VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMANÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

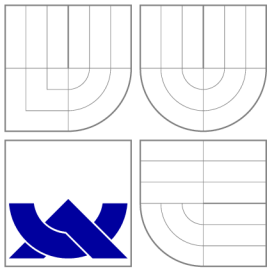
TECHNIKY VIRTUALIZACE VÝPOČETNÍCH PLATFOREM V LINUXU

BAKALÁŘSKÉ PRÁCE
BACHELOR'S THESIS

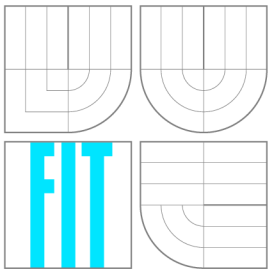
AUTOR PRÁCE
AUTHOR

JIŘÍ ŽUPKA

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMANÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

TECHNIKY VIRTUALIZACE VÝPOČETNÍCH PLATFOREM V LINUXU

TECHNIQUES OF VIRTUALISATION OF COMPUTING PLATFORMS IN LINUX

BAKALÁŘSKÉ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ ŽUPKA

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. TOMÁŠ VOJNAR, Ph.D.

BRNO 2008

Abstrakt

Práce testuje a porovnává nástroje pro virtualizaci výpočetních platforem a vysvětluje pojmy důležité pro pochopení problematiky virtualizace. Jejím hlavním přínosem je srovnání s pohledu výkonnosti, efektivnosti, škálovatelnosti a robustnosti virtualizačních nástrojů. Toto srovnání má pomoci firmě Red Hat rozhodnout, zda opustit XEN jako hlavní virtualizační nástroj v jejich distribucích a přejít na jiný novější, uživatelsky příjemnější virtualizační nástroj, jako je například KVM.

Klíčová slova

virtualizace, QEMU, KVM, XEN, VMware, XEN, VirtualBox, lmbench, ab, bonne++, netperf, plná virtualizace, paravirtualizace, hybridní virtualizace, hypervizor, chráněný režim, migrace

Abstract

The thesis tests and compares various existing virtualization tools for virtualization of computing platforms and explains important conceptions of virtualization technology. It's main contribution is a performance, efficiency, scalability and robustness comparison of virtualization tools. This comparison should help the Red Hat company to make a decision whether they should leave XEN as a main virtualization tool in their distributions and move on to another newer and more user friendly virtualization tool like KVM.

Keywords

virtualization, QEMU, KVM, XEN, VMware, XEN, VirtualBox, lmbench, ab, bonne++, netperf, full virtualization, paravirtualization, hybrid virtualization, hypervizor, protected mode, migration

Citace

Jiří Župka: Techniky virtualizace výpočetních platforem v Linuxu, bakalářské práce, Brno, FIT VUT v Brn, 2008

Techniky virtualizace výpočetních platforem v Linuxu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing.,Tomáše Vojnara, Ph.D.

.....
Jiří Župka
14. května 2008

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu doc. Ing,Tomáši Vojnarovi, Ph.D. za vedení mé práce. Dále bych chtěl poděkovat panu Ing. Radkovi Vokálovi, který mi zprostředkovával kontakt s firmou Red Hat.

© Jiří Župka, 2008.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informaních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovanch případů.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce se zaměřuje na oblast virtualizačních technologií. Cílem práce je shrnout, otestovat a vysvětlit dostupné virtualizační nástroje v prostředí operačního systému Linux.

1.1 Virtualizace

O co vlastně jde? Když se řekne slovo virtualizace, obecně si představujeme něco neexistujícího, co se vytváří uměle např. virtuální realita. V našem případě nejde o virtualizaci reality, nýbrž jen počítače, případně jen části nutné pro běh operačního systému.

1.1.1 Proč virtualizaci využívat?

Testování Vývojáři ocení možnost *otestovat* svůj software na jiném operačním systému bez nutnosti restartu fyzického počítače. Při vývoji jádra je díky virtuálnímu stroji možné ladit chyby, které by se na reálném hardwaru hledaly opravdu obtížně.

Efektivita Počítače a hlavně servery v dnešní době často nejsou využívány efektivně. Díky virtualizaci je možné na serverech rozjet několik nezávislých systémů, aniž by se navzájem podstatně ovlivňovaly, a využít tak výkon počítačů.

Bezpečnost Virtualizace nám přináší i rozsáhlé možnosti *zabezpečení*. Představme si situaci, kdy máte např. farmu webových serverů a chceme je zabezpečit proti softwarovému útoku. Stačí nám vytvořit virtuální server a několikrát ho spustit na fyzickém serveru a je to. Dojde-li k pádu jednoho virtuální serveru, nic se neděje, ostatní ho zastoupí. Mezitím se nějakým triviálním způsobem opraví zhroucený virtuální server a klient nic nepozná.

Přenositelnost Další podstatný přínos je *přenositelnost*. Vytvoříme počítač, který poskytuje služby v České republice, pošleme ho přes půl světa, kde ho spustí bez ohledu na nainstalovaný hardware a klienti až na [1] menší problém s DNS servery nic nepoznají.

[1] Při přesunu serveru přes celý svět do jiné sítě se musí zajistit, aby DNS servery, pod které spadá překlad DNS jména přesouvaného serveru, ukazovaly na počítač, kam se virtuální server přesouval.

1.1.2 Přínos práce

V rámci této práce jsou shrnuty základní pojmy týkající se virtualizačních technologií. Dále jsou zde podrobněji rozebrány vybrané virtualizační nástroje jako je QEMU, KVM, VMware, Xen a VirtualBox. Virtualizační nástroje byly porovnány podle kvality uživatelského rozhraní, škálovatelnosti a robustnosti definované na základě konzultace s odborníkem z firmy Red Hat. Pro uskutečnění testů, které jsou součástí zadání, bylo nutné vyvinout testovací nástroj, jehož popis je taktéž součástí této práce. Výsledky testů virtualizačních nástrojů, jejich vysvětlení a srovnání jsou uvedeny v poslední části práce.

1.1.3 Struktura práce

V druhé kapitole práce je uvedeno rozdělení typů virtualizačních technik a jsou zde vysvětleny pojmy důležité pro pochopení testů a virtualizace. Ve třetí části jsou rozebrány vybrané virtualizační nástroje na základě kritérií uvedených v zadání a v předchozí části kapitoly 1.1.2. Kapitola číslo čtyři se zabývá metodikou testování. Je zde uvedeno zamyšlení proč, co a jakým způsobem testovat. Dále je zde popsán program, který bylo nutné vyvinout, pro testování paralelního běhu virtualizovaných systémů. V poslední části čtvrté kapitoly jsou rozebrány konkrétní testovací nástroje. Předposlední kapitola číslo pět obsahuje porovnání vybraných virtualizačních nástrojů. V další části páté kapitoly je možné nalézt srovnání efektivity vybraného virtualizačního nástroje při spuštění virtualizovaného systému s různým počtem procesorů v smp (symmetric multiprocessing) režimu. V poslední části kapitoly pět je uvedeno srovnání efektivity spuštění více virtuálních systémů na jednom hostitelském systému. Poslední kapitola šest je závěr práce.

Kapitola 2

Virtualizační technologie

V této kapitole budou vysvětleny základní pojmy týkající se virtualizačních nástrojů a technologií.

2.1 Dělení virtualizačních technik

Dělení virtualizačních technik je částečně přejato z wikipedie [6] a doplněno o novinky. Některé vyjmenované virtualizační techniky je možné navzájem kombinovat.

Emulace Virtuální stroj simuluje celý hardware, dovoluje běh neupraveného OS hosta na zcela odlišném procesoru. Tento přístup je dlouho používán za účelem tvorby softwaru pro procesory, které nejsou fyzicky dostupné. Příklady zahrnují Bochs, PearPC (verze Virtual PC pro PowerPC, QEMU bez akcelerace a emulátor Hercula. Emulace je implementována širokou škálou technik od stavových automatů až po dynamickou rekompilaci na plně virtualizovaných platformách.

Plná virtualizace Virtuální stroj simuluje dostatečné množství hardwaru tak, aby umožnil oddělený běh neupraveného OS hosta určeného pro stejný druh CPU. Pro volání privilegovaných instrukcí, ke kterým nemá virtualizovaný systém přístup, je využita buď hardwarová virtualizace, nebo softwarové řešení v podání dynamické rekompilace. Oba tyto pojmy jsou popsány a vysvětleny níže v kapitole 2.2.3 a v kapitole 2.2.1. Obvykle je možný souběh více instancí. Tento přístup se objevil v roce 1966 u systému CP-40 a CP[-67]/CMS (předchůdce rodiny VM od IBM). Příklady zahrnují VirtualBox, Virtual Iron, Virtual PC, VMware Workstation, VMware Server (dříve znám jako GSX Server), VMware ESX Server, QEMU, Parallels Desktop, Adeos, Mac-On-Linux, Win4BSD, Win4Lin Pro a z/VM.

Paravirtualizace Virtuální stroj nemusí nezbytně simulovat hardware, ale místo toho (nebo navíc) nabízí zvláštní API, které může být použito jen z upraveného virtualizovaného systému. Toto systémové volání hypervizoru se nazývá „hypercall“ v Xenu, Parallel Worksations a Enomalism. Volání je implementováno hardwarovou instrukci DIAG („diagnose“) v CMS od IBM pod VM (kde se pojem „hypervizor“ poprvé objevil). Příklady zahrnují Win4lin 9x, logické domény od Sunu a z/VM.

Hybridní virtualizace Jde o spojení toho nejlepšího z obou metod plné virtualizace a paravirtualizace. Pomocí plné virtualizace je možné na virtuálním stoji spustit neupravený

systém. Pomocí paravirtualizace lze urychlit vybraný virtuální hardware, aby bylo docíleno větší výkonu a efektivity. Příkladem je KVM s rozhraním virtio.

Částečná virtualizace Virtuální stroj simuluje více instancí mnoha (ale ne všech) prostředí hardwaru, na kterém běží hostitel, především adresního prostoru. Takové prostředí podporuje sdílení zdrojů a izolaci procesů, ale neumí oddělit instance OS hostů. Ačkoliv obecně nelze hovořit o virtuálním stroji, jedná se o významný přístup z historického hlediska. Byl použit u systémů CTSS, pokusného IBM M44/44X a zřejmě i u VMS. (Mnoho dalších systému, jako Microsoft Windows nebo Linux a systémy ze zbývajících kategorií níže, používají tuto techniku.)

Virtualizace na úrovni operačního systému Virtualizuje se fyzický server na úrovni OS, což umožňuje běh více izolovaných bezpečných virtuálních serverů na jednom fyzickém serveru. Prostředí OS hosta sdílejí jeden OS s hostitelským systémem – tj. stejné jádro OS je použito pro implementaci OS hosta. Aplikace běžící v prostředí hosta jej však vnímají jako samostatný systém. Mezi příklady patří Linux-VServer, Virtuozzo (pro Linux nebo Windows), OpenVZ, kontejnery Solarisu a FreeBSD Jail.

Aplikační virtualizace Desktopové nebo serverové aplikace běžící na daném stroji, používají místní zdroje, ale běží ve zvláštním virtuálním stroji. To je rozdíl oproti tradičnímu lokálnímu běhu nativních aplikací, tj. softwaru nainstalovaném přímo na systému. Taková aplikace běží v malém virtuálním prostředí obsahujícím komponenty nutné ke spuštění např. položky registrů, soubory, proměnné prostředí, prvky uživatelského rozhraní a globální objekty. Toto virtuální prostředí se chová jako vrstva mezi aplikací a operačním systémem, která zabraňuje konfliktům mezi aplikací a OS nebo mezi aplikacemi vzájemně. Příklady zahrnují Java Virtual Machine od Sunu Softricity, Thinstall, Altiris, Portable Apps a Trigence. (Tento druh virtualizace je zřetelně odlišný od všech předešlých. Dělí je pouze tenká hranice od virtuálních prostředí jako je Smalltalk, Forth, Tcl, P-kód a další interpretované jazyky.)

Hardwarová virtualizace Hardware zajišťuje podporu pro izolovaný běh virtualizovaných systémů. Procesory firem Intel (Intel VT) a AMD (AMD-V) mají implementovánu hardwarovou podporu virtualizace. Mezi virtualizační nástroje využívající tuto podporu patří KVM, VMware, Xen, VirtualBox.

2.1.1 Podpora virtualizace v CPU

U procesoru s architekturou x86 se počátky podpory plné virtualizace datují asi od roku 2005 a 2006, kdy intel a pak i AMD uvedli na trh procesory s rozšířením IVT (Intel) a AMD-V (AMD). V linuxu si můžeme jejich přítomnost zjistit v souboru */proc/cpuinfo*. V těle souboru v sekci flags je identifikováno jako vmx pro Intel a svm u procesoru od AMD.

Virtualizaci musí podporovat i základní deska, na které je procesor provozován. V dnešní době by již každá deska, která obsahuje chipsety vyvinuté v posledních několika letech, neměla mít s podporou virtualizace problém.

Procesory podporující virtualizaci řady x86

Procesory AMD:

- *Athlon 64* stepping F a G
- *Athlon 64 X2* stepping F a G
- *Turion 64* Richmond
- *Turion 64 X2* všechny
- *Opteron* Santa Ana a novější
- *Phenom* Zavádí Nested Paging pro rychlejší přepínání kontextu virtuálních strojů

Procesory Intel:

- *Pentium 4* modely 662 a 672
- *Core Duo* až na modely T2300E a T2050/T2150/T2250
- *Core 2 Duo* až na T52x0,T530,T54x0,T5500,E2xx0,E4x00,E8190

2.2 Další důležité pojmy

V této části jsou popsány klíčové pojmy, které umožní získat nad problémem virtualizace větší nadhled a usnadní pochopení testů virtualizačních nástrojů.

2.2.1 Metody emulace

Dynamická rekompilace jedná se o metodu virtualizačních emulátorů, díky níž jsou tyto emulátory schopny spouštět programy určené pro jinou architekturu. Jak už napovídá název, jde o překlad za chodu, kdy pomocí disassembleru analyzuje kód spouštěného programu a instrukce, které neumí spustit nebo jsou ve virtuálním prostředí zakázány, převede na instrukce patřící do architektury hostitelského procesoru. Kvůli náročnosti dynamické rekompilace se často virtualizační nástroje snaží spouštět kód přímo na hostitelském procesoru. Výsledek dynamické rekompilace uveden na obrázku 2.1.

```
beginning:
    mov A,[first string pointer]    ; Put location of first character of source string
                                    ; in register A
    mov B,[second string pointer]   ; Put location of first character of destination string
                                    ; in register B
loop:
    mov C,[A]             ; Copy byte at address in register A to register C
    mov [B],C             ; Copy byte in register C to the address in register B
    cmp C,#0              ; Compare the data we just copied to 0 (string end marker)
    inc A                 ; Increment the address in register A to point to
                          ; the next byte
    inc B                 ; Increment the address in register B to point to
                          ; the next byte
    jnz loop              ; If it wasn't 0 then we have more to copy, so go back
                          ; and copy the next byte
end:                      ; If we didn't loop then we must have finished,
                          ; so carry on with something else.
```

Obrázek 2.1: Výsledek po užití disassembleru a dynamické rekompilaci

Adaptivní optimalizace Tato technika souvisí s pojmem dynamické rekompilace, kdy se při rekompilaci ještě optimalizuje kód, aby se vykonával efektivněji. O čemž se můžeme přesvědčit na obrázku 2.2.

```
beginning:
    mov A,[first string pointer]    ; Put location of first character of source string
                                    ; in register A
    mov B,[second string pointer]   ; Put location of first character of destination string
                                    ; in register B
loop:
    movs [B],[A]          ; Copy 16 bytes at address in register A to address
                          ; in register B, then increment A and B by 16
    jnz loop              ; If the zero flag isn't set then we haven't reached
                          ; the end of the string, so go back and copy some more.
end:                      ; If we didn't loop then we must have finished,
                          ; so carry on with something else.
```

Obrázek 2.2: Výsledek po adaptivní optimalizaci

2.2.2 Hypervizor

Hypervizor je něco jako minikernel, kerý zajištuje IO operace přidělování paměti a veškerou komunikaci hostů (virtuálních strojů u xenu včetně hlavního kernelu Dom0).

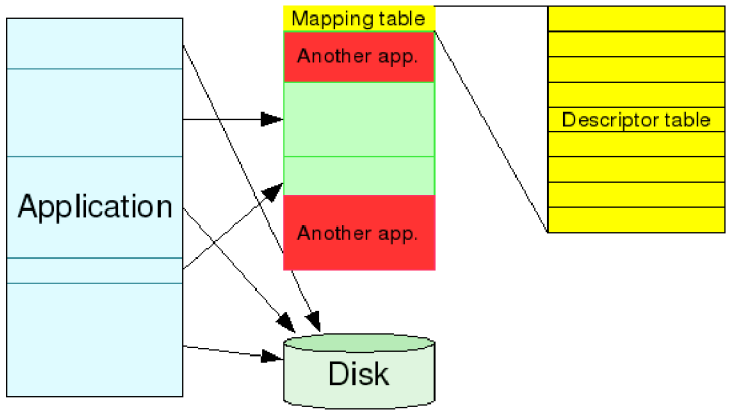
2.2.3 Chráněný režim

Chráněný režim je operační mód procesorů řady x86 a kompatibilních. V tomto režimu si každý proces myslí, že může přistupovat k celé operační paměti, u 32b v chráněném režimu systémů 4GB, aniž by byl omezován jiným procesem. Aby byla možnost kontrolovat procesy a zamezit jim v přímém přístupu k hardwaru či spouštění instrukcí, které by ohrožovaly chod systému, byly zavedeny úrovně oprávnění, tvz. *ringy*.

Správa paměti

Správa paměti je zajištěna pomocí virtuální paměti (*virtual memory*), proto se chráněný režim nazývá taktéž Protected virtual address mode. Systém zajišťuje mapování virtuální paměti do fyzické paměti. Aby mapování virtuální paměti na fyzickou fungovalo dost rychle, je virtuální paměť a fyzická paměť rozdělena na stejně velké stránky (segmenty o velikosti

4KB nebo větší v závislosti na konkrétní architektuře a systému). Tabulka namapovaných stránek složená z deskriptorů se ukládá na začátku fyzické operační paměti. [1] Aby byla iluze pro spouštěné programy dokonalá toto nestačilo. Z důvodu omezení velikosti fyzické paměti přibylo ukládání části paměti na disk tzv. Stránkování(paging). Jestliže je fyzická paměť plná, stránkování zajistí uvolnění části fyzické paměti uložením na pevný disk. Výběr stránek, které se přesunou na disk se často řeší pomocí algoritmu *LRU* (last recently use). V případě, že jsou požadována data, která jsou obsažena ve stránce uložené na disku, je vyvolána výjimka. Při této výjimce systém zajistí načtení stránky z disku. Jestliže již ve fyzické paměti není dostatek místa, uloží se jiná stránka na disk a načte se požadovaná stránka z disku do paměti. Celé schéma adresace je popsáno na obrázku 2.3.



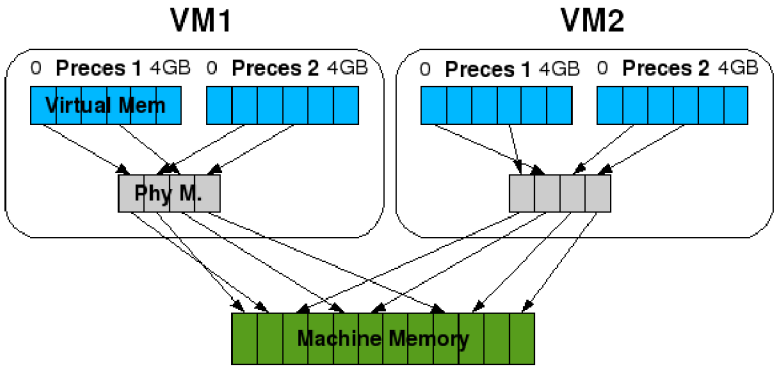
Obrázek 2.3: Adresace paměti v chráněném režimu

Aby se překlad virtuálních adres na fyzické zrychlil, vznikla nová hardwarová jednotka v procesoru *MMU* (memory management unit), která zajišťuje mapování. Pro další urychlení vzniklo *TLB* (Translation Lookaside Buffer) tabulka deskriptorů uložená přímo v procesoru, kde jsou uloženy deskriptory na nejvíce používané stránky.

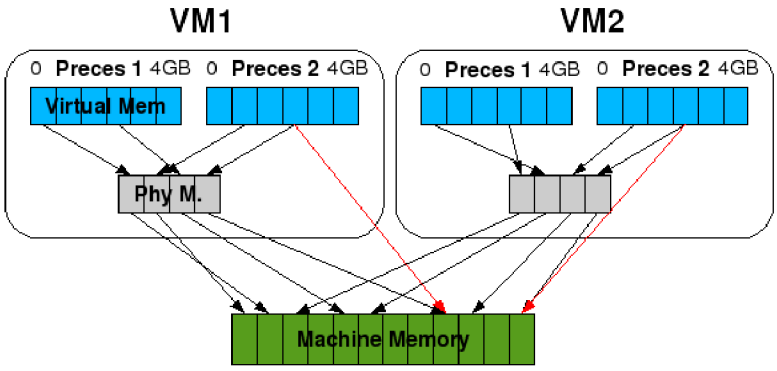
Jenže pro virtualizaci toho nestačí. Bylo nutné vytvořit ještě jednu vrstvu, která by virtualizovala fyzickou paměť. Posloupnost adresování je popsána na obrázku 2.4. Paměť do které se virtualizuje fyzická paměť je možné nazvat Strojová paměť (*Machine Memory*).

Toto řešení bohužel přináší dvoji překlad adres, což znamená ztrátu výkonu. Z tohoto důvodu se zavedl *shadow page tables*. Princip shadow page tables spočívá ve vytvoření nové tabulky, která obsahuje mapovaní přímo z Virtuální paměti do Strojové paměti. Tímto způsobem je možné se vyhnout jednomu překladu z fyzické na strojovou a znovu skoro dosáhnout výkonu obyčejného chráněného režimu. Jak to funguje je zobrazeno na obrázku 2.5.

[1] Deskriptor je něco jako ukazatel obohaceny o další parametry,které jsou nutně v chráněném módu.



Obrázek 2.4: Adresace paměti v chráněném režimu s podporou virtualizace



Obrázek 2.5: Adresace paměti v chráněném režimu s shadow page tables

Nevýhody shadow page tables:

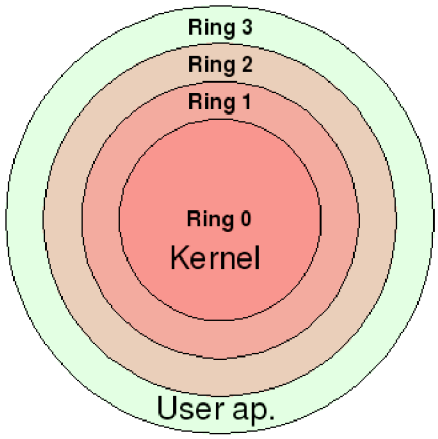
- nutnost synchronizovat záznamy v primární mapovací tabulce a v shadow page tables,
- virtualizační software se musí starat o správu shadow page tables,
- u vSMP (virtual symmetric multiprocessing) je nutné udržovat shadow page table pro každý vCPU (virtual CPU),
- problém přesouvání procesů mezi vSMP zahrnuje i nutnost přesunout shadow page tables.

Tyto nevýhody se snaží řešit hardwarová podpora u procesorů AMD Barcelona v podobě NPT (*Nested/Extended Page Tables*). MMU převádí Virtuální adresu na fyzickou a fyzickou adresu na strojovou při plnění TLB tabulky. Pro zvýšení výkonnosti NPT je dobré používat velké stránku (large/huge page). Zmenší se tím počet uložených stránek v TLB. Nevýhoda velkých stránek je menší efektivita využití paměťového prostoru.

Oprávnění přístupu

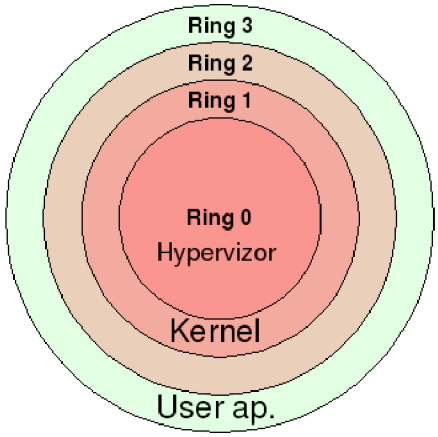
Oprávnění přístupu se v chráněném režimu dělí do 4 úrovní tzv. Ring 0 - 3, které zobrazuje obrázek 2.6. Ring s menším pořadovým číslem má vždy větší oprávnění. To znamená že

ring 0 má nejvyšší oprávnění. Úroveň oprávnění určuje jaké zdroje (paměť, disky a jiný hardware) a do jaké úrovně bude moci proces přistupovat. Pro přístup do úrovně s vyšším oprávněním se používá speciální *brány* nebo *volání*.



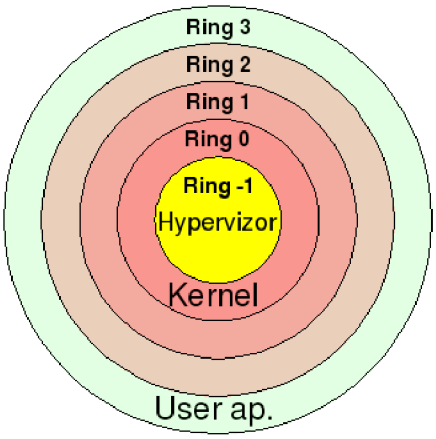
Obrázek 2.6: Standardní využití úrovní oprávnění

Bohužel systémy často nevyužívají všechny úrovně oprávnění, jak je vidět na obrázku. Téhle vlastnosti operačních systémů využívá paravirtualizace. U paravirtualizace se kernel přesune o úroveň výš do často volného ringu 1. Hypervizor se zavede místo původního kernelu do ringu 0. Tato situace je graficky znázorněna na obrázku 2.7. Tímto je zajištěno, že jádro nebude mít přímí přístup k hardwaru. Bohužel kernel není (nebyl) standardně uzpůsoben pro běh v ringu 1. Proto se musí jádro upravit, aby bylo schopno volat hypervizor pro přístup k hardwaru.



Obrázek 2.7: Využití úrovní oprávnění při paravirtualizaci

Pro plnou virtualizaci je potřeba rozšíření procesoru např. AMD-V a Intel VT. Tyto rozšíření vytvoří Ring -1, ve kterém se spustí hypervizor a nad tímto hypervizorem je možné spustit neupravený systém, což je ilustrováno na obrázku 2.8.



Obrázek 2.8: Využití úrovní oprávnění při plné virtualizaci

2.2.4 Škálovatelnost

Pojem škálovatelnost se rozumí schopnost virtualizačního nástroje reagovat na vzrůstající nároky virtuálního stroje. Například možnosti v migraci na jiné fyzické stroje, případně jiné virtualizační nástroje, přidávání paměti a procesorů za chodu atd.

2.2.5 Robustnost

Jde o zhodnocení celého řešení ze systémového pohledu. Jak se daný virtualizační nástroj chová, jaké jsou jeho možnosti nasazení v reálném prostředí. Tato definice robustnosti byla sestavena na základě konzultace s odborníkem na virtualizaci z firmy Red Hat.

2.2.6 Migrace

Migrace je přenos aktuálního stavu virtualizovaného systému z jednoho virtuálního stroje na jiný. Stav virtuálního systému je obsah operační paměti plus nějaké drobnosti potřebné pro správnou inicializaci hardwaru virtuálního stroje, na který se virtuální systém migruje. Migrace se dělí na skupinu *offline* a skupinu *live migration*.

Offline

Při Offline migraci se přenášený systém zastaví, jeho se stav uloží na disk nebo jiným způsobem přesune do nového virtuálního stroje. Po ukončení přenosu, nebo načtení stavu virtuálního systému, z disku se nový virtuální stroj spustí. Nevýhoda tohoto způsobu migrace je nutnost odstavit virtualizovaný systém. Zástupcem offline migrace je KVM.

Live migration

Live migration je takzvaná migrace za života (za plného vědomí). Jestliže virtualizační nástroj podporuje tuto vymoženost, je možné za chodu přenést běžící systém, aniž by připojený klient něco poznal (například u webového serveru). Například KVM umožňuje live migration. Migrace je řešena pomocí přenosu paměti přes síť pomocí protokol TCP a jiné. Migrovaný virtuální stroj musí mít identický hardware jako cílový stroj.

Uveďme nyní příklad postupu migrace u KVM, členěny po jednotlivých krocích migrace:

1. Nastavení
    - Je nutné nastartovat systém, na který se má provést migrace, poté zapnout logování dirty page (neboli stránky paměti změněné od poslední synchronizace) a další.
2. Přesun paměťi
    - Zdrojový virtualizovaný systém je stále v provozu a klienti mohou přistupovat.
    - Nastaví se limitace přenosu dat.
    - Při prvním přesunu se přenese celá paměť virtuálního stroje.
    - Přesouvaný virtuální stroj loguje dirty page. v jeho systému a přesouvá změny do cílového virtuálního systému.
3. Zastavení přesouvaného systému
    - Při zastavení se uloží všechny rozpracované soubory na pevný disk, aby cílový systém měl přístupná všechna potřebná data.
4. Přesun stavu virtuálního stroje
    - Přesun probíhá bez limitace rychlosti.
    - Přesouvají se stavy virtuálního hardwaru a poslední dirty pages.
5. Spuštění cílového virtuálního systému
    - Cílový systém pomocí síťového rozhraní broadcastem oznámí, kde se nachází.

2.2.7 Balloon driver

Pomocí balloon driveru je možné za chodu virtuálního stroje regulovat jeho zabranou fyzickou paměť virtuálního systému. Přidávání paměti pomocí balloon driveru je hojně používaná metoda. Většina níže popisovaných virtualizačních nástrojů tuto možnost podporuje. Driver se jmenuje balloon, protože se často přirovnává k balónu (operační paměti hostitelského systému) a vzduchu v něm (zabrané operační paměti virtualizovaných systémů). Při přetlaku je ho nutné vyfouknout a při podtlaku do sebe nechá vstoupit vzduch bez odporu.

Metoda spočívá v ovládání paměťového managementu virtualizovaných počítačů, aby bylo možné co možná nejefektivněji využít operační paměť hostitelského systému, a tím urychlit běh všech virtualizovaných klientů. Do klientů se nainstaluje tzv. balloon driver, který vytvoří spojení mezi hostitelským systémem a virtualizovanými systémy. Tento kanál umožňuje hostitelskému systému poslat virtualizovaným systémům informaci o zaplnění paměti. V případě, že je paměť prázdná, klientům je dovoleno načíst do paměti swapované stránky, ale v případě přeplnění paměti, hostitelský systém klienty donutí odswapovat nepotřebná data na klientský swapovací disk.

Kapitola 3

Testované virtualizační nástroje

V této kapitole budou představeny testované virtualizační nástroje. U každého shrneme použité virtualizační metody, robustnost, škálovatelnost a možnosti uživatelského rozhraní. Výkonnost virtualizačních nástrojů bude uvedena v kapitole 5, nazvané Testy virtualizačních nástrojů.

1. QEMU
2. KVM
3. Xen
4. VMware
5. VirtualBox

3.1 QEMU

QEMU [3] patří do rodiny emulačních virtualizačních nástrojů.

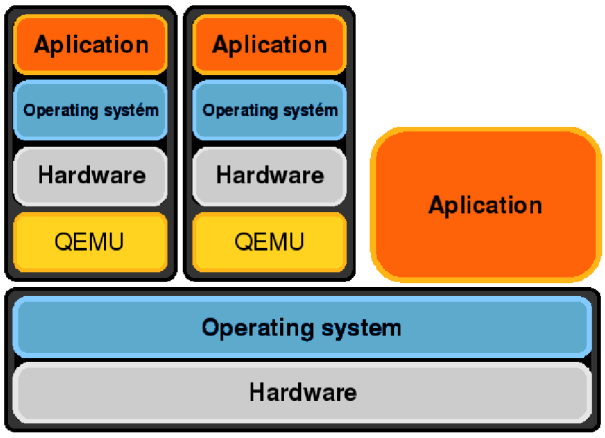
3.1.1 Virtualizační technologie

V porovnání s ostatními emulačními nástroji je QEMU rychlý. Pracuje na principu dynamického překladu instrukcí. Proto je jednoduše portovatelný na novou hostitelskou platformu a jednoduše se do něj implementuje podpora pro nové procesory (architektury).

QEMU pracuje ve dvou režimech:

1. Plná emulace systému. V tomto režimu QEMU plně emuluje celý systém např. PC, včetně jednoho nebo více procesorů a dalších potřebných periférií.
2. Uživatelský mód emulace. V tomto režimu QEMU spustí proces kompilovaný pro jednu architekturu CPU na jiné architektuře CPU. Je možné ho použít pro spuštění Windows API ve Wine a jiné.

Z obrázku 3.1 je zřejmé, QEMU že běží čistě v uživatelském režimu. Z toho plyne, že nepotřebuje žádné jaderné ovladače. Díky těmto vlastnostem je vhodný pro testování softwaru pro nedostupné procesory. Nevýhoda tohoto přístupu plyne z jeho pomalosti.



Obrázek 3.1: Zařazení QEMU do systému

Dosahuje průměrně 10%-20% výkonu nativního stroje. Proto je pro virtualizaci serverů takřka nepoužitelný.

Tuto nevýhodu se snaží odstranit jaderné moduly *kqemu* a *QVM86*, ty umožňují instrukce odpovídajícího virtuálního CPU spouštět přímo na hostitelském CPU např. x86 guest na x86 host. Kqemu je využit ve Win4Lin a QVM86 je užíván ve VirtualBoxu. Výkon kqemu a QVM86 se blíží výkonu reálného počítače.

3.1.2 Uživatelské rozhraní

Co se týče uživatelského rozhraní se QEMU řadí spíše ke klasickým linuxovým programům, kde je ovládání řešeno pomocí příkazové řádky a grafické rozšíření je necháno na nějaký frontend např. Qemu Launcher. O grafický výstup virtuálního počítače se stará buď VNC (Virtual Network Computing), nebo std-vga importovaná z Bochs. [1] VNC rozhraní se dá s výhodou použít zobrazení grafické plochy vzdáleného serveru.

3.1.3 Škálovatelnost

Škálovatelnost čistého QEMU bez akcelerací nemá význam rozebírat, protože QEMU je pomalé. I přesto QEMU umožňuje migraci. Podporuje jak offline tak live migraci. Upravené QEMU rozhraní používá KVM. Migrace je blíže popsána v odstavci migrace 2.2.6.

3.1.4 Robustnost

Qemu je spíše emulátor než virtualizační nástroj, proto je příliš pomalý na nasazení v provozním prostředí.

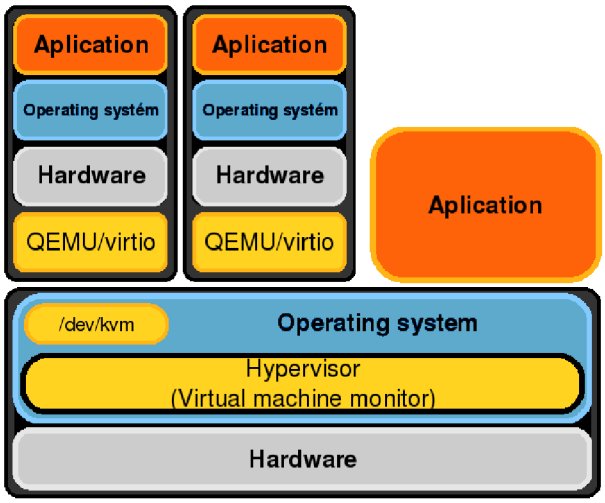
[1] Neumožňuje přístup k virtuální stroji pomocí vzdálené plochy či přeposílání X. Naproti tomu zajišťuje větší rozlišení.

3.2 KVM – Kernel Virtual Machine

Již z názvu vyplývá, že jde o virtualizaci implementovanou přímo v kernelu. Plné virtualizace dosahuje pomocí hardwarové podpory AMD-V nebo INTEL-VT. KVM využívá jako rozhraní jemně upravený QEMU emulátor.

3.2.1 Virtualizační technologie

Úprava spočívá v možnosti připojit QEMU emulátor na KVM rozhraní kernelu (jádra systému). Výhodou KVM vůči QEMU je rychlost a oproti paravirtualizaci umožňuje běh nemodifikovaných systémů Linux,Windows atd. Začlenění KVM do systému je zobrazeno na schématu 3.2.



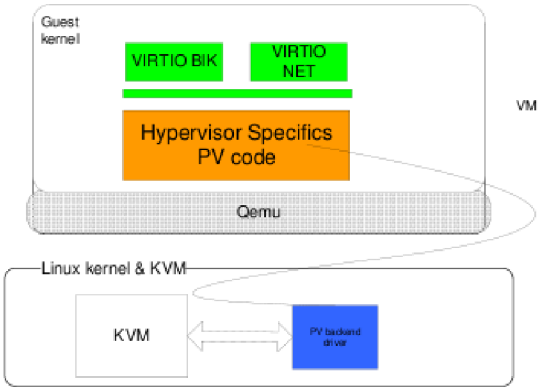
Obrázek 3.2: Zařazení KVM do systému

Obecně platí, že emulace hardwaru není vždy to pravé. Důsledkem emulace je často velká ztráta výkonu. Proto KVM umožňuje i paravirtualizaci hardwaru pomocí *virtio* popsaného na obrázku 3.3. Virtio jsou *frontend* moduly zabudované v jádře hosta, pomoci nichž se naváže přímo na *backend* moduly hardwaru v jádře hostitele a tím značně urychlí práci hardwaru. Nevýhoda tohoto přístupu spočívá v nutnosti nainstalovat do virtualizovaného systému správné ovladače.

Díky paravirtualizaci hardwaru se rychlost síťového přenosu z emulovaných 361Mb/s vzroste na 805Mb/s paravirtualizovaných a zpoždění klesne z 800μs na 300μs, což je dobrý výsledek.

3.2.2 Uživatelské rozhraní

Virtualizační nástroj KVM lze ovládat z příkazové řádky přes upravené QEMU rozhraní, nebo grafické rozhraní virtManageru. VirtManager je grafické rozhraní nad knihovnou libvirt, která zajišťuje napojení grafického rozhraní a ovládání KVM. Qemu má vůči virtManageru výhodu detailnějšího nastavení. KVM spuštěné pomocí qemu přes sdl rozhraní, umožňuje přepnutí do konzoly ovládající virtuální stroj. Ve virtuální konzole je možné přistupovat k low-end nastavením virtuálního stroje, od zastavení přes přidávání hardware za chodu až po live migraci.



Obrázek 3.3: Způsob paravirtualizace zařízení v KVM. [1]

3.2.3 Škálovatelnost

Co se týče škálovatelnosti je na tom virtualizační nástroj KVM dobře. Podporuje snapshoty, offline i live migraci, přidávání a odebírání zařízení za chodu a jiné. V novějších verzích, přesněji od verze 62, umí KVM za chodu přidávat či odebírat procesory přidělené virtuálnímu systému. Přidávání paměti a odebírání paměti je řešeno pomocí balloon driver, který je popsán v kapitole 2.2.7. Postup migrace je popsán u definice pojmu migrace 2.2.6

3.2.4 Robustnost

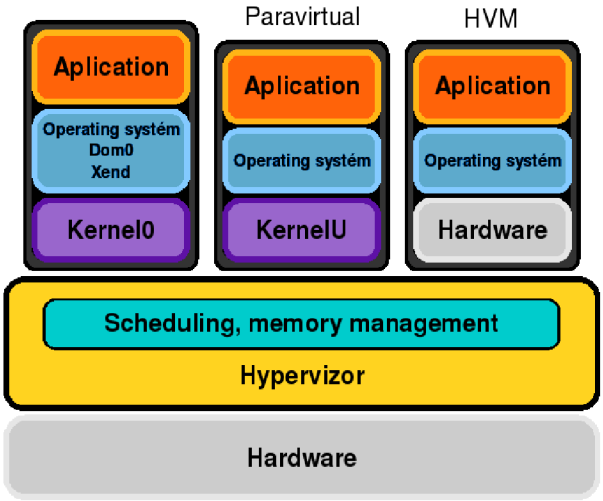
Z pohledu robustnosti si KVM také nevede špatně. Sice jeho uživatelské rozhraní přívětivostí neoplývá, ale pro zběhlé linuxové uživatele to nebude představovat žádný problém. KVM je integrováno přímo do každého novějšího kernelu (od verze 2.6.20), takže jeho použití je bezproblémové. Na rozdíl od Xenu takřka žádným způsobem neovlivňuje běh normálního operačního systému, což je hlavně pro obyčejné uživatele obrovská výhoda. Grafické rozhraní KVM lze tunelovat přes SSH X nebo přímo pomocí VNC. KVM je z pohledu uživatele velice příjemný virtualizační nástroj, je zadarmo a jak již bylo řečeno takřka neovlivňuje chod hostitelského systému. Pro firemní využití je tento nástroj asi ještě moc mladý, i když s trochou dobré vůle by mohl nahradit téměř jakýkoliv z testovaných virtualizačních nástrojů.

3.3 Xen

Jedná se o virtualizační nástroj založený na paravirtualizaci, díky čemuž získal velký výkon za cenu úpravy virtualizovaného systému.

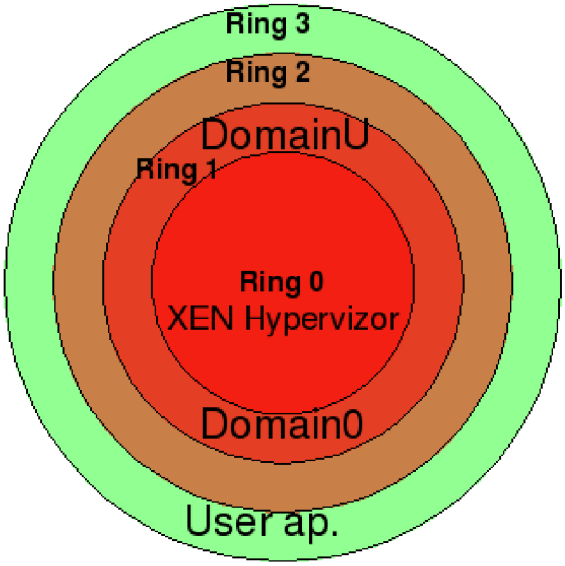
3.3.1 Virtualizační technologie

S příchodem podpory virtualizace pomocí instrukcí procesoru i Xen umožňuje plnou virtualizaci. Diky tomuto rozšíření je možné na Xenu rozjet i Windows a jiné systémy bez nutnosti úpravy operačního systému. Začlenění Xenu do operačního systému je zobrazeno na obrázku 3.4



Obrázek 3.4: Zařazení virtualizačního nástroje Xen do systému

V paravirtualizačním režimu se hypervizor zavede do tvz. ringu 0, kde by měl být normálně zaveden kernel. Kernel systému se pak zavede do vyššího ringu 1. Díky tomuto se kernel musí pro přístup k hardwaru dotazovat hypervizor. Protože, klasický kernel není na dotazy vybaven musí se předem upravit. Kernel a pod ním spuštěné aplikace se nazývají domain. Grafický popis paravirtualizace v podání Xenu je zobrazen na obrázku 3.5



Obrázek 3.5: Umístění Xenu ve vrstvách chráněného režimu při paravirtualizaci

V plně virtualizovaném režimu, kdy je nutné zajistit běh closesource klientu např. Windows atd., se využije hardwarové rozšíření AMD-V či Intel VT. Rozšíření vytváří tzv. *ring -1*. V tomto ringu se spustí hypervizor, který může jednoduše odchytávat požadavky nemodifikovaných systémů. Nemodifikované systémy běží klasicky od ringu 0 do ringu 3,

jak je zobrazeno na obrázku 2.8

Domain0 je hlavní doména, která jako jediná má přístup přímo k fyzickému zařízení i přesto musí být upravená. V domain0 se navíc spouští xen daemon (xend). Tento démon umožňuje ovládání samotného xenu (hypervizoru) např. nastavování množství paměti přidělené klientům aj.

Ostatní domény DomainU jsou virtuální počítače. Od verze 3.0 Xen umožňuje i plnou virtualizaci pomocí virtualizace hardwaru převzaté z QEMU. Pro plnou virtualizaci je nutné mít podporu pomocí procesoru.

3.3.2 Uživatelské rozhraní

Xen má stejně jako KVM hlavní rozhraní realizováno pomocí příkazové řádky a virtManageru popřípadě dalších rozhraní. Vzdálený přístup k virtuálním systémům je realizován pomocí VNC, bohužel ne v příliš povedené formě. Po instalaci systému je výhodnější přejít na vzdálenou správu realizovanou pomocí nainstalovaného systému.

3.3.3 Škálovatelnost

V oblasti škálovatelnosti se Xen opravdu nenechá zahanbit. Umožňuje migraci offline i live migraci. V případě nutnosti je bez problémů možné měnit počet přidělených procesorů i velikost přidělené paměti. Vytváření a práce se snapshoty je skoro samozřejmostí.

3.3.4 Robustnost

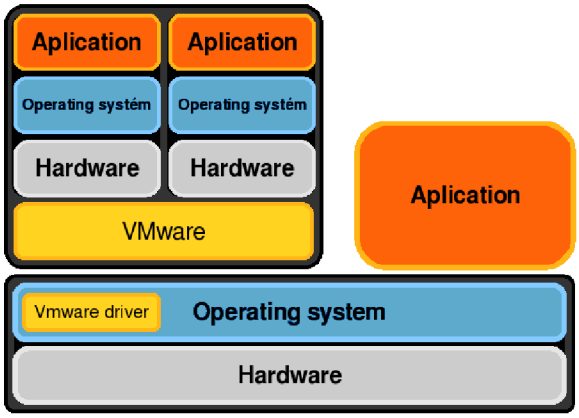
V případě že nevlastníme nějakou enterprise verzi je Xen stejně jako KVM uživatelsky méně přívětivý. Podstatná nevýhoda Xenu je v nutnosti instalovat a spouštět i hostitelský systém pod hypervizorem. To přináší řadu nevýhod i příjemných vlastností. Mezi základní nevýhody patří častá nekompatibilita ovladačů hardwaru s Xen domain0. Protože ovladče Xenu typu domain0 nejsou přímo obsaženy v kernelu, je nutné pro hostitelský systém používat pouze kernel pro který byl vydán patch. Výhodou ovlivnění celého hostitelského systému je možnost zamaskovat před reálným systémem fyzické zařízení a poté ho přímo namapovat do virtuálního systému. Pro nasazení do ostrého provozu je Xen vybaven vším potřebným, a proto je ho možné doporučit hlavně pro firemního aplikace. Pro normální uživatelé je Xen spíše přítěží než užitečný virtualizační nástroj, a to hlavně kvůli ovlivnění hostitelského systému.

3.4 VMware

Komerční sada virtualizačních nástrojů využívajících pro svůj běh dynamický překlad a přímé spouštění kódu. Zahrnuje v sobě nástroje zaměřené pro vývojáře, IT profesionály a běžné uživatele VMware Workstation, tak i pro servery VMware ESX Server. Další informace je možné nalézt na stránkách firmy VMware [5].

3.4.1 Virtualizační technologie

VMware Workstation zajišťuje jak plnou virtualizaci, tak experimentální podporu paravirtualizace pro linuxové systémy. Po instalaci VMware tools je možné užívat interakci mezi hostitelským systémem a virtuálním strojem. Například copy/paste. Začlenění do systému je popsané na obrázku 3.6.



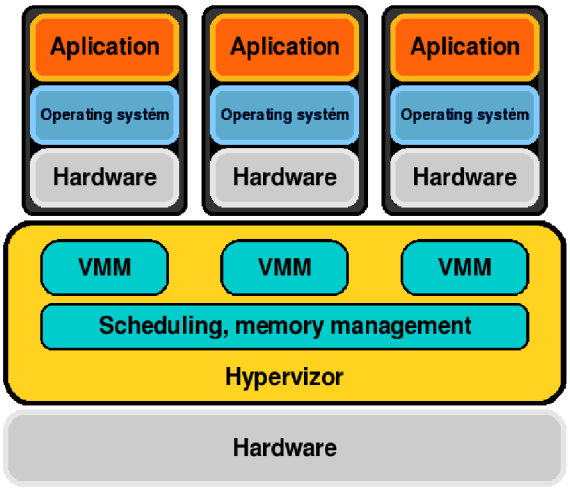
Obrázek 3.6: VMware workstation, server, player

VMware ESX Server běží jako Bare-metal architektura, proto nepotřebuje pro svůj běh hostitelský operační systém, protože VMware ESX Server je operačním systémem. Server obsahuje hypervizor, který zajišťuje vysoký výkon virtuálních strojů. Pro jeho instalaci je nutné mít procesor, který podporuje plnou virtualizaci. Začlenění do systému je znázorněno na obrázku 3.7

VMware nabízí i dva volně použitelné nástroje VMware Server a VMware Player. Volné nástroje jsou sice volně ke stažení, ale je to vykoupeno menším výkonem hlavně v IO operacích s diskem, sítí atd. I přesto jsou velmi kvalitně zpracované a na základní práci a testování postačují.

3.4.2 Uživatelské rozhraní

Uživatelské rozhraní je na rozdíl od předešlých nástrojů takřka dokonalé. Jen pomocí klikání myši je možné udělat skoro cokoliv co virtualizační nástroj umožňuje, ať už se jedná o VMware Server, Player nebo Workstation zaměřené na uživatele, nebo VMware ESX Server pro firmy. Dále podporuje rozšíření pracovní plochy virtuálního stroje o další obrazovku.



Obrázek 3.7: VMware ESX server

3.4.3 Škálovatelnost

Pokud uvažujeme komerční verze, jakou je např. VMware ESX Server, je na tom VMware se škálovatelností výborně. Pro migrování má speciální nástroj zvaný VMotion, který umí stejně jako u jiných virtualizačních nástrojů migrovat spuštěný virtuální systém on live. VMotion zajišťuje migrování virtuálního stoje v případě, že pevné disky virtualizovaného systému jsou přístupny ve všech strojích účastnících se migrace. Pro migraci disku má VMware další nástroj nazvaný VMFS (Virtual Machine File System), který takřka za běhu migruje pevné disky. Další v tomto případě jsou snapshoty, které umí VMware přes pěkné grafické menu spravovat. Přidávání a odebírání hardwaru za běhu je u VMwaru problém aspoň v nástrojích zaměřených na uživatele (VMware Workstation), či volně dostupných nástrojích (VMware Server). Jediné, co je možné měnit za běhu virtualizovaného systému (v VMware Wokrstation a Server) je připojení k vnější síti a výměnná zařízení.

3.4.4 Robustnost

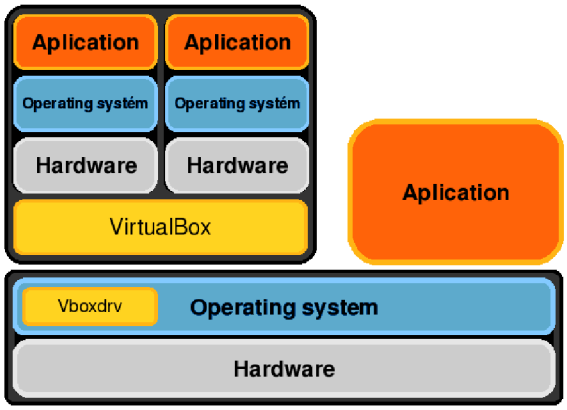
VMware se dá považovat za velice robustní a propracovaný nástroj. Pokud máte dostatek peněz není problémem téměř žádná administrativní operace, jakou si vymyslíte. Pokud hovoříme o VMwaru Workstation, nebo free verzích, je instalace jednoduchá. Jediný problém nastane v případě, že máte novější kernel, do kterého není možné přiložené moduly nainstlovat. Tento problém řeší patch třetí strany. Po instalaci téměř nepoznáte, že se systém změnil. VMware si přidá několik driverů a většinou podle voleb instalace vytvoří bridge a virtuální síťovou kartu, přes kterou připojuje virtualizované systémy do hostitelského systému. Vzhledem k uvedeným skutečnostem je možné VMware doporučit jak pro firemní, tak soukromé užití. Ani jeden z nástrojů by neměl zákazníka zklamat. Jediný problém je cena produktů od VMware. VMware sice nabízí free verze, ale ty jsou výkonnostně limitovány, jak bude ukázáno na testech.

3.5 VirtualBox

Virtualizační nástroj s uživatelsky příjemným rozhraním, zaměřený na běžné uživatele.

3.5.1 Virtualizační technologie

Virtualbox je založený na QEMU. Pro získání rychlosti vychází z upraveného jaderného modulu QVM86. Snaží se spouštět všechen kód přímo na fyzickém procesoru a rekompilaci využívá jen ve výjimečných situacích. Avšak rekompilace není největší problém, tím je množství chyb generovaných např. při přístupu k IO z ring 3 nebo spouštění ring 0 aplikací (kernel) v ring 1, kvůli privilegovaným instrukcím, které se v ringu 1 nedají spustit. V těchto případech musí přijít na řadu emulace. K odstranění těchto problémů se používá „Patch Manager“ (PATM) a „Code Scanning and Analysis Manager“ (CSAM). Před spuštěním kódu standardně běžícího v ring 0 se nejprve provede rekurzivní scan k vyhledání problematických instrukcí a poté se nahradí in-situ(bez potřeby další paměti) skokem na hypervizor, kde je připraven kód, který již chybu nevyvolá. Při každé chybě se zkouší vytvořit patch, aby se již při dalším volání části kódu chyba neopakovala. Začlenění do systému je realizováno způsobem popsaným na obrázku 3.8



Obrázek 3.8: VirtualBox

3.5.2 Uživatelské rozhraní

Virtualbox má pěkné a jednoduché klikací uživatelské rozhraní. I naprostý začátečník si spustí svůj virtuální systém. Protože je virtualBox mířen na uživatelské použití, přichází v tomto ohledu se zajímavými vylepšeními, jako je například bezešvý mód. V bezešvém módu se vše, až na pozadí pracovní plochy virtualizovaného systému, zobrazí na plochu hostitelského systému. To umožní mít na jednom monitoru spuštěné dva plně funkční systémy bez toho, aby se výrazně ovlivňovaly. Doporučuji vyzkoušet. Virtualbox má i ovládání příkazovou řádkou, pomocí níž jde dělat řada věcí, které by v grafického rozhraní nebyly vůbec možné. Pro spouštění virtuálního počítače z jiného počítače je využíván *RDP* (Remote desktop protokol). RDP protokol je podporován ve Windows, Linuxech (rdesktop) a jiných systémech. Pro vzdálený přístup je RDP vhodnější než VNC, protože i při menším datovém toku jsou reakce na povely uživatele takřka okamžité.

3.5.3 Škálovatelnost

Vzhledem k zaměření VirtualBox v ohledu škálovatelnosti příliš nevyniká. Jedinou vymožeností, kterou získal, je práce se snapshoty.

3.5.4 Robustnost

VirtualBox se vydal směrem uživatelské přívětivosti a jednoduchosti používání. Při instalaci si do jádra nainstaluje pouze jeden driver. Podstatnou nevýhodou je možnost virtualizovat jen x86 procesory a maximálně 2GB paměti. K virtualboxu je možné přistupovat ze vzdáleného počítače. Do firemního prostředí, kde je potřeba spolehlivost a variabilita, bych virtualbox v žádném případě nedoporučoval. VirtualBox je ideální pro uživatele, kteří potřebují na svém počítači nainstalovat další operační systém s poměrně rychlou odezvou a pokročilejšími grafickými funkcemi.

Kapitola 4

Metodika testování

Na začátku kapitoly jsou rozebrány kategorie, ve kterých budou virtualizační nástroje testovány. Dále je zde popsán program vyvinutý pro testování paralelního běhu virtuálních systémů a pro usnadnění samotného průběhu testování. V poslední části této kapitoly jsou popsány benchmarky vybrané pro testování.

4.1 Co testovat a jak testovat?

Jak se rozhodnout, který virtualizační nástroj je ten nejlepší? Vzhledem k problémům, například malá efektivita a výkon IO operací, či přeplnění TLB tabulky a dalších, rozebraných v kapitole 2.2.3, se při virtualizaci zaměříme právě na ně. Bohužel ne vždy tyto testy budou objektivní vůči reálnému nasazení virtualizačních systémů. Někdy se může stát, že virtualizační nástroj v testech propadne, ale v reálném nasazení bude ostatní nástroje překonávat. Konkrétní popis oblastí testování bude následovat.

4.1.1 CPU

Efektivita zpracování kódu je při virtualizaci jeden z hlavních faktorů, i když ne ve všech případech. Například QEMU, kde je na úkor výkonu možno emulovat skoro jakýkoliv procesor.

Testovat lze pomocí operací, které řádně zatíží procesor, ale ostatní části virtuálního stroje zatíží jen minimálně. Pro tyto účely je dobré využít testy, které jsou volně dostupné na internetu, například *Povray*. Povray sice není program určený přímo pro testování, ale obsahuje i testovací mód.

4.1.2 IO operace a zatížení procesoru při těchto operacích

Dá se říct, že každý virtualizační nástroj má IO operace řešené jiným způsobem, ať už se jedná o KVM či VMware a jiné. Otázka zní, jak měřit efektivitu IO operací. Pro tento účel byl v rámci této práce vyvinut testovací nástroj, který umožňuje měřit zatížení testovaného systému a zároveň zatížení systému, na kterém běží virtualizovaný systém. Testovací nástroj je popsán v následující kapitole 4.2.

Disk Pro běh systému je nutné mít i efektivní přístup k datům. Bohužel při plné virtualizaci je nutné emulovat i hardware nutný pro přístup k disku. Pro srovnání zvolíme virtualizační nástroj KVM, ve kterém není problém přejít mezi režimem plné virtualizace a

paravirtualizace disků. Pro testování použijeme virtualizační nástroj *bonnie++*. Bonnie++ nám umožní otestovat sekvenční i náhodný přístup k disku a současně ukáže zatížení CPU při testovacích operacích.

Network Jestliže chceme využít virtuální stroj jako server např. pro webové služby, bude nás zajímat rychlost a efektivita síťových zařízení. I v tomto případě pro ukázání rozdílu mezi plnou virtualizací a paravirtualizací použijeme virtualizační nástroj KVM. V KVM je změna virtualizace síťových zařízení jen otázkou úpravy startovacího skriptu. Pro testování prostupnosti sítě použijeme testovací nástroj *netperf*.

Grafika Bylo by krásné, kdyby se dala na virtuálních strojích provozovat i 3D grafika. Je to sice možné, ale zatím je to řešeno jen pomocí VGML [4], které umožní podporu openGL ve verzi 1.5. Tato možnost je dostupná jen pro operační systém Linux, FreeBSD, OpenSolaris, protože je nutné využít speciální ovladač v klientském i hostitelském systému. Tyto testy vynecháme, protože problematika je teprve v počátku vývoje.

4.1.3 Systém

Je sice dobré otestovat každou část systému zvlášť, ale není to dostačující řešení. Z tohoto důvodu je nutné přistoupit i ke komplexním testům. Pro tento účel je ideální nástroj *lmbench*, který v sobě obsahuje řadu syntetických testů. Například test *lat_ctx* nástroj pro testování doby nutné k přepnutí kontextu a další.

4.1.4 Komplexní testy

Komplexními testy je možné otestovat systém jako celek, to znamená paměťové operace, diskové operace, síťové operace a výkon či efektivita zpracovávání kódu. Syntetické testy často neukazují výkon systému jako celku, ale jen jeho konkrétních částí. Často se může stát, že systém má po částech velký výkon, ale jakmile se spojí více činností dohromady začne se hroutit. Pro tyto testy byly vybrány testy serveru Apache a MYSQL serveru případně jejich kombinací. Při použití Apache serveru je využit procesor a paměť, disk i síťové rozhraní. MYSQL server zase zajistí zátěž disku, paměti, procesoru. Zatížením obou těchto serverů najednou pomocí benchmarku *ab*, který je standardně obsažen v instalaci apache serveru, bylo dosaženo největšího reálného zatížení. Přes benchmark ab se dotazovalo na php stránku, která obsahovala php kód, pomocí něhož se přistupovalo na MYSQL server. Server MYSQL obsahuje sám o sobě také benchmark, který otestuje všemožné SQL operace, jako je vytváření a mazání tabulek, záznamů atd.

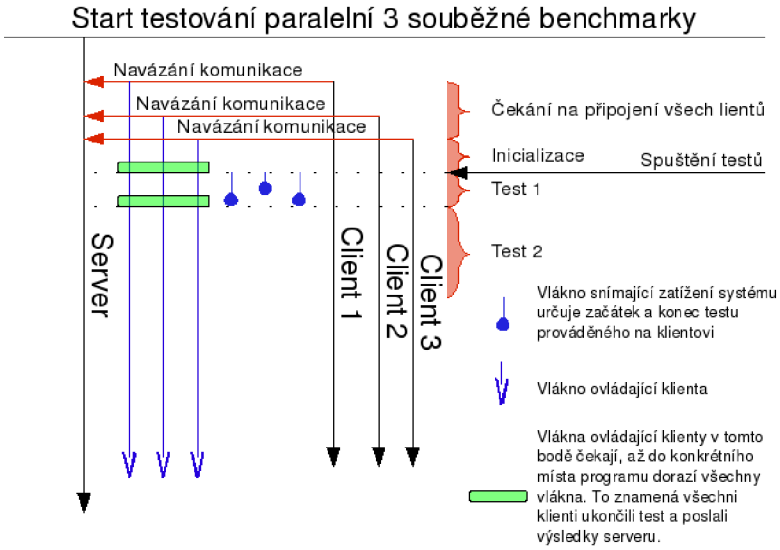
4.1.5 Architektura procesorů

Chod virtualizovaných systémů nezávisí jen na virtualizačním nástroji, ale podstatnou měrou se o efektivitu virtualizace zasluhuje i architektura hardwaru, na kterém se virtualizace provádí. Procesory jako takové je obtížné porovnat, proto je nutné otestovat výkon systému jen relativně. To znamená, zjistit výkon reálného systému, na kterém je virtualizovaný systém spuštěn. Pro zjištění výkonu hostitelského systému byly použity stejné testy, jako pro zjišťování výkonu virtualizovaných systémů.

4.2 Testovací program

Testovací program byl v rámci této práce navržen za účelem testování efektivity virtualizačních nástrojů. Je naprogramován ve skriptovacím jazyce Python. Zdrojové kódy je možné nalézt na přiloženém DVD v adresáři benchmark. Volbou Pythonu je zajištěna přenositelnost a jednoduchost psaní kódu. Aplikace je navržena jako Klient/Server. Komunikace je zajištěna přes TCP/IP spojení. TCP/IP komunikace byla zvolena, protože bylo nutné zajistit přenositelnost i mezi virtualizačními nástroji. Obě části jsou ovládány pomocí příkazové řádky.

Celé testování řídí serverová strana, která obsahuje kontrolu potřebných programů pro testování, zadání testů a řídí pořadí provádění testů. Testy je možné spustit sériově, nebo paralelně. Pro každého klienta se vytvoří samostatné vlákno. Při spuštění testu na klientovi se vytvoří další vlákno, které nezávisle na činnosti serveru monitoruje zatížení počítače, na kterém běží server. Po ukončení testu klient pošle serveru výstup z testu. V tuto chvíli server ukončí monitorovací vlákno a sejme výstup. Pro každého klienta se pro ukládání výsledků vytvoří speciální soubor, jehož jméno je určeno parametrem zadaným při spuštění klientské části programu.



Obrázek 4.1: Ukázka paralelního spuštění benchmarku

Jediná podstatná slabina testovacího progamu je v určení zátěže hostitelského systému, kdy při velice krátkých testech, v určitých případech neodpovídá naměřená hodnota reálnému zatížení CPU.

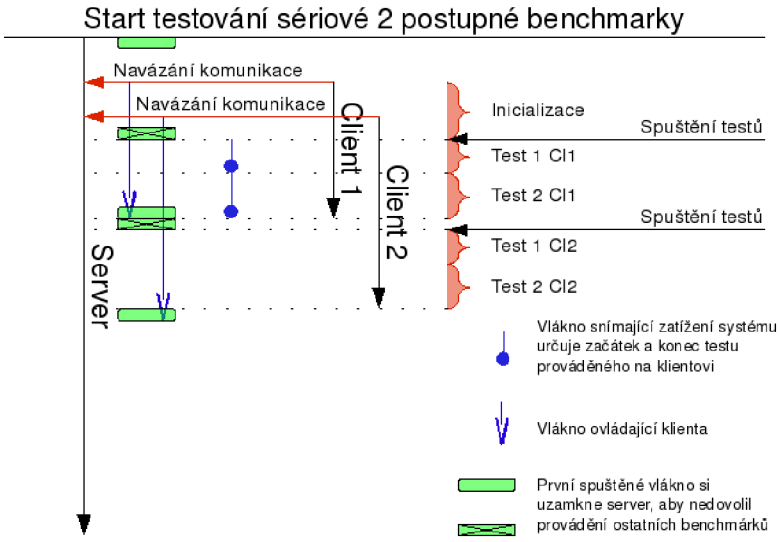
4.2.1 Paralelní běh

Při paralelním spuštění server čeká na připojení zadaného počtu klientů, aby byla zajištěna synchronizace testů. To je ukázáno na obrázku 4.1. V případě, že by některý z testů byl spuštěn na rychlejším stroji, je zajištěna synchronizace začátků testů na všech testovaných strojích. Paralelní provádění testů bylo navrženo pro test zahlcení hostitelského systému velkým počtem virtuálních systémů. Zahlcení hostitelského systému je nutné pro zjištění

efektivity přepínaní kontextů virtuálních systémů. Tento problém je popsán v odstavci správa paměti 2.2.3.

4.2.2 Sériový běh

Pří sériovém spuštění serveru se na server může připojit určené množství klientů a server je postupně odstartuje. Ukázka sériově spuštěného benchmarku je 4.2.



Obrázek 4.2: Ukázka sériového spuštění benchmarku

4.2.3 Výstup programu

Výstup programu je čitelný z obrázku 4.3. Obsahuje tyto části:

- příkaz, který byl poslán na klienta,
- výstup prováděného benchmarku,
- čas a zatížení procesoru na straně klienta,
- informace i zatížení na straně serveru.

```
Comm : bash
/usr/bin/time /usr/lib/lmbench/bin/i686-pc-linux-gnu/lat_ctx -P 2 -s 4 2 4
OK

"size=4k ovr=1.87
2 11.28
4 19.61
6.83user 100.62system 0:54.30elapsed 197%CPU (0avgtext+0avgdata
0maxresident)k 0inputs+0outputs (0major+7043minor)pagefaults 0swaps

****************************************
Prumery:
User:  130
Nice:  0
System:  62
Idle:  629
IOwait:  0
Irq:  0
SoftIrq:  0
User+System:  192
Time:  54.9996659756 s
****************************************
```

Obrázek 4.3: Ukázka výstupu z programu

4.3 Spouštěné benchmarky

V této kapitole bude rozebrán účel benchmarku volaných z testovacího programu popsaného v předchozí kapitole 4.2. Pro testování byly vybrány volně dostupné, či integrované benchmarky, u kterých není problém otestovat téměř jakýkoliv systém.

4.3.1 lmbench

Lmbench je komplexní nástroj uzpůsobený pro testování základních systémových operací a hardwaru. Skládá se ze série micro benchmarků. Je zaměřený na tři základní skupiny: latenci, přenosové pásmo a ostatní. Budou zde probrány benchmarky použité v testech. Další podrobný popis je možné nalézt v manuálových stránkách nebo na internetu [2].

Latence

Virtualizovaný systém má mezi sebou a reálným světem určitou vrstvu softwaru, která zajišťuje virtualizaci. Tato vrstva s největší pravděpodobností způsobí zpoždění. Z tohoto důvodu bude dobré toto zpoždění otestovat. Seznam použitých benchmarků je zde:

- *lat_ctx* je benchmark testující zpoždění při přepínání kontextů. Jako vstupní parametry pro spuštění má velikost a množství procesů, mezi nimiž se má přepínat kontext. Výstup testu je uveden v mikrosekundách.
- *lat_proc* zjistí zpoždění v mikrosekundách při vytváření procedur a procesů pomocí fork a exec.
- *lat_pagefault* je benchmark vracející čas v mikrosekundách, který je třeba pro načtení stránky paměti odložené při swapování na pevný disk.

přenosové pásmo

U velkých datových přenosů, které musí řešit procesor, může dojít k přetížení procesoru, nebo jiné kritické části a způsobit omezení datového toku. U virtualizace toto tvrzení platí dvojnásobně, protože přístup k hardwaru je často ještě odstíněn softwarovou vrstvou.

- *bw_mem* je testovací nástroj na testování rychlosti přístupu k paměti. Výstup je uveden v MB/s.
- *bw_file_rd* je benchmark zjišťující rychlost čtení dat ze souborového systému v MB/s.
- *bw_pipe* je nástroj na zjištění rychlosti přenosu pomocí pipe, jehož výstup je uváděn v MB/s.

Ostatní

Ostatní benchmarky obsahují různé nástroje, například náhradu za program dd (z operačního systému linux) nazvanou *lmdd*.

4.3.2 povray

Povray je program primárně určený pro renderování scén např. pomocí raytracingu a ostatních metod. Ale vzhledem k tomu, že povray využívá procesor na 100% a ostatní zdroje systému téměř nevyužívá, je to ideální nástroj pro testování procesorového výkonu v reálné zátěži. Pro spuštění povray v benchmark režimu stačí přidat jako parametr *–benchmark*.

4.3.3 bonnie++

Bonnie++ je benchmark vyvinutý pro testování rychlosti a efektivnosti diskových operací. Zahrnuje v sobě sekvenční a náhodné testy pro čtení a zápis na disk. Tyto testy provádí po znacích, po blocích a různými přepisy. Dále otestuje rychlost vytváření objektu ve filesystému.

4.3.4 apache

Apache byl zvolen, protože je hojně používaný webový server a obsahuje integrovanou testovací utilitu. Testovací utilita se jmenuje *ab*. Utilita pracuje na principu kladení dotazů na požadovaný server. Apache je velice komplexní server, proto i touto jednoduchou utilitou jde testovat velké množství typů zátěže webového serveru. Při samotném testování je pak apache serverem možné zatížit síťové, diskové a paměťové rozhraní, v neposední řadě i procesor.

4.3.5 MYSQL

MYSQL obsahuje utilitu na testování zátěže SQL serveru. Tato utilita se nazývá *sql-bench*. Nejedná se ani tak o utilitu, ale spíše o soubor benchmarků zaměřených na konkrétní záležitosti. Vzhledem k tomu, že netestujeme MYSQL server jako takový, nýbrž ho používáme jen jako test na výkon celého systému, pro spuštění použijeme obecně nakonfigurovaný benchmark *run-all-tests*.

4.3.6 apache + MYSQL

Spojením obou těchto serverů dostaneme test, který simuluje takřka reálné zatížení celého systému. Testování probíhá opět pomocí utility *ab*, kterou se dotazujeme apache serveru na php stránku, jejíž obsahem je dotaz na MYSQL databázi. V tomto případě už jde o komplexní test, který by mohl ukázat, jak na tom jsou virtualizační nástroje v reálném prostředí.

Kapitola 5

Testy virtualizačních nástrojů

V této kapitole budou uvedeny výsledky testů popsané v předešlé kapitole 4, provedené na virtualizačních nástrojích popsaných v kapitole 3, jako je KVM verze 65, VirtualBox 1.56, VMware Server 1.0.5, Xen 3.1.0. Výsledky testů jsou uvedeny vždy od nejlepšího na levé straně až po nejhorší na straně pravé.

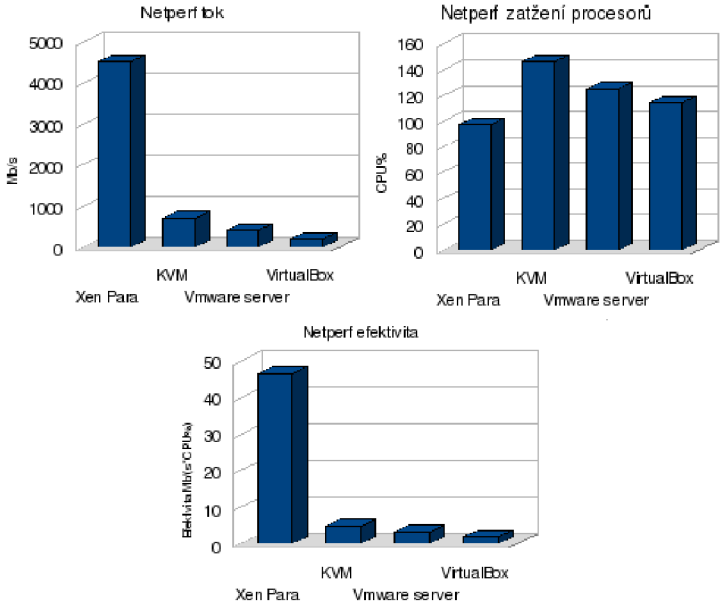
5.1 Porovnání výkonnosti a efektivity virtualizačních nástrojů

V této části bude porovnán výkon výše popsaných virtualizačních nástrojů. Bude zde srovnání výkonnosti v síťových aplikacích. Dále zde bude porovnán výkon operací souborového systému, konkrétně sekvenční i náhodné čtení a zápis. Virtualizační nástroje zde budou srovnány i podle výkonnosti celého systému. Pro tuto část testu budou použity testy prováděné na procesoru Intel Xeon 2x1,6GHz. Pro testování byl použit Linux, distribuce Fedora8, do kterého byl nainstalován kernel 2.6.25. Kernel 2.6.25 má plně implementovaný nový plánovač CFQ (Completely Fair Queuing) a integrovanou podporu virtio popsané v kapitole 3.2.1. Pro virtualizační nástroj Xen bylo nutné při paravirtualizaci použít Kernel 2.6.21. Výsledky zatížení procesoru mohou přesáhnout 100%, to je způsobeno použitím dvou jádrového procesoru a programu vyhodnocujícího zatížení procesoru, uváděného jako součet zatížení všech procesorů. I když tento způsob vyjádření může ze začátku působit zmatečně, je z něj více patrné, kolik procesorů se daného testu účastnilo.

5.1.1 Síťové operace

V těchto testech je testován přístup směrem od klientských virtuálních strojů do hostitelského systému. Tento postup byl zvolen, protože je měření nezávislé na cizím provozu na síti a ukazuje maximální výkon síťového rozhraní. Pro kontrolu, zda hostitelský systém neomezuje síťový přenos, byl vždy ověřen i samotný hostitelský systém. V tomto případě byl na hostitelském systému naměřen tok 8195 Mb/s při zatížení procesoru 102%. Z toho vyplývá, že prováděné testy ani v nejmenším nedosáhnou maximálního síťového toku hostitele. V grafech je zobrazena i efektivita. Efektivita je vyjádřena jako poměr datového toku a zatížení procesoru.

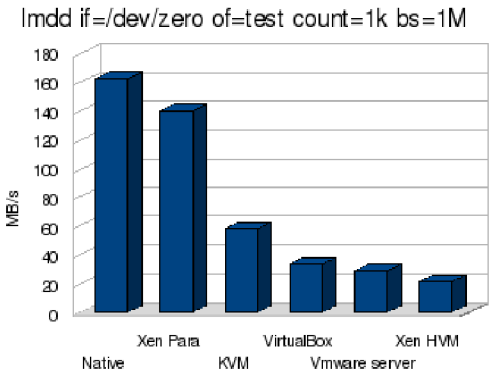
Testy zobrazené na obrázku 5.1 ukazují převahu virtualizačního nástroje Xen v režimu paravirtualizace, kdy všechny ostatní nástroje daleko předčil, ještě ke všemu s menším zatížením procesoru. Na druhém místě skončilo KVM, které pro síťový provoz používalo paravirtualizované drivery. Ostatní virtuální stroje měly nainstalované drivery z přiložených virtuálních CD.



Obrázek 5.1: Porovnání výkonnosti síťových operací

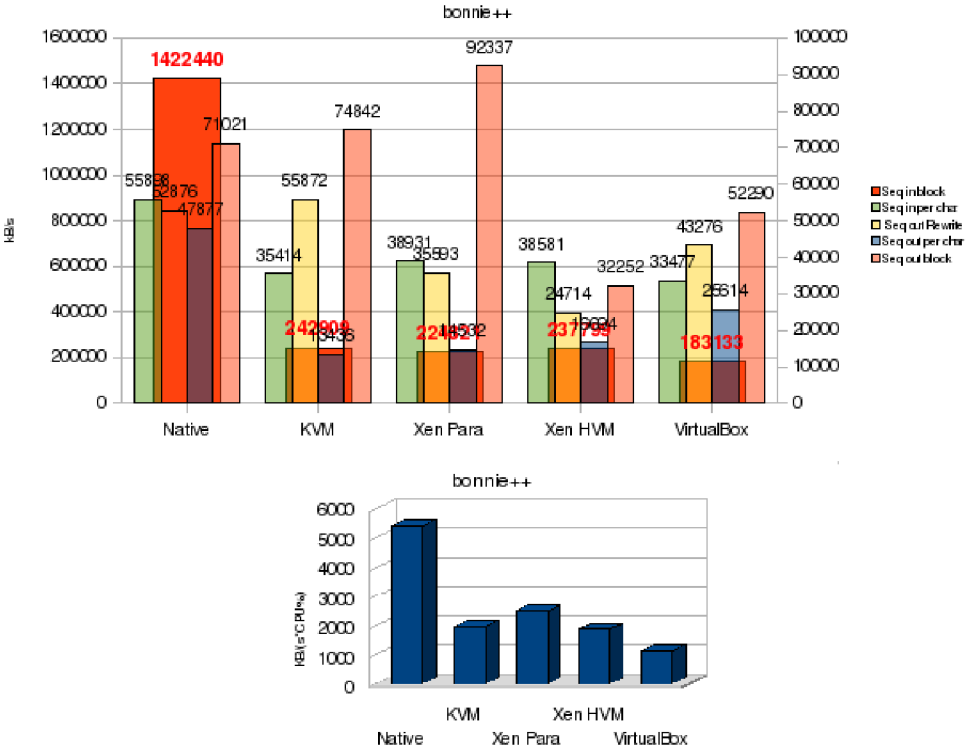
5.1.2 Diskové operace

Do této sekce se dají zařadit výsledky benchmarku bonnie++ i microbenchmarku lmdd obsažený v balíku benchmarků lmbench. V podání lmdd jde jen o test zápisu na disk. Zápis je prováděn po blocích o velikosti 1MB. Výsledky testu jsou uvedeny v grafu 5.2.



Obrázek 5.2: lmdd porovnání výkonu blokového zápisu na disk

Benchmark lmdd by jako benchmark diskového provozu nestačil, proto bylo nutné použít i benchmark bonnie++. Z výsledků bonnie++, které jsou uvedeny v grafu 5.3, je sice možné vyčíst výkonnost, ale s určením efektivity může být problém. Proto bylo nutné zavést další veličinu vyjádřenou poměrem průměrné rychlosti a zatížením procesoru.



Obrázek 5.3: Výsledky testu bonnie++

U diskových operací bylo pro změnu první KVM. KVM vyhrálo hlavně díky sekvenčním operacím, ať už se jedná o čtení, zápis, nebo přepis. Jediná výrazná slabost KVM je jeho pomalost v zápisu po znacích. V případě, že by se do hodnocení počítala i efektivita, zvítězil by opět Xen, protože při o trochu menším výkonu diskových operací byl efektivnější. Na posledním místě se umístil VMware server, jehož efektivita je vůči KVM jen čtvrtinová.

5.1.3 Systémové microbenchmarky lmbench

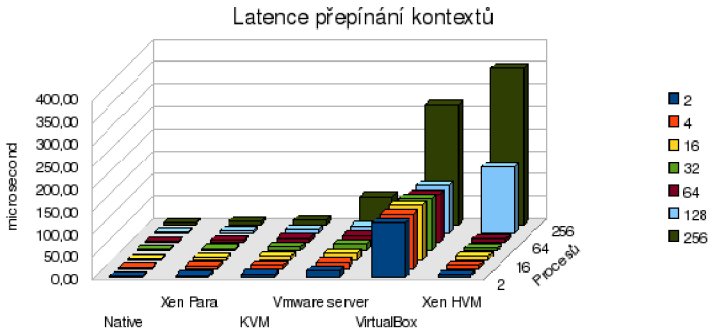
Po otestování IO operací zbývá otestovat systémové operace. Lmbench je pro to ideální nástroj, jak již bylo napsáno výše v kapitole 4.3.1. Nejprve bylo provedeno testování zpoždění, následně pak testy datového toku.

Zpoždění

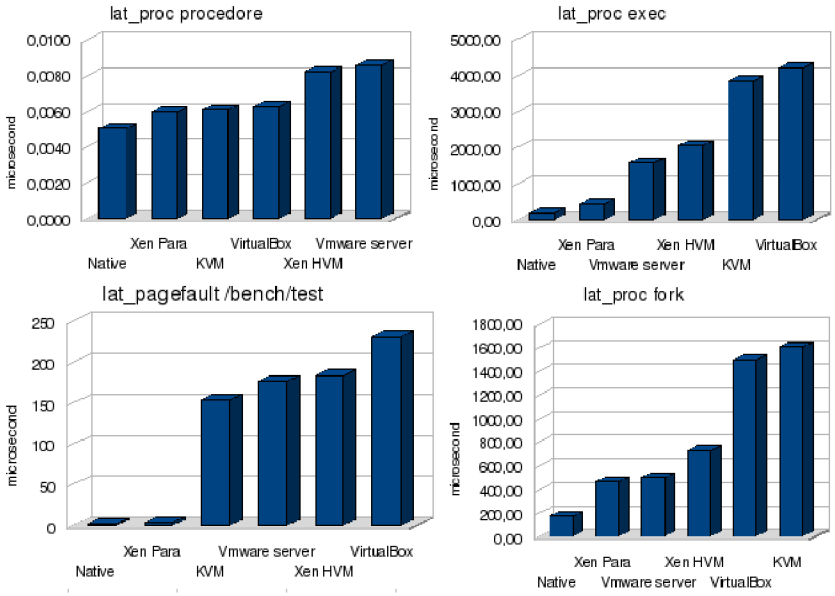
První test je přepínání kontextů pomocí lat_ctx. V grafu 5.4 je z testu patrné, že opět zvítězil Xen, zatímco nejhůře, podle průměru naměřených hodnot, dopadl VirtualBox. Přesto při velkém množství přepínaných kontextů je na posledním místě Xen HVM.

Další nepříjemnou záležitostí je pagefault obzvláště u virtuálních systémů, kdy se odložená stránka musí načítat z virtuálního disku, což je ještě pomalejší, než u normálního systému. Graf 5.5 ukazuje jak si virtualizační nástroje vedou.

Dále byla testována rychlost vytváření procedur a procesů. Z grafu 5.5 je patrné, že překvapivě jeden z nejhorších výsledků měl KVM společně s VirtualBoxem.



Obrázek 5.4: Latence přepínání kontextů

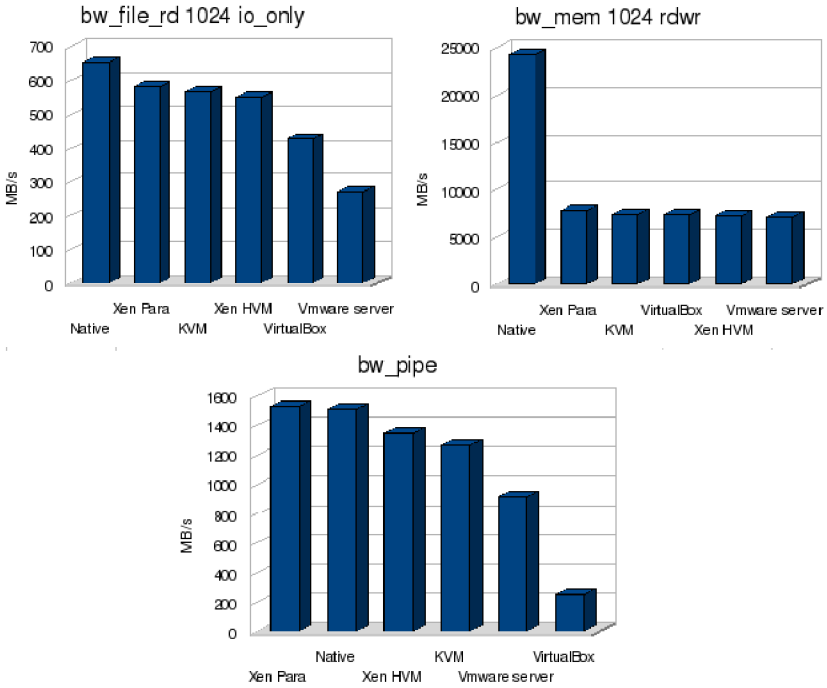


Obrázek 5.5: Zpoždění při pagefault, vytváření procesu atd..

Přenosové pásmo

V této části budou virtualizační nástroje srovnány z pohledu jejich datové prostupnosti vyjímaje diskových a síťových operací, jež byly otestovány výše 5.1.1. Půjde hlavně o prostupnost paměti, prostupnost rozhraní pro čtení z disku, ale ne přímo o čtení z disku. Dále bude otestována propustnost oblíbeného linuxového nástroje pipe.

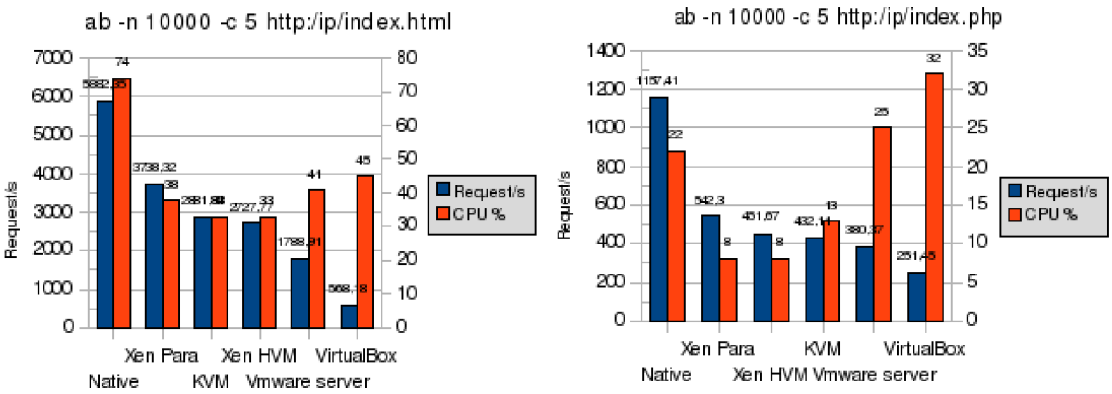
Pokud se jedná o datová rozhraní většinou paravirtualizovaný Xen nemá takřka konkurenci. To je patrné z obrázku 5.6. Zajímavý úkaz vznikl na grafu bw_pipe, kde díky své rychlosti a menší chybě měření předběhl nativní systém.



Obrázek 5.6: přenosové pásmo systémových rozhraní

5.1.4 Komplexní testy

V této části bude testován systém jako celek. Pro tento test byl vybrán program ab, který je instalován jako součást apache serveru.



Obrázek 5.7: Počet vyřízených dotazů a zatížení procesoru

Jak je viditelné z grafů 5.7, na prvním místě se umístil paravirtualizovaný Xen, což se dalo předpokládat z předchozích testů. Na dalších dvou místech se umístili KVM a Xen HVM. Nejhůře se umístil VirtualBox.

5.1.5 Shrnutí porovnání virtualizačních nástrojů

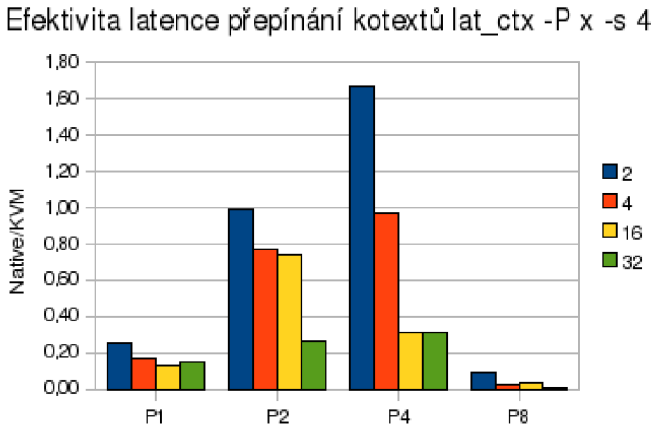
Co se výkonnosti týče, na prvním místě se umístil Xen, protože měl nejlepší výsledek v 14 z 16 testů. Na druhém místě se umístil KVM s paravirtualizovaným ovladačem disku a síťové karty a na třetím Xen HVM. Jak je patrné z testů, v dnešní době je paravirtualizace nejvýkonnějším řešením, ale jen do doby, kdy přijde Hybridní virtualizace, která by měla paravirtualizaci překonat.

5.2 Porovnání efektivity využití výce procesorů v smp

Porovnání efektivity procesorů bylo testováno na nástroji KVM. KVM bylo vybráno, protože dopadl nejlépe z plně virtualizovaných nástrojů, a také kvůli jeho jednoduché konfiguraci a správě virtuálních systémů. Testy byly prováděné na virtualizovaném systému, který obsahoval paravirtualizované disky a síťová rozhraní. Virtuální systémy byly testovány stejným způsobem jako v minulé části. Aby se otestoval symetrický multiprocesing, byly testy v paralelním režimu. Počet paralelních běhů odpovídal počtu přidělených fyzických procesorů. Některé testy neměly žádnou vypovídací hodnotu (z výsledků nebylo nic poznat), proto byly vypuštěny. Další testy byly vypuštěny z časových důvodů. Příkladem za všechny je netperf, jehož výkon změna počtu procesorů vůbec neovlivnila. Tyto testy byly měřeny na počítači, který obsahoval dva 4 jádrové procesory Intel Xeon na frekvenci 2.6GHz.

5.2.1 Systémové microbenchmarky lmbench

Na testech latence z grafu 5.8 není nic zvláštního, jediné co stojí za zmínku je nedokonalost nástroje KVM s osmi přidělenými procesory. V grafech je vidět obrovský nárůst zpoždění, který neodpovídá zdvojnásobení zátěže procesoru. [1]

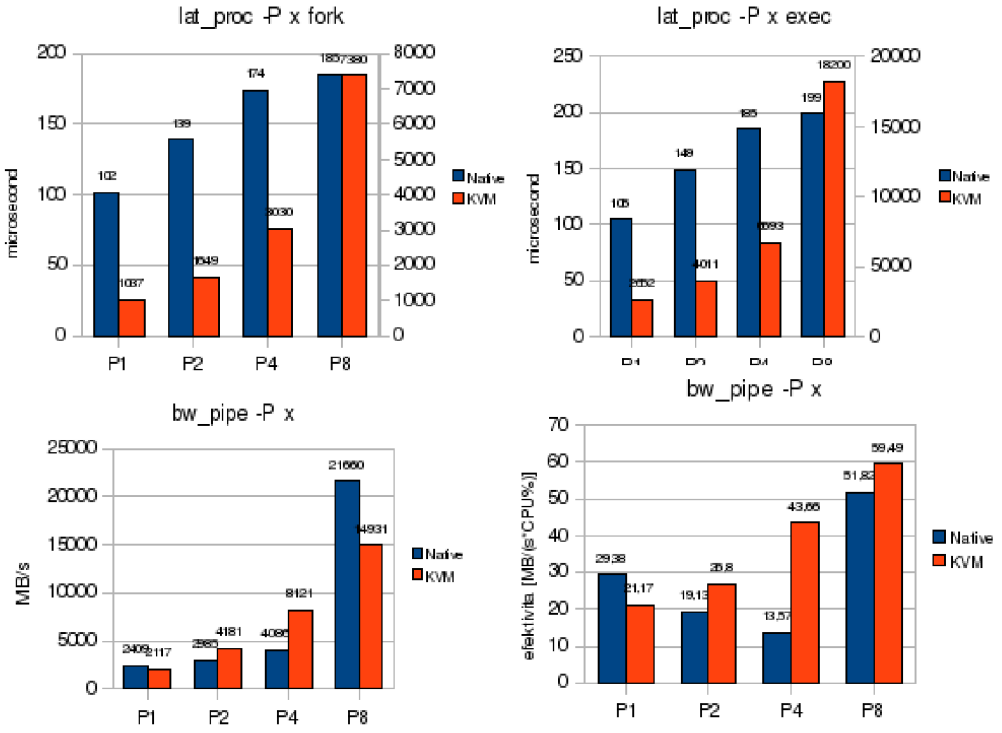


Obrázek 5.8: Efektivita přepínání kontextů

Velkou latenci je možné vysvětlit i přeplněním tlb tabulky v procesoru, která musela

[1] Zajimavé je sledovat výsledky testů spuštěných na nativních procesorech. Jestliže není zatíženo všech 8 procesorů je nárůst výkonu odpovídající předpokladům. V případě zátěže všech 8 procesorů se dá vypozorovat až několikanásobný nárůst výkonu. Jediné vysvětlení, které mě napadá je buď nějaká optimalizační technika v procesorech intel, nebo všechny procesory zpracovávají jediný program. V případě číslo dvě procesor není obtěžován jinými procesy a proto nemusí čekat na sběrnici a jiné části počítače.

udržovat velké množství informací o paměti mapované virtuálními systémy. Tento problém by mohla vyřešit další generace virtualizace, která je obsažená v procesorech od AMD s kódovým označením jádra barcelona. Barcelona, jak už bylo zmíněno v předchozích kapitolách 2.2.3, podporuje tzv. nested paging. U testů datového toku, jak je vidět v grafu 5.9, si KVM vede o poznání lépe a efektivita přenosu pomocí pipe je dokonce lepší jak u nativního systému.



Obrázek 5.9: Efektivita využití více procesorů

Efektivita je opět vyjádřena jako poměr datového toku a zátěže procesoru v %. Překvapivě nejlepší efektivitu přepínání kontextů vůči nativnímu systému má virtualizovaný systém, kterému byly přiděleny 4 procesory. Je to způsobeno poklesem výkonu v nativním systému a překvapivým udržením výkonnosti ve virtuálním stroji.

5.2.2 Shrnutí testů smp ve virtuálních systémech

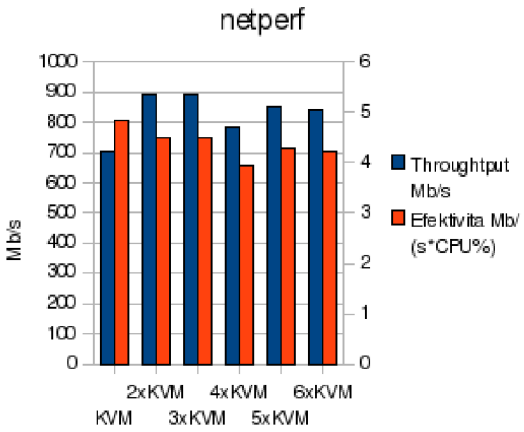
V případě, že je potřeba, aby virtuální systém umožňoval zpracovávat velké datové toky, je dobré virtuálnímu systému přidělit větší množství procesorů. Naproti tomu v případě, kdy nám jde o zpoždění systému, nemá přidělování velkého množství procesorů význam. Nevýhodou virtuálních systémů je jejich náročnost na TLB. Každý virtuální systém má své mapování do fyzické paměti a tím přetíží TLB. Vyřešit tento problém je možné pomocí nových generací virtualizací, které vnáší do procesorů další prvek virtualizace, a to lepší podporu virtualizace paměti, případně použitím hugepage. Hugepage hostitelskému systému umožní vytvořit větší paměťové segmenty, čímž se zmenší jejich počet a uvolní se tolik potřebný prostor v TLB.

5.3 Výkon a efektivita provozu více virtuálních systémů najednou

V této části bude porovnána výkonnost a efektivita spuštění jednoho až 6 virtuálních systémů na jednom fyzickém. Testy byly prováděny na sestavě obsahující procesor Intel Xeon s dvěma jádry o frekvenci 1.6GHz a 8GB RAM. Byla vybrána sestava se dvěma procesory i přes to, že byla k dispozici sestava s osmi jádry, protože na sestavě s méně procesory je lépe patrné přetížení. Testy byly prováděné na virtualizačním nástroji KVM, které obsahovalo paravirtualizované disky a síťová rozhraní.

5.3.1 Síťové operace

Výkon síťových operací byl testován vůči hostitelskému systému, aby se zamezilo vlivu okolí a bylo možné dosáhnout maximálního výkonu.



Obrázek 5.10: Efektivita a propustnost síťového rozhraní

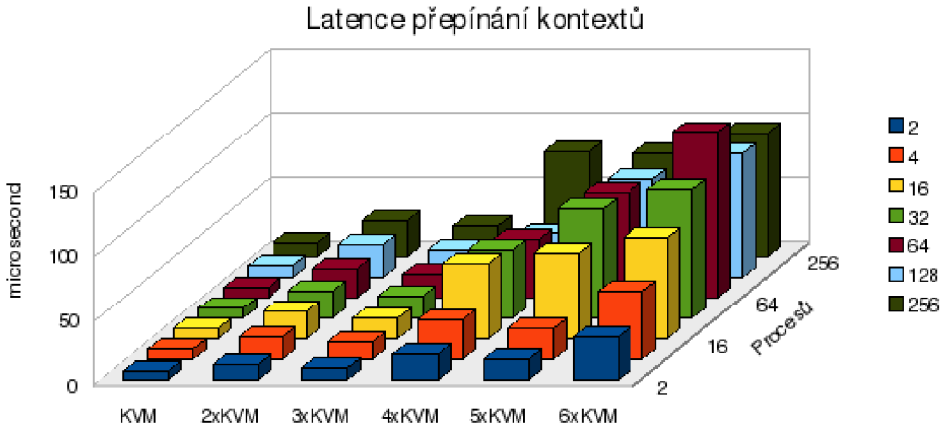
Z testů uvedeného v grafu 5.10 je zřejmá stabilita výkonu síťových operací. Sice na úkor efektivity, ale to jen minimálně. Při spuštění více virtuálních systémů je možné dosáhnout dokonce většího datového toku.

5.3.2 Systémové microbenchmarky lmbench

První z kategorie testů zabývající se zpožděním je opět lat_ctx, jehož výsledky jsou uvedeny v grafu 5.11. Při provozování jedné až tří virtuálních systémů současně není s přepínáním kontextů problém, problém opět nastává při alokaci velkého množství paměti, kde už se TLB tabulka umístěná v procesoru dostává do problémů. Procesor je nucen tabulky mapování z virtuální paměti na fyzickou a z fyzické na strojovou paměť načítat přímo ze strojové operační paměti.

Zpoždění

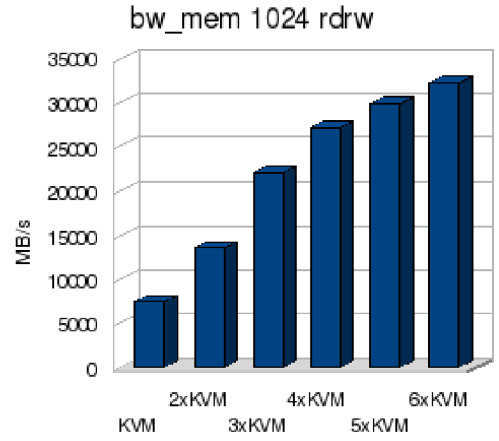
Obecně u testů týkajících se latence není problém provozovat 3-4 plně vytížené virtuální systémy na jednom dvou jádrovém procesoru.



Obrázek 5.11: Latence přepínání kontextu

Přenosové pásmo

I z testů propustnosti paměti, uvedených v grafu 5.12 není s testovanou sestavou problém provozovat 3-4 plně vytížené virtuální systémy. Při zvětšování počtu virtuálních systémů se musí dělit o paměťovou prostupnost a klesá efektivita.

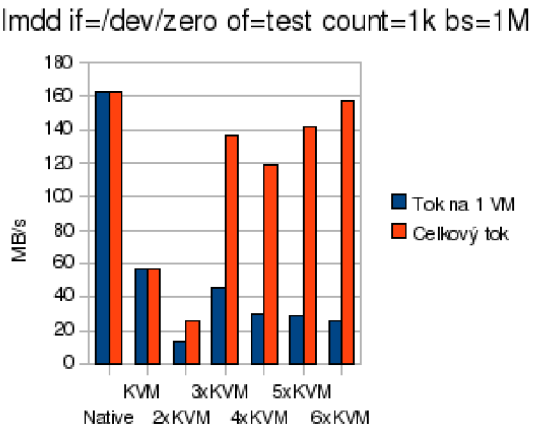


Obrázek 5.12: Propustnost pamětí

Diskové operace

Rychlost diskových operací byla měřena jen zběžně. Proto byla využita utilita lmdd, která je schopna otestovat zápis na disk. Měření ale poskytlo zajímavé výsledky, jak je uvedeno v grafu 5.13.

Při zápisu jednoho virtuálního systému na disk dosahuje svých standardních 57MB/s z možných 160MB/s. Po spuštění druhého virtuálního stroje bylo provedeno buď chybné měření nebo došlo k nějaké výjimečné situaci v systému, proto tento test nebudeme brát v úvahu. Zajímavé to začíná být při spuštění třetího virtuálního systému. Celková propustnost

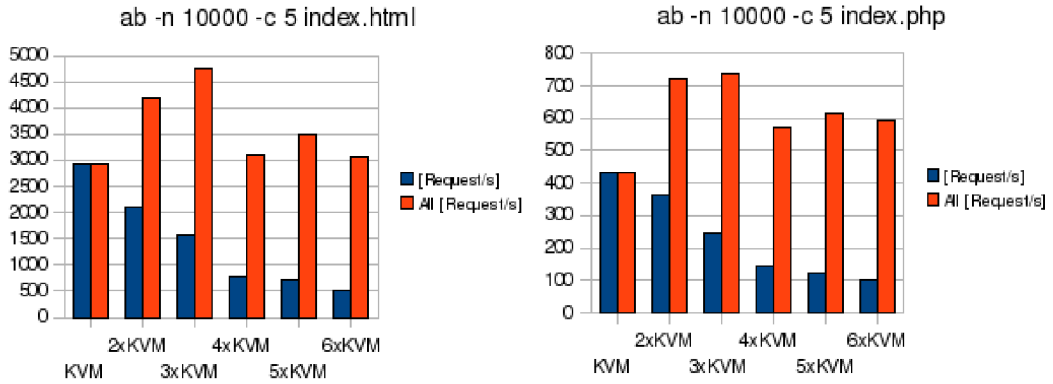


Obrázek 5.13: Prostupnost blokového čtení z disku

zápisu na disk je rázem 135MB/ s, což už je skoro srovnatelné s nativním systémem. Při zvyšování počtu virtuálních systémů propustnost s menším kolísáním nadále roste. Skoro to vypadá, jako by mělo KVM chybný virtuální ovladač, nebo bylo uměle limitováno.

5.3.3 Komplexní testy

Komplexní testy systému jen potvrdily to co napovídaly předešlé syntetické testy. Speciálně u apache serveru je výhodnější na jednom fyzickém systému provozovat více virtuálních systémů.



Obrázek 5.14: Apache server benchmark ab

Optimální využití serveru s dvoujádrovým procesorem je 2-3 virtuální systémy s webovým serverem, což je zřejmé z grafů 5.14.

5.3.4 Shrnutí výsledků testů virtualizace více virtuálních systémů na jednom fyzickém stroji

Jestliže chceme efektivně využít server pomocí virtualizace, je téměř vždy výhodnější spustit více virtuálních systémů na jednom fyzickém, a to i za cenu toho, že jednotlivé virtuální

stoje budou pomalejší. V případě, že bychom chtěli provozovat jen jeden virtuální systém, často ztratíme podstatnou část výkonu.

5.4 Porovnání plné virtualizace a paravirtualizace u KVM

V této části bude porovnán výkon KVM s plně emulovaným hardwarem a KVM pro jehož diskové a síťové operace je použit paravirtualizovaný hardware. V tomto případě taktéž na virtualizačním nástroji KVM. KVM bylo testováno na AMD AthlonX2 jádra o frekvenci 2.5GHz a operační paměti o velikosti 8GB. Při tomto porovnání bude postupováno trochu jiným způsobem, protože jsou porovnávány jen tři výsledky. Nebudou zde uvedeny grafy, ale jen souhrnné výsledky testů uvedené v tabulce 5.15, které by měly být jen prostým pohledem rozlišitelné.

5.4.1 Výsledky provedených testů

Po prohlédnutí testů uvedených v tabulkách v obrázku 5.15 je zřejmé, že paravirtualizovaný síťový ovladač funguje naprosto bez problému, dosahuje dvakrát většího výkonu a přitom využívá procesor stejně jak plně virtualizovaný síťový ovladač. V diskových operacích testovaných pomocí bonnie++ dosahuje paravirtualizovaný přístup takřka stejných výsledků jako plně virtualizovaný systém. Jediný podstatný rozdíl je v sekvenčním čtení, kde paravirtualizovaný přístup na disk dosahuje o 1/3 větší výkon. Podstatná výhoda je v efektivitě IO operací, kde jsou paravirtualizovaná zařízení 2x efektivnější ať už jde o rychlost nebo spotřebu procesorového času. V sekci micro benchmarku z lmbench, systém s paravirtualizovaným zařízením dopadl téměř srovnatelně s plně virtualizovaným systémem, což se dalo očekávat, protože jádro virtuálního systému běží v režimu plné virtualizace. Propad výkonu paravirtualizovaného KVM v syntetických benchmarcích si nedovedu rozumě vysvětlit.

5.5 Nástroje potřebné pro provedení vlastních testů

Pro vlastní testování je na DVD v adresáři image přibalen i obraz disku virtualizovaného operačního systému Fedora8. Obrazy diku jsou uloženy ve formátu raw pro zajištění jednoduché přenositelnosti mezi virtualizačními nástroji. Soubor fedora.img nese systém s plně virtualizovaným hardwarem, který je vhodný pro převod do jiného virtualizačního nástroje a obraz fedoraio.raw obsahuje systém, do kterého byly přidány virtio moduly, které zajišťují paravirtualizovaný chod vybraného hardwaru pod virtualizačním nástojem KVM. Na DVD jsou ještě v adresáři tools přibalené virtualizační nástroje KVM, VMware Server, VirtualBox. Nástroj Xen na DVD není, protože jeho instalace příliš zasahuje do systému a je lepší ho nainstalovat s balíčku distribuce. Výsledky testů jsou uvedeny v souboru s názvem test.xls

5.6 Ostatní testy uvedené na internetu

V této části budou shrnuty další testy uvedené na internetu. První z uvedených, Linux KVM Virtualization Performance, je možné nalézt na stránkách serveru Phoronix, který se z velké části zabývá problematikou Linuxu. V jednom ve svém článku na stránce http://www.phoronix.com/scan.php?page=article&item=623&num=1 porovnává QEMU + kqemu, Xen 3.0.3 a KVM Linux 2.6.20-rc3. Výsledky testů uvedené v článku odpovídají výsledkům

uvedeným v této práci. Za zmínku stojí snad jen zlepšení výkonu KVM při paměťových operacích, oproti verzi KVM uvedené na stránkách. Další test je uveden v článku na stránce http://www.linuxinsight.com/finally-user-friendly-virtualization-for-linux.html. V tomto článku byly srovnány virtualizační nástroje založené na QEMU, jako je KVM, samotné QEMU a QEMU + kqemu. Testy vyšly taktéž podle předpokladů, kde KVM skončilo na prvním místě, na druhém QEMU + kqemu a na posledním samotné QEMU. Další stránka, která je k dosažení na odkazu http://www.cmg.org/measureit/issues/mit41/m_41_1.html, neukazuje ani tak výsledky testů, ale ukazuje, jak je boj o první příčky virtualizačních nástrojů silný. V článku jsou srovnány Xen a VMware ESX Server 2x v jednom testu, každý od jiné firmy a pokaždé s jiným výsledkem. Firmám asi stojí za to, zfalšovat výsledky testů.

Netperf

	virt CPU %	rel CPU %	čas s	tok Mb/s	Efektivita Mb/(s*CPU%)
Native	97	172	10	6695	38.92
KVMfull	97	121	10	361	2.98
KVMpara	96	130	10	805	6.19

lat_ctx -s 4

	2	4	16	32	64	128	256	virt CPU %	rel CPU %
Native	0.71	1.30	2.02	2.39	3.57	4.32	4.47	98	98
KVMfull	12.22	22.23	25.12	26.79	29.62	30.53	30.01	97	127
KVMpara	12.59	18.18	22.87	24.55	27.08	27.67	28.13	95	96

lat_proc procedure

	u sec
Native	0.0032
KVMpara	0.0038
KVMfull	0.0041

lat_proc exec

	u sec
Native	357.00
KVMfull	1638.00
KVMpara	3125.00

bw_file_rd 1024 io_only

	MB/s
Native	654
KVMfull	627
KVMpara	567

lat_proc fork

	u sec
Native	141.00
KVMfull	1616.00
KVMpara	1761.00

bw_mem 1024 rdwr

	MB/s
Native	21069
KVMfull	13061
KVMpara	12148

bw_pipe

	MB/s
Native	1458
KVMpara	842
KVMfull	817

lmdd if=/dev/zero of=test count=1k bs=1 M

	MB/s
Native	242
KVMfull	65
KVMpara	83

lat_pagefault /bench/test

	u sec
Native	1.38
KVMfull	10
KVMpara	10

bonnie++ -u test:test

	Seqencni vystup						Seqencni vztup			
	Seq out per cha	CPU	Seq out block	CPU	Seq out Rewrite	CPU	Seq in per chai	CPU	Seq in block	CPU
Native	62766	98	131223	45	122499	23	76224	99	2587878	99
KVMfull	46221	95	147074	35	117451	42	52246	98	346791	89
KVMpara	43154	83	61034	10	147188	59	46100	94	426297	94

HOST		Guest		
CPU	time s	CPU	průměr	efektivita [kB/(s*CPU%)]
73	104	73	596118	8166
102	133	78	141957	1391.73
90	82	64	144755	1608.38

/usr/bin/time ab -n 10000 -c 5 http://127.0.0.1/index.html

	time	Request/s	Time/request ms	Transfer rate KB/s	CPU %	HOST CPU %
Native	1.8	5555.56	0.18	2510	79	183
KVMpara	5.5	1818.18	0.55	772	35	97
KVMfull	6.3	1587.3	0.63	672	32	99

/usr/bin/time ab -n 10000 -c 5 http://127.0.0.1/index.php

	time	Request/s	Time/request ms	Transfer rate KB/s	CPU %	HOST CPU %
Native	9.23	1083.42	0.92	58773	18	156
KVMpara	26.5	377.36	2.65	18948	16	98
KVMfull	27.6	362.32	2.76	18253	16	99

Obrázek 5.15: Výsledky testů KVM s emulovaným a KVM s paravirtualizovaným hardwarem

Kapitola 6

Závěr

V této práci jsem se snažil o shrnutí důležitých pojmů týkající se virtualizace na platformě x86 a metodách virtualizace vybraných virtualizačních nástrojů. Vybrané nástroje byly otestovány, a pokud to bylo možné, navzájem porovnány. Testy byly prováděny ve spolupráci s firmou Red Hat, která zajistila potřebné hardwarové vybavení nutné pro testování.

Virtualizační nástroje byly porovnávány v následující oblastech: výpočetní výkon a efektivita provádění základních i komplexních systémových operací, možnosti škálovatelnosti a robustnosti, nebylo opomenuto ani na srovnání možností administrace a přívětivost uživatelského rozhraní. Výsledky testů jsou vždy uvedeny přímo v místě, kde je daná problematika testována, aby bylo zřejmé jakým způsobem, a proč se k daným výsledkům došlo. Testování bylo rozšířeno o porovnání výkonnosti paravirtualizace a plné virtualizace u virtualizačního nástroje KVM.

Dále je možné práci rozšířit o další virtualizační nástroje, popřípadě další architektury počítačů umožňujících virtualizaci. V budoucnu by bylo možné v práci pokračovat, otestovat a srovnat nové virtualizační nástroje pomocí dalších, lépe cílených testovacích nástrojů.

Literatura

[1] WWW stránky. Kvm homepage. http://kvm.qumranet.com/kvmwiki.

[2] WWW stránky. manualove stranky lmbench. http://www.bitmover.com/lmbench/lmbench.8.html.

[3] WWW stránky. Qemu documentation. http://fabrice.bellard.free.fr/qemu/qemu-doc.html#SEC1.

[4] WWW stránky. vmgl homepage. http://www.cs.toronto.edu/ andreslc/xen-gl/.

[5] WWW stránky. Vmware homepage. http://www.vmware.com.

[6] WWW stránky. Wikipedie internetová encyklopedie. http://en.wikipedia.org/wiki/Virtualization.