kterém má ESXi uložené konfigurační soubory, logy a aktualizace určené k instalaci. Souborový systém je v podstatě udělán tak aby se co nejvíce podobal souborovému systému, servisní konzole z ESX classic. Například ESXi konfigurační soubory jsou uloženy v */etc/vmware*, log soubory jsou v */var/log/vmware* a soubory připravené pro aktualizace jsou v */tmp*. Tento souborový systém je oddělen od VMware VMFS souborového systému který se používá k ukládání virtuálních strojů. Stejně jako je tomu u ESX i VMware VMFS datová úložiště mohou být vytvořena na lokálním disku hostujícího systému, nebo na nějakém vzdáleném sdíleném úložišti. ESXi nevyžaduje, aby byl nainstalován přímo na lokálním disku hostujícího systému. Tím že se spustí, bez-

disková metoda se zvýší spolehlivost celého hypervisoru, protože se vyvarujeme hardwarového selhání pevného disku. [13]

Jednou z dalších nezbytných součástí VMware ESXi architektury je „Direct Console User Interface" (DCUI). Je to lokální uživatelské rozhraní, které se zobrazuje pouze na konzoli ESXi systému. Vizuálně velmi blízce připomíná BIOS, ale je to soustava různých menu, které usnadňují interakci se zbytkem systému. Hlavní náplní tohoto rozhraní je umožnit prvotní konfiguraci a pak následně i pomoci v případě řešení problémů. Jeden ze systémových uživatelských profilů v VMkernelu je profil který nese stejnojmenný **dcui**, ten je používaný DCUI procesy, aby je bylo jednoduší identifikovat během komunikace s ostatními komponentami v systému. [13]

Prvotní konfigurace pomocí DCUI se skládá z:

- Nastavení administrátorského hesla. [13]
- Konfigurace sítě (pokud toto není provedeno automaticky pomocí DHCP). [13]

Troubleshooting (řešení problémů) pomocí DCUI se skládá z:

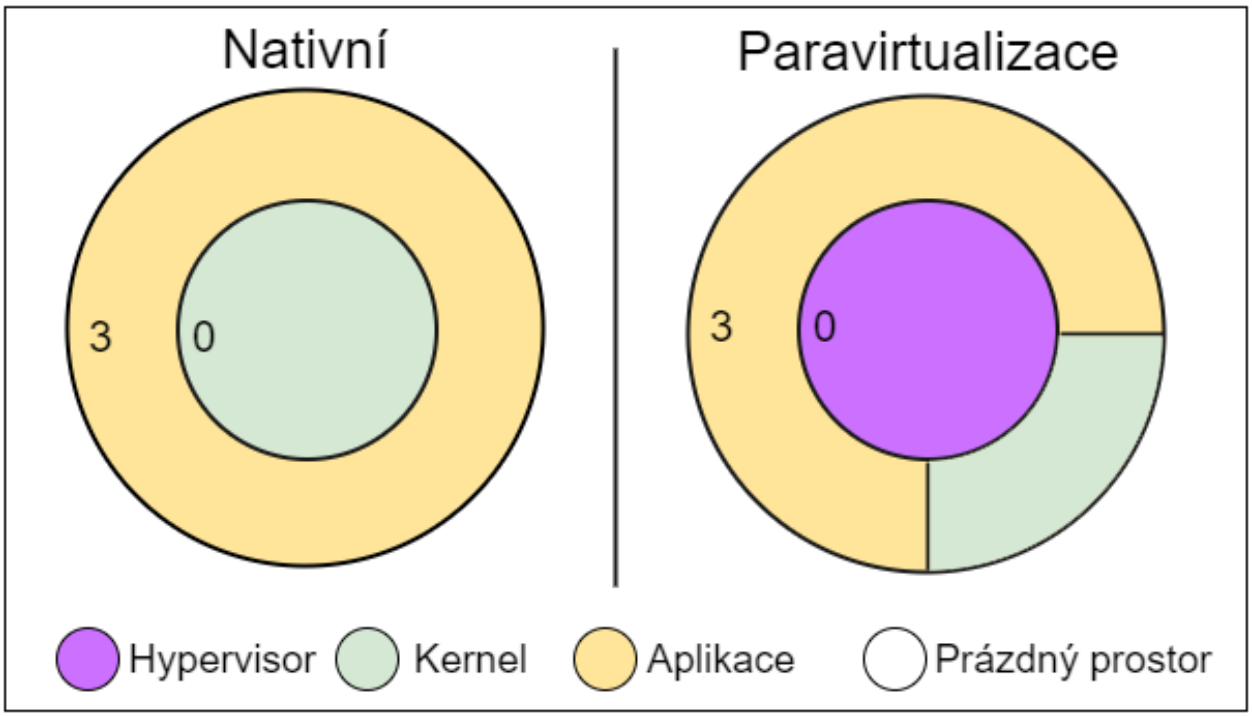
- Základní testování síťové konektivity. [13]
- Prohlížení Logů. [13]
- Restartování agentů. [13]
- Obnova základního nastavení. [13]

Když systém provádí první start, VMkernel provede několik operací, jako první začne objevovat veškerá dostupná zařízení a vybere pro ně vhodné ovladače. Dále pak prozkoumá lokální úložiště, pokud jsou nalezené disky prázdné, tak je VMkernel okamžitě naformátuje, aby mohli být následně použity pro uložení virtuálních strojů. Během tohoto prvního startu, VMkernel automaticky vytvoří konfigurační soubory obsahující dostupná základní nastavení (například nastaví síťové rozhraní pomocí informací, které mu byly přiděleny DHCP serverem). Uživatelé mohou upřesnit nastavení systému pomocí DCUI nebo pomocí standartních řídících nástrojů společnosti VMware, jako je například VMware VirtualCenter a VI Client. U verzí ESXi které jsou dodávány společně s hardwarem serveru, jsou konfigurační soubory uloženy ve specifickém paměťovém modulu, z kterého jde jak číst, tak do něj zle i zapisovat. U všech následujících restartů si systém přečte konfiguraci z pevné paměti. V dalších částech startovacího procesu se inicializuje systém, a pevný paměťový systém je zapsán do operační paměti. Jako další se načtou ovladače hardwaru, spustí se různé agenty a nastartuje se DCUI. [13]

4.3 XEN

Xen je pevně usazen mezi operačním systémem a hardwarem. Zprostředkovává prostředí, v kterém může běžet Kernel a skládá se ze tří hlavních komponent hypervisoru, kernelu a aplikací uživatelského prostředí. To jak do sebe tyto aplikace zapadají je důležité, protože způsob jak jsou tyto vrstvy na sebe kladené není absolutní, a ne všechny vytvořené virtuální stroje jsou si rovny. [14]

Největší změna pro kernel, který běží pod Xenem je v tom že se nevyskytuje v kruhu 0. Kde se v tu chvíli kernel vyskytuje, se pak liší v závislosti na platformě. Například na architektuře IA32 (Intel Architecture, 32-bit známí též jako i386) je kernel přesunut do kruhu 1. To kernelu umožní přistupovat k paměti, která je alokována pro aplikace, které běží v kruhu 3, zároveň ho, ale toto opatření před aplikacemi a ostatními kernely chrání. Hypervisor je uložen v kruhu 0, tam je chráněn před kernely které jsou v kruhu 1 a aplikacemi, které jsou v kruhu 3. V rámci vývoje architektury x86-64, prošla mnoha úpravami i výše zmíněná architektura IA32, jedna z věcí která byla změněna, byl počet ochranných kruhů. Z důvodu absence kruhu 1 a 2 bylo nutné upravit Xen tak aby umísťoval operační systém do kruhu 3 společně s uživatelskými aplikacemi, tak jak je vyobrazeno na obr. 15. [14]



Obrázek 15 - Využívání kruhů v x86-64 nativním a paravirtualizovaném systému [14]

Tento přístup využil Xen i u jiných CPU architektur, jako je například IA64 který má také jen dva ochranné kruhy. Další věc, která byla u architektury x86-64 odstraněna byla ochrana „segment-based“ paměti. To znamená, že Xen musel začít spoléhat na stránkovací ochranný mechanismus, aby mohl sám sebe lépe izolovat od hostů. [14]

Z perspektivy paravirtualizovaného kernelu je hned několik rozdílů mezi fungováním v prostředí Xenu nebo přímo na hardwaru hostitele. První z těchto rozdílů je mód procesoru během startu. Všechny x86 procesory od série 8086 startovaly v reálném módu. Pro 8086 a 8088 byl jediný dostupný mód 16-bitový mód, který měl přístup do 20-bitového adresního prostoru, ale neměl žádnou správu paměti. Od všech následujících modelů x86 CPU architektury, se očekává, že budou schopny spouštět starší verze softwaru, (tzv. „legacy software“) a to včetně operačních systémů. Jedním z prvních úkolů moderních operačních systémů je přehodit CPU do chráněného módu, který zprostředkovává možnost isolovat stav paměti jednotlivých procesů, a tím umožňuje vykonávání 32-bitových instrukcí. [14]

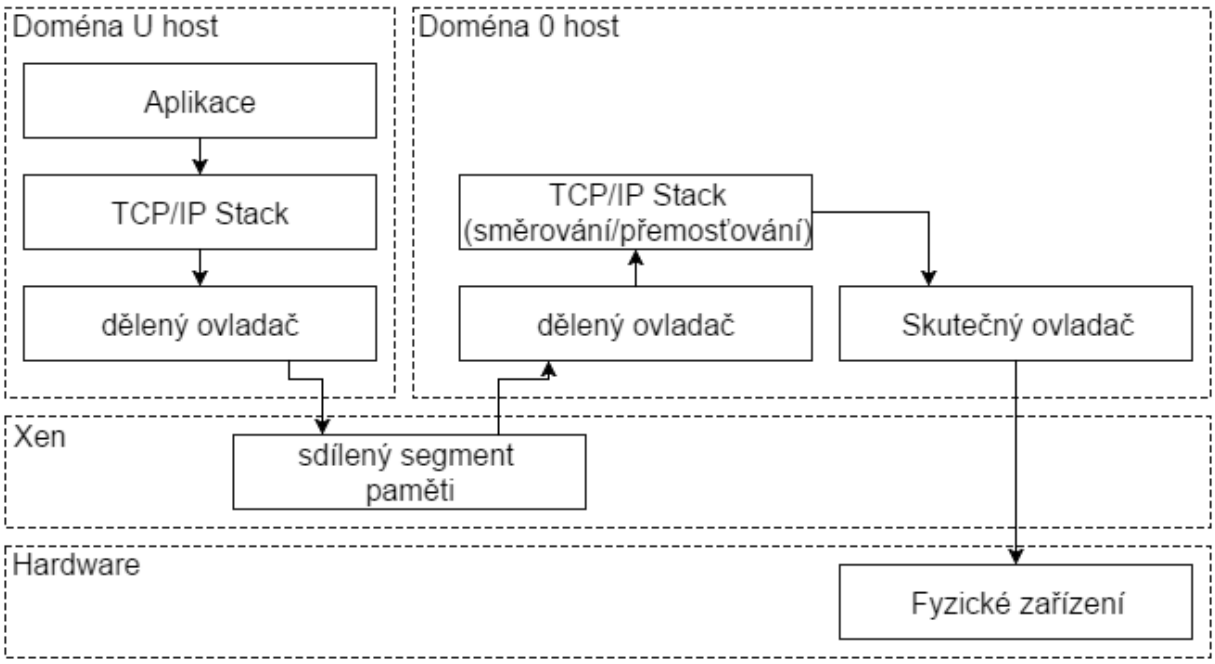
Jelikož Xen má na starost spouštění celého systému, provádí změnu, CPU módu do chráněného módu on sám. Pokud by tak neučinil, nebyl by schopný sám sebe dostatečně izolovat od toho, aby do něj zasahovaly hostované operační systémy. To dále znamená, že hostovaný kernel startuje v hodně rozdílném prostředí. Novější x86 systémy mají „Extended Firmware Interface“ (EFI), což je náhrada za PC BIOS. Všechny systémy, které jsou vybaveny EFI mohou startovat v chráněném módu. Nicméně i novější systémy nadále podporují starý PC BIOS. [14]

Tím, že je Xen umístěn v kruhu 1 namísto v kruhu 0, dochází, jak jsem již zmínil ke změně prostředí pro kernel. Kernel totiž v této situaci nemůže používat privilegované instrukce, namísto toho jsou tyto strojové instrukce nahrazeny hyper voláními (hypercalls), ta jsou konceptem velmi podobná systémovému volání. K další změně dochází v tom, jak je udržován čas. Operační systém potřebuje přesně měřit čas dvěma způsoby. Potřebuje vědět množství reálně uplynulého času a množství CPU času. První je k tomu aby měl uživatel přehled o tom jaký je reálný čas, ale zároveň je používán systémovými démony jako třeba Cron nebo k synchronizaci událostí napříč sítí. Druhý způsob je vyžadován pro multitasking, a to z toho důvodu aby každý proces pracoval s CPU jen po dobu pro tento proces přidělenou. Když kernel funguje ve svém nativním prostředí mimo hypervisor, je reálný čas a CPU čas, ta samá věc. Vše co má kernel na starosti je kontrolovat kolik času přiřadil jednotlivým procesům a jejich vláknům. Pokud se ovšem kernel vyskytuje v prostředí Xenu, musí sdílet dostupné procesory s jinými operačními systémy. To pro kernel znamená, že procesor bude pro jeho procesy přiřazen pouze po zlomek sekundy na každou celou sekundu reálného času. Proto musí konstantně re-synchronizovat svoje vnitřní hodiny s prostředky Xenu, které mají na starost držení reálného času.[14]

Účel hypervisoru je, aby umožňoval chod hostovaných virtuálních strojů. Xen provozuje virtuální stroje v prostředích, která označuje jako domény, které obsahují kompletní, fungující, virtuální prostředí. Když Xen startuje jedna z prvních věcí, která se vykoná je, nahrání Domény 0 (dom0) na hostovaný kernel. Ta je většinou specifikována v boot loaderu jako samostatný modul a proto může být nahrána i bez toho aniž by byly dostupné jakékoliv ovladače souborového systému. Doména 0 je první host, který je spuštěn a dále má větší privilegia než ostatní hosté. Ostatní hosté jsou proto označováni jako domény U (domU) U je pro Unprivileged (neprivilegované). Nicméně je možné delegovat některé povinnosti, které vykonává dom0, na některé z nižších domU hostů a to lehce mlží rozdíly mezi těmi to doménami. [14]

Doména0 je pro Xen velmi důležitá. Xen sám o sobě tož neobsahuje žádné hardwarové ovladače ani základní uživatelské rozhraní. Všechny tyto vlastnosti jsou zprostředkovány operačním systémem a uživatelskými aplikacemi, které fungují nebo se vyskytují v hostu domény0. Doména0 je většinou Linux, avšak mohou být použity i systémy jako je NetBSD a Solaris. Linux je používán většinou developerů kteří pracují na Xenu a obojí je distribuováno pod stejnými podmínkami a to GNU General Public Licence. [14]

Jedna ze stěžejních funkcí, kterou má na starost host domény 0 je ovládání hardwarových zařízení. Tento host funguje ve vyšší úrovni privilegií než ostatní hosté a proto má přístup k hardwaru. Z tohoto důvodu je nutné, aby privilegovaný host byl dostatečně zabezpečen. Část zodpovědnosti za zacházení s hardwarem je jeho multiplexování pro virtuální stroje. Je to z důvodu toho, že většina hardwaru nativně nepodporuje přístup několika operačních systémů naráz. Je dokonce nezbytné pro některé časti systému, zajistit, aby každý host měl své vlastní virtuální zařízení. [14]



Obrázek 16 - Cesta paketu z neprivilegovaného hosta skrze systém [14]

Na obrázku č. 16 je zobrazeno co se děje s paketem který je odeslaný aplikací, která běží na neprivilegovaném hostu. Jako první paket putuje skrze TCP/IP Stack (TCP/IP zásobník) tak jak by putoval normálně. Na dně stacku ovšem není standartní ovladač síťového zařízení. Jednoduchý kus kódu, kterým je tvořena první část děleného ovladače, umístí paket do paměti, která byla Xenem již předem určena ke sdílení pomocí „Xen grant table“, což je mechanismus který umožňuje neprivilegovaným hostům přístup do stránkovacích souborů, které jim nepatří. Paket je dále ze sdílené paměti přečten druhou částí děleného ovladače, která běží v doméně 0, ten ho dále vloží do firewallových komponent, kterými disponuje operační systém, typicky se jedná IPtables. Tyto komponenty s paketem zacházejí jako by přišel z reálného rozhraní. Jakmile paket projde skrze firewall, je odeslán na skutečný ovladač. Ten je schopen zapisovat do určitých částí paměti která je vyhrazená pro I/O a může vyžadovat přístup k IRQ skrze Xen. Fyzické zařízení pak následně odešle paket do sítě. [14]

Dělený ovladač v této situaci funguje pro hostovaný systém v doméně U stejně jako reálné rozhraní síťové karty. Xen zprostředkovává zjednodušený interface pro tato zařízení, který je jednoduché implementovat pro lidi kteří migrují svůj systém na Xen. Každý ovladač se skládá ze tří komponent. [14]

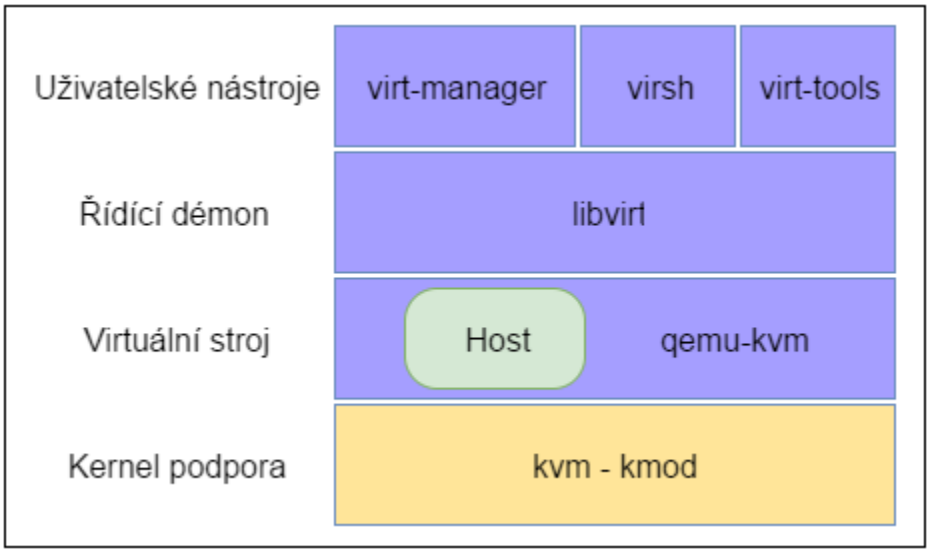
- Dělený ovladač
- Multiplexér
- Reálný ovladač

Neprivilegované domény (domU) operují s mnohem většími restrikcemi než dom0. DomU host většinou nemá povoleno provádět žádné hypercally které přistupují přímo na hardware, i když v některých situacích je možné domU povolit přístup k jednomu nebo více zařízením. Namísto toho aby domU přistupoval přímo k hardwaru, raději implementuje front end nějakého děleného ovladače. Jako minimalistické řešení funguje použití XenStore (informační paměť sdílená napříč doménami) a ovladače konzole. Většina domU hostů ještě implementuje síťový interface. Protože jsou toto generické, abstraktní zařízení, domU hosté potřebují implementovat pouze jeden ovladač od každé kategorie zařízení. Z tohoto důvodu bylo mnoho rozdílných operačních systémů, které z důvodů chabé podpory hardwaru na kterém přímo fungovaly, upraveno tak aby mohli fungovat jako hosti domény U v prostředí Xenu. Xen umožňuje hostům využívat výhod rozšířené podpory hardwaru hosta domény 0. [14]

4.4 KVM

KVM (kernel-based virtual machine) reprezentuje nejnovější generaci open-source virtualizací. Cílem projektu bylo vytvořit moderní hypervisor který je postaven na zkušenostech předešlých technologií a přitom plně využívá výhod moderního dostupného hardwaru. Tedy hlavně hardwarové podpory procesoru (VT-x, AMD-v). [15]

KVM jednoduše transformuje Linux kernel na hypervisor (když nainstalujete KVM kernel modul). Jelikož standartní Linux kernel funguje podobně jako hypersor, může ze změn tohoto kernelu KVM benefitovat ve formě lepší podpory paměti, plánování atd. Z optimalizace těchto linuxových komponent, těží jak hypervisor (hostitel) tak hostované linuxové operační systémy. Pro emulaci I/O používá KVM software zvaný QEMU. QEMU je program který má na starost hardwarovou emulaci. Dokáže, emulovat CPU a řadu dalších zařízení jako je pevný disk, síťová karta, serial nebo USB port. Vybuduje tak kompletní virtuální hardware, na který pak následovně lze nainstalovat hostovaný operační systém. [15]

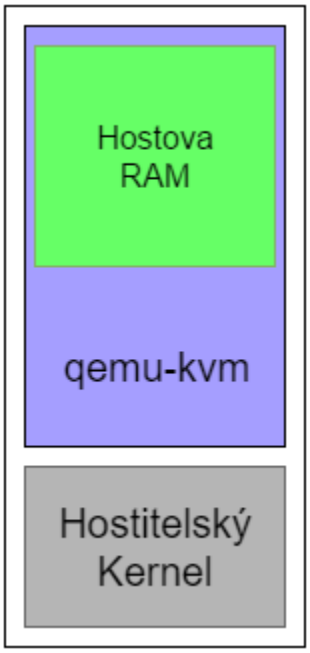


Obrázek 17 - Schéma nástrojů KVM [16]

V srdci KVM je Linux kernel modul, který bezpečně vykonává hostův kód přímo na procesoru hostitele. Toto efektivní řešení je možné díky výše zmíněným technologiím od firem Intel a AMD. Ty uvedli hardwarová rozšíření VT-x a AMD-v na trh kolem roku 2006, nyní jsou dostupná v každém moderním x86 procesoru. Tato rozšíření přidala nový virtualizační mód pro přímý přístup na procesor, paměti a jiné zdroje, které nebyli dříve dostupné pro nemodifikované hosty. [16]

Zatímco hostův kód se vykoná v bezpečí přímo na procesoru hostitele, většina I/O volání je blokována namísto toho, aby byla posílána přímo na hostitelská zařízení. Host vidí emulovaný čip set a PCI sběrnici, na které lze připojit emulovaná zařízení a emulované přeposílací adaptéry (např. VGA adaptér). KVM disponuje paravirtualizovanými síťovými rozhraními, úložišti a tzv. balónkovými ovladači, ty umožňují dynamické alokování a uvolňování operační paměti hosta během fungování virtuálního stroje. [16]

Emulace hardwaru je vykonávána díky qemu-kvm a to v procesech uživatelského prostředí přímo na hostiteli. To umožňuje kernel modulu zůstat nezatíženým a plně soustředěným na výkonnostně kritické aspekty, zatímco zařízení uživatelského prostředí se emulují v izolovaných procesech, které se vykonávají mimo kernel hostitele. Software sVirt je řídící program který usnadňuje manipulaci s KVM a qemu-kvm, sVirt je vybaven schopností uzamykat qemu-kvm procesy pomocí SELinux „povinné kontroly přístupu“, tudíž qemu-kvm může přistupovat pouze k souborům a zdrojům které potřebuje a ničemu jinému. [16]



- QEMU je proces uživatelského prostředí
- Sobory a zdroje hostitele jsou chráněny SELinuxem
- Každý KVM virtuální procesor (vCPU) je interpretován jako vlákno procesu.
- Plánovač hostitelského kernelu rozhoduje o tom, kdy budou virtuální procesory hostů fungovat.

Obrázek 18 - QEMU model procesů [17]

Správcovské nástroje potřebují monitorovat a přistupovat na hosty, kteří jsou lokálně nebo na vzdáleném serveru. To je řešené skrze různé API a služby, které umožňují aplikacím manipulovat s hosty a automatizovat řídící povinnosti. Libvirt poskytuje „language binding“, což znamená, že řídící povely mohou být zakomponovány do různých programovacích jazyků. Dále nabízí řádkové rozhraní, které usnadňuje skriptování běžných operací. [16]

Hostitel běží společně se svým libvirt démonem, ten zprostředkovává vzdálenou správu, ale může být použit i pro lokální nastavení všech přítomných hostů a nemusí být viditelný ze sítě. Libvirt démon udržuje konfigurace hostu během restartů a je centrem pro vytváření sítě a připojování vzdálených úložišť. [16]

5. Benchmark XEN/KVM

V rámci práce je nutné provézt testování prací vytyčených vyrtualizačních softwarů v předem vytyčeném typu virtualizace. Testoval jsem, proto jednotlivé hardwarové komponenty CPU, HDD a RAM pod hypervisory Xen a KVM za pomoci správcovského software Libvirt a v módu hardwarově asistované virtualizace.

Všechny testy byly prováděny s maximálním ohledem na stejné podmínky pro běh těchto testů. Po každé sadě testů byly systémy restartovány s následovnou šedesátisekundovou pauzou, která zajistila stabilitu systému.

5.1 Sestava, Kernel, Verze.

Jako testovací stroj jsem zvolil jeden z počítačů v síťových laboratořích FIM UHK v budově J. V PC jsem za pomoci softwaru G-parter a GRUB2 vytvořil prostor pro další operační systém tak aby mohl hostující operační systém komunikovat přímo s Hardwarem.

- CPU: Intel Core 2 duo 3.00GHz
- RAM: 4,00 GB DDR2

Jako hostující a hostované linuxové distribuce byly zvoleny CentOS a Debian, obě tato distra byla zvolena z důvodů, že jsou to kořenové distribuce, na jejichž vývoji se neustále pracuje a kernel je tak vysoce stabilní. Obě použité verze jsou tkzv. Stabilní verze, jejichž jádro se nemění s každou novou aktualizací, jako je tomu u Rolling verzí.

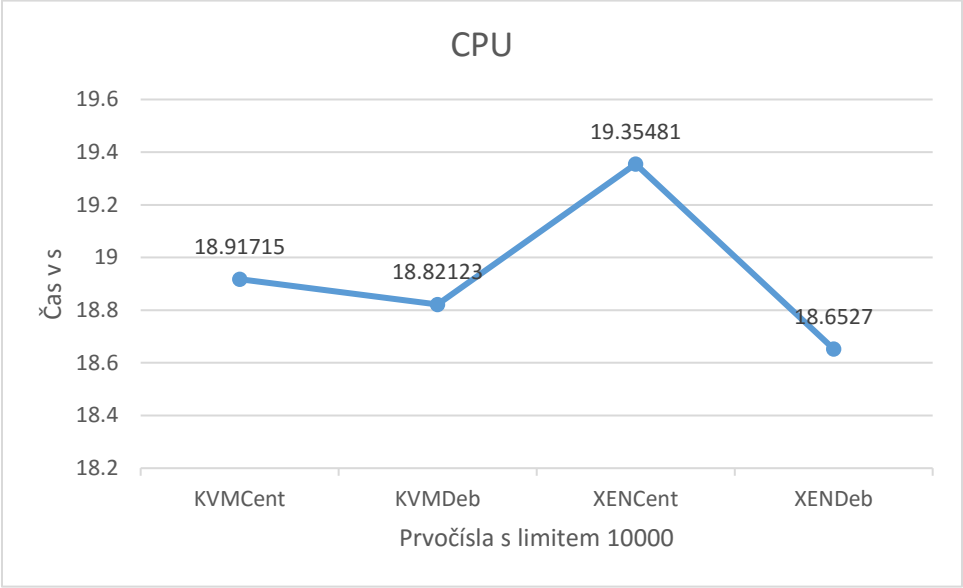
- CentOS 7 (1511)
- Debian (“wheezy“)

Verze hypervisorů byly použity ty nejnovější v době provádění testů. Bylo tak učiněno proto, aby se předešlo velkému množství problémů s nekompatibilitou výše zvolených operačních systémů. Nová vydání totiž vždy opravují chyby, které většinou vzniknou s předchozím vydáním OS. U KVM se bohužel musela požít starší verze pro Debian OS z důvodu toho že novější nebyla plně stabilní.

- Xen 4.5
- KVM 3.2 pro Debian

5.2 CPU test

Jako první bylo rozhodnuto otestovat výkonost CPU. Hostem byl Operační systém CentOS a testovala se výkonost v HVM (Hardware Virtualized Machine) jak u KVM, tak u XEN. Test byl proveden za pomoci nástroje „Sysbench“ a parametry testu byly následující: Výpočet prvočísel s limitem 10 000. Příkaz, který jsem použil je následující. *„sysbench –test=cpu –cpu-max-prim-10000 run“.*



Obrázek 19 - CPU test

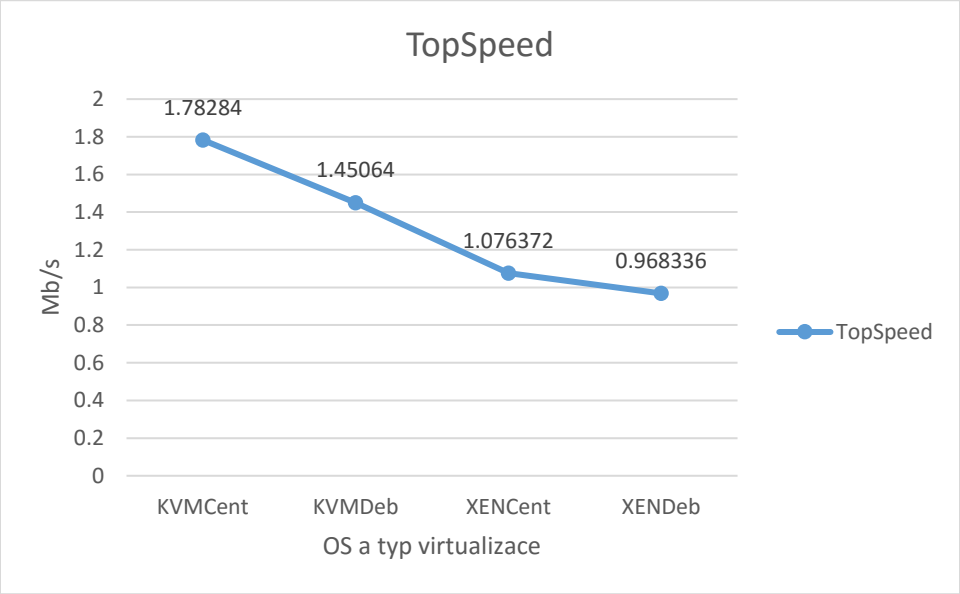
CPU

KVMCent		**KVMDeb**		**XENCent**		**XENDeb**	
Průměr	Směr. Odchylka	Průměr	Směr. Odchylka	Průměr	Směr. Odchylka	Průměr	Směr. Odchylka
18,91715	0,33	18,82123	0,22	19,35481	0,31	18,6527	0,06

Test byl proveden desetkrát se stejnými parametry. Jak je patrné z grafu a tabulky, všechny systémy a virtualizační nástroje si vedly velmi srovnatelně. Jedinou výjimku zde představuje Systém CentOS běžící na virtualizačním nástroji XEN. To může být způsobeno několika věcmi, nejpravděpodobněji tím, že kernel, operačními systému Centos posílá, příkazy na CPU takovým způsobem, díky kterému je XEN může interpretovat hostitelskému systému rychleji.

5.3 I/O test

Tento test se zaměřuje na efektivnost zapisování a čtení z HDD, benchmark vytvoří náhodně generovaná data, která budou, rozdělena na několik kusů s celkovou velikostí větší než je paměť RAM. Následně provede náhodné zapisování a čtení těchto dat. Stejně jako v předešlém případě, i nyní byl použit testovací nástroj Sysbench. Příkaz, který byl použit, je následující:
Přípravný příkaz – *„sysbench --test=fileio --file-total-size=5G prepare“*
Spouštěcí příkaz – *„sysbench --test=fileio --file-total-size=5G --file-test-mode=rndrw --init-rng=on --max-time=300 --max-requests=0 run“*
Graf zobrazuje průměrné nejvyšší rychlosti přenosu dat. Dom0 byl v tomto případě Distribuce CentOS.



Obrázek 20 - I/O test

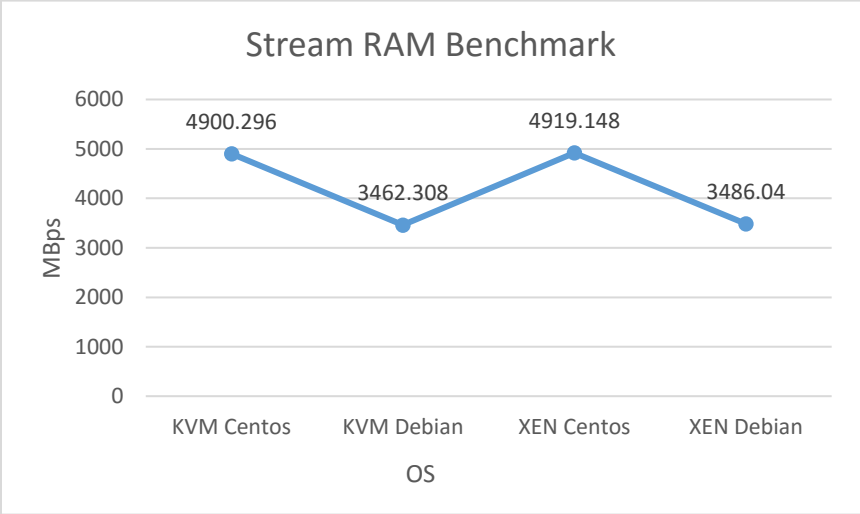
I/O TopSpeed

KVMCent		**KVMDeb**		**XENCent**		**XENDeb**	
Průměr	Směr. Odchylka	Průměr	Směr. Odchylka	Průměr	Směr. Odchylka	Průměr	Směr. Odchylka
1,78Mb/s	0,25	1,45MB/s	0,17	1,08MB/s	0,17	0,97MB/s	0,02

Tento test nám ukázal mnohem větší rozdíly mezi nasazenými hypervizory, zatímco rozdíly mezi použitými operačními systémy byly zanedbatelné. Toto se dá vysvětlit způsobem, jakým KVM cachinguje data při zápisu a při čtení. KVM má několik nastavení jakými ukládá data do bufferu, a tak může podávat ve stejných situacích odlišné výsledky.

5.4 RAM Benchmark

Tento test je určen, aby plně zatížil RAM paměť. K testu byl použit „Stream Benchmark“, který je součástí „Phoronix test suite“ nebo také PST. Stream je navržen tak, aby měřil maximální udržitelnou memory bandwidth namísto chvilkových maxim výkonu systému. Stream benchmark má čtyři operační módy: copy, scale, sum, triad. V tomto testu byl užit pouze mód copy, jelikož zbylé tři módy spoléhají na silné CPU, aby mohly provést různé výpočty s daty, předtím než je začnou zapisovat do paměti. Toto je v kontrastu s čistou kopií která měří jen transferní hodnoty, aniž by prováděla další aritmetiku. Stream namísto toho kopíruje objemné řady informací z jedné lokace do druhé. Benchmarker specifikuje řadu tak, aby byla větší, než cache stroje, tak aby data nebylo možné použít znovu.



Obrázek 21 - RAM test

Stream RAM Benchmark

KVMCent		**KVMDeb**		**XENCent**		**XENDeb**	
Průměr	Smě. Odchylka	Průměr	Smě. Odchylka	Průměr	Smě. Odchylka	Průměr	Smě. Odchylka
4900,3MBps	4	3462,3MBps	9,18	4919,1MBps	1,85	3486,04MBps	4,17

Při pohledu na hrubé výsledky jednotlivých testů byl Stream RAM Benchmark velmi konzistentní. Hlavně když se porovnají s výkyvy výsledků v některých předešlých testech. Operační systém Debian ovšem výrazně zaostává za Operačním systémem CentOS. V tomto případě ovšem nejspíše nebude na vině Hypervisor, který překládá požadavky systému na Hostující fyzickou RAM, na které konstantně refreshuje cache hosta, ale vnitřní manipulace s řadami informací, které Stream pro test vytvořil.

Závěr

Závěrem této práce by se dalo říci, že její cíle byly naplněny. Prošli jsme základní koncepty virtualizace, dále jsme si přiblížili různé virtualizační principy, popsali jsme různé typy virtualiza a blíže jsme se seznámili s několika virtualizačními řešeními. Na konci práce jsme si představily výsledky zátěžových testů CPU, HDD a RAM, ve virtualizačních prostředích XEN a KVM na dvou hlavních Linuxových distribucí. Z výsledků testů jsme zjistili že výkonost těchto hypervisorů při hardwarově asistované virtualizaci je více či méně stejná. Pro koncového uživatele tedy otázka výkonu nebude nutná a bude moci zvolit hypervisor který bude třeba méně konfliktní s jeho stávající konfigurací.

Dá se říct, že toto platí, jak u drobných uživatelů, tak u větších firem či korporací. Běžný uživatel s největší pravděpodobností na své Linuxové distribuci na první pohled nepozná jaký hypervisor používá. Pokud se bude jednat o hardwarově asistovanou virtualizaci, bude s největší pravděpodobností hypervisor lokálně ovládán skrz balíček aplikací Libvirt a jednotlivé hostované stroje se budou chovat navenek stejně jak u KVM, tak u XEN. Ani velké společnosti se nerozhodují, jaké virtualizační řešení použijí na základě výkonnostních testů. Budou postupovat podle toho, jaká společnost jim dodá lepší doprovodný servis a zabezpečení a správcovské API. A nejspíše i podle toho jaké řešení jim přinese co nejméně rozruchu do jejich stávajícího systému. Proto nejspíše i nadále budou freeware verze XEN a KVM hypervisorů záležitostí vývojářů, začínajících systémových administrátorů, nadšenců a běžných uživatelů Linuxových systémů.

Literatura:

1. PORTNOY, Matthew. *Virtualization essentials*. Indianapolis, IN: John Wiley & Sons, Inc., 2012. ISBN 1118176715.

2. HESS, Kenneth a Amy NEWMAN. *Practical virtualization solutions: virtualization from the trenches*. Upper Saddle River, NJ: Prentice Hall/Pearson Education, c2010. ISBN 0137142978.

3. TAKEMURA, Chris a Luke S. CRAWFORD. *The book of Xen: a practical guide for the system administrator*. San Francisco: No Starch Press, c2010. ISBN 1593271867.

4. DITTNER, Rogier a David RULE JR. *Best damn server virtualization book period: including Vmware, Xen, and Microsoft Virtual Server*. Oxford: Elsevier Science [distributor], 2007. ISBN 1597492175.

5. *VMware understanding full Virtualization, Paravirtualization, and hardware assist* Z: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf (cit.: 18 Května 2016).

6. AMD (2012) *AMD64 technology AMD64 architecture programmer's manual volume 2: System programming*. Z: http://developer.amd.com/wordpress/media/2012/10/24593_APM_v2.pdf. (cit: 18 Května 2016).

7. WOLF, Chris. a Erick M. HALTER. *Virtualization: from the desktop to the enterprise*. New York, NY: Distributed in U.S. by Springer-Verlag New York, c2005. ISBN 9781590594957.

8. GOLDEN, Bernard. *Virtualization for dummies*. Hoboken, N.J.: Wiley, c2008. ISBN 0470148314.

9. Smyth, N. (2009) *Xen Virtualization essentials 1, Neil Smyth, eBook*. https://www.amazon.com/Xen-Virtualization-Essentials-Neil-Smyth-ebook/dp/B0026IBZJE ASIN: B0026IBZJE

10. ZAHIR HUSSAIN SHAH. *Windows server 2012 hyperv: deploying the hyperv enterprise server*. Online-Ausg. S.l.: Packt Publishing Limited, 2013. ISBN 1849688346.

11. MARSHALL, Nick. *Mastering VMware vSphere 6*. Indianapolis: John Wiley and Sons, 2015. ISBN 9781118925157.

12. GUTHRIE, Forbes a Scott LOWE. *VMware vSphere design*. 2nd ed. Indianapolis, Ind.: Sybex, c2013. ISBN 978-1-118-40791-2.

13. VMware (2008) *The architecture of VMware ESXi*. Dostupné na : http://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/ESXi_architecture.pdf

14. CHISNALL, David. *Definitive guide to the xen hypervisor*. S.l.: Prentice Hall, 2013. ISBN 0133582493.

15. MUKHEDKAR, Prasad, Anil VETTATHU a Humble CHIRAMMAL. *Mastering KVM Virtualization*. Birmingham, United Kingdom: Packt Publishing, 2016. ISBN 1784399051.

16. Hajnoczi, S. ibmvirtualization (2010) KVM architecture: The key components of open Virtualization with KVM (Virtualization@IBM). Dostupné na: https://www.ibm.com/developerworks/community/blogs/ibmvirtualization/entry/kvm_architecture_the_key_components_of_open_virtualization_with_kvm2?lang=en.

17. Hajnoczi, S. (2015) 'KVM architecture overview', Dostupné na :

http://vmsplice.net/~stefan/qemu-kvm-architecture-2015.pdf.