VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NASAZENÍ VIRTUÁLNÍCH SERVERŮ PRO ISP

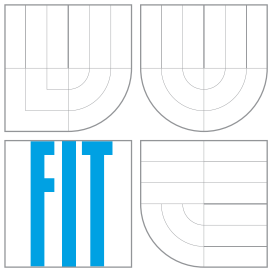
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN ZELENÝ

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NASAZENÍ VIRTUÁLNÍCH SERVERŮ PRO ISP

USE OF VIRTUAL SERVERS BY ISP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN ZELENÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. TOMÁŠ KAŠPÁREK
SUPERVISOR

BRNO 2008

Abstrakt

Práce popisuje principy fungování virtualizace založené na projektu Xen. Text je zaměřen hlavně na zprovoznění systému za účelem jeho pozdějšího využití v prostředí ISP. Cílem této práce je také prozkoumat možnosti právě tohoto pozdějšího nasazení v prostředí ISP a vybrané řešení dále rozpracovat, aby bylo možné jej buď nasadit, nebo na vývoji pokračovat nezávisle na autoru této práce. Práce navrhuje a z větší části také implementuje systém, na kterém lze založit provoz širokého spektra služeb, kdy pro provoz těchto služeb není třeba zasahovat do již naprogramovaného systému.

Klíčová slova

virtualizace, paravirtualizace, Xen, hypervisor, Xen hypervisor, XenLinux, virtuální systém, ISP, PHP, MySQL, modulární systém, balíčkovací systém, webová administrace, Linux, Apache, webhosting, Ubuntu, privilegovaný systém, neprivilegovaný systém, dom0, dom-0, dom-U, domU, kernel

Abstract

This thesis describes principals of virtualization based on the Xen project. Text focuses mainly on putting system into service for purpose of it's subsequent use by ISPs. The goal of this work is also to explore the possible usages of virtualized systems based on the Xen by ISPs. One of these possibilities will be specified to the point when it will by usable or it will be able to be developed independently of author of this work. This work also designs and mostly implements system, which can be used as a base for wide spectrum of of services, when installing these services doesn't effect the rest of the system.

Keywords

virtualization, paravirtualization, Xen, hypervisor, Xen hypervisor, XenLinux, virtual system, ISP, PHP, MySQL, modular system, packaging system, web based administration, Linux, Apache, webhosting, Ubuntu, privileged guest, unprivileged guest, dom0, dom-0, dom-U, domU, kernel

Citace

Jan Zelený: Nasazení virtuálních serverů pro ISP, bakalářská práce, Brno, FIT VUT v Brně, 2008

Nasazení virtuálních serverů pro ISP

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Kašpárka

.....
Jan Zelený
12. května 2008

Poděkování

Tímto bych rád poděkoval p. Stanislavu Slepičkovi, díky kterému jsem získal odborný náhled do praktického využití některých technologií uvedených v této práci.

Obsah

Kapitola 1

Úvod

Virtualizace se v posledních letech stává stále více využívanou metodou jak provozovat více operačních systémů na jednom počítači. Nachází velmi široké uplatnění od desktopových až po serverové aplikace. S přibývajícím výpočetním výkonem přibývá i nevyužitý výkon. Čím dál víc se tak ztrácí režijní výkon spojený s přepínáním kontextu procesoru. Navíc při současném trendu úspory energie je o důvod víc, proč virtualizaci využít – lépe se využijí prostředky nabízené počítačem.

Systém Xen je jedním z nejmladších systémů působících na tomto poli. O to je však jeho návrh kvalitnější. Využívá totiž o něco modernější principy funkčnosti a podporuje tak širší škálu možností. Jeho otevřenost z něj navíc dělá projekt, který se může velmi rychle rozvíjet. Má tak šanci dohnat i konkurenty, kteří těží z dlouhodobě pevné pozice na trhu.

Cílem této práce je prozkoumat možnosti nasazení systémů provozovaných virtualizovaně právě pomocí Xenu v prostředí poskytovatelů připojení k internetu a vybrané řešení dále rozpracovat, aby bylo možné jej buď nasadit, nebo na vývoji pokračovat nezávisle na autoru této práce.

Práce je logicky rozdělena na dva velké celky. Prvním celkem je úvod do virtualizace a systému Xen, kde je v kapitolách 2 a 3 vysvětleno, jak Xen funguje a jaké jsou jeho možnosti a omezení. Kapitola 4 je z pohledu další práce důležitá, jelikož jde o jakési logické pojítko, které zasazuje poslední část práce do kontextu virtualizace, tedy první části. Ve druhé části práce kapitola 5 popisuje základní instalaci virtualního systému tak, aby bylo možné na něm provozovat modulární aplikaci pro jeho správu. Návrhem a rozborem této aplikace se zabývá hlavně kapitola 6, doplněná o popis vybraných částí implementace v kapitole 7. Při tvorbě každé aplikace vznikají nápady, co by se mohlo objevit v další verzi. Ani tento případ není vyjímkou. Některé nápady do budoucích verzí jsou naznačeny v kapitole 9.

Kapitola 2

Úvod do systému Xen

2.1 Terminologie

Virtualizace je technika, která slouží k abstrakci hardware počítače takovým způsobem, aby na něm mohlo být provozováno více operačních systémů najednou. Virtualizace má různé stupně abstrakce hardware počítače.

Plná virtualizace je jedním ze stupňů virtualizace. Je při ní simulována většina hardwaru počítače. Provozovanému systému se tak jeví, jako by byl provozován na reálném stroji. Proto zpravidla tento typ virtualizace neomezuje typ provozovaných operačních systémů.

Paravirtualizace je opět jedním ze stupňů virtualizace. Při použití této techniky však nemusí být nutně simulován hardware hostitelského počítače, jen je provozovaným systémům poskytnuto speciální programové API, za pomocí kterého mohou přistupovat k periferiím počítače. Nevýhodou tohoto systému je nutnost modifikovat provozované operační systémy tak, aby toto API mohly využívat. Oproti tomu výhodou je vyšší výkonnost.

Hypervisor je program, který je spuštěn při startu počítače, ještě před startem jádra spouštěného systému. Hypervisor řídí přepínání aktivity mezi paravirtualizovanými systémy a je zprostředkovatelem přístupu k hardware počítače.

Privilegovaný systém je operační systém, který je spolu s hypervisorem spouštěn při startu počítače. Jako jediný má privilegia řídit ostatní systémy (spouštět, zavírat, migrovat, či modifikovat parametry. Privilegovaných systémů může být na počítači víc, přičemž každý obsluhuje část hardware. Tato možnost se ale standardně nepoužívá, proto pro účely této práce může na počítači být pouze jeden privilegovaný systém. Je také označován jako *dom-0* (domain 0).

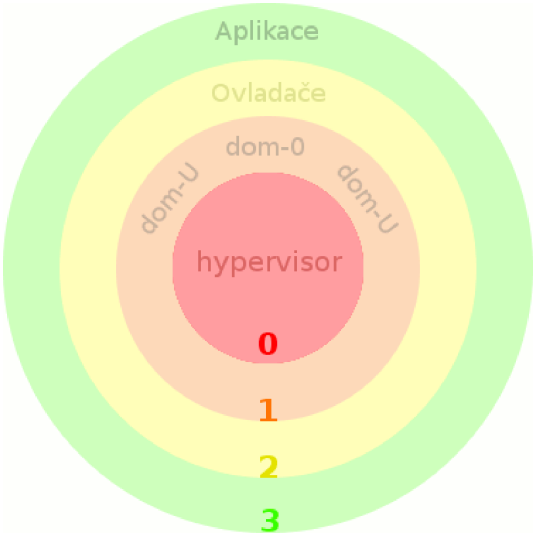
Neprivilegovaný systém je jakýkoliv jiný systém běžící na počítači. Je spouštěn z privilegovaného systému a nese označení *dom-U* (unprivileged domain).

Okruhy bezpečnosti jsou mechanismem k ochraně dat a funkcí před chybami a zlomyslným chováním. V tomto mechanismu existuje několik úrovní přístupu k prostředkům počítače. Každá tato úroveň je označena jako okruh. Čím nižší číslo okruhu, tím více privilegovaný přístup k prostředkům počítače lze v tomto okruhu získat. Pro účely této práce jsou okruhy podrobněji vysvětleny v části 2.2

Operační systém (dále jen OS) bude v kontextu této kapitoly mít význam shodný s významem výrazu *jádro OS*

2.2 Systém Xen

Xen je systém pracující primárně na principu paravirtualizace. V takovém režimu vypadá práce s okruhy tak, že v okruhu 0 je hypervisor a v okruhu 1 jsou všechny systémy – jak privilegovaný, tak neprivilegované. Tento režim se nazývá *Hypervisor mode* a liší se oproti běžnému režimu, nazývanému *Supervisor mode*. V takovém případě je totiž v okruhu 0 jádro samostatně provozovaného (nevirtualizovaného) systému. V okruhu 3 jsou v obou případech aplikace a v okruhu 2 mohou být například ovladače jednotlivých zařízení. Tento princip je však využíván jen zřídka. To samé platí u okruhu 1 v supervisor módu. Celý princip je graficky znázorněn schématu 2.1. [11, 9]



Obrázek 2.1: bezpečnostní okruhy paravirtualizovaného systému

Kromě paravirtualizace podporuje hypervisor Xen od verze 3.0 i plnou virtualizaci. K tomu ovšem potřebuje podporu virtualizace v procesoru. U procesorů Intel se tato technologie nazývá iVT, nebo též *Vanderpool*. AMD tuto možnost má označenou jako AMD SVM, nebo též AMD-V, či *Pacifica*. Při plné virtualizaci v podání Xenu vytváří procesor novou sadu bezpečnostních okruhů, na kterých může virtualizovaný systém pracovat. [11]

2.3 Privilegované operační systémy

2.3.1 GNU/Linux

Linux je jednoznačně nejpoužívanějším OS pro tento účel. V rámci projektu Xen je vyvíjena verze tohoto OS založená na vanilla kernelu, která se jmenuje XenLinux. Základem je aplikace patchů, které přidávají novou subarchitekturu. Ta umožňuje provoz tohoto OS jak v privilegovaném, tak neprivilehovaném režimu. Při kompilaci je potřeba si zvolit režim, ve kterém bude provozován. V současnosti (počátek roku 2008) je k dispozici XenLinux verze 2.6.18, což je verze dosti zastaralá. Někteří distributoři situaci řeší tak, že si upravují novější

verze vanilla kernelu po vzoru XenLinuxu. V distribucích Fedora a Debian je tak dostupný Linux verze 2.6.22.

Asi největší výhodou použití tohoto OS pro běh jako dom-0 je právě jeho podpora. Jelikož byl Xen pro tento systém vyvinut, je nejméně problémová jeho instalace, případně aktualizace jako privilegovaného systému.

Největší nevýhodou nasazení Linuxu byla až do verze Xenu 3.0 jeho výkonnost. Systému spuštěnému jako dom-0 klesala podle testů výkonnost až o 75% při provádění diskových operací. [8]. To se však s příchodem této verze Xenu změnilo a tento výkonnostní problém zanikl. [6].

2.3.2 NetBSD

Použití NetBSD jako privilegovaného systému je možné od jeho verze 4.0. Na provoz na 64-bitových počítačích je potřeba ještě novější, nejméně z prosince 2007.

Využití tohoto systému pro ISP jako privilegovaného je vhodné v případě virtualizovaného databázového serveru. V kombinacemi s databází MySQL tento systém vykazuje mnohem efektivněji vykonávané V/V operace.[4] Jinak tento systém v porovnání s Linuxem nevykazuje žádné další výhody, kvůli kterým by se jej vyplatilo využít jako privilegovaný.

Z nevýhod je nejvýraznější omezení diskového prostoru, na kterém může být privilegovaný systém provozován. Kvůli určitým vlastnostem grubu je potřeba, aby kořenový diskový oddíl dom-0 systému měl maximálně 512MB a byl typu FFSv1, 8k bloky/1k fragmenty. Při nedodržení těchto požadavků by mohlo dojít k chybě při načítání některých souborů. [1]

2.3.3 Solaris

Tento text se bude zabývat konkrétně verzí OpenSolaris. Okolo tohoto systému existuje část komunity, která se zabývá maximální funkčností OpenSolarisu na systému Xen. Projekt, na němž tato komunita pracuje, se nazývá xVM. V současnosti xVM umožňuje provoz Solarisu na Xenu jak v privilegovaném, tak v neprivilegovaném módu. Problémy však zatím mohou nastat při provozu neprivilegovaných systémů v situaci, kdy Solaris je spuštěn jako dom-0. Provoz Linuxu není garantován (resp. nebylo prováděno žádné rozsáhlejší testování) a systémy Windows buď nebudou fungovat, nebo budou velmi pomalé. Z těchto důvodů zatím Solaris pro provoz jako dom-0 systém nelze doporučit. [2]

Kapitola 3

Základy práce s Xenem

3.1 Instalace

3.1.1 Distribuční balíčky

Ve většině velkých distribucí GNU/Linuxu existují všechny balíčky potřebné k plnému zprovoznění Xenu. Problém je však s výkonností. Při testu distribučních balíčků systému Debian bylo zjištěno, že rychlost veškerých V/V operací klesla o tolik, že systém se stal prakticky nepoužitelným.

3.1.2 Automatická instalace ze zdrojových kódů

Mezi instalační balíky oficiálně nabízené na webu Xenu patří také univerzální instalační balíček pro GNU/Linux, kdy je vše instalováno ze zdrojových kódů. Celý proces je přitom automatický a je řízen dodávaným instalačním nástrojem. V případě instalace z takovéhoto balíku jsou po zadání příkazu `make world` nainstalovány všechny základní komponenty systému Xen – hypervisor, jádro pro privilegovanou doménu a Xen tools – sada nástrojů na základní ovládání celého systému.

Nevýhodou této instalace je, že některé nástroje v ní chybí. V distribučních balíčcích Debianu je totiž udržována sada nestandardních nástrojů, které mohou práci s Xenem značně zjednodušit. Na druhou stranu se nejedná o kriticky důležité nástroje a zkušený správce se obejde i bez nich.

Poněkud nepříjemné jsou některé komplikace s privilegovaným systémem. Během instalace totiž dochází ke stažení Linuxu ze stránek kernel.org, následné aplikaci patchů a kompilaci výsledného celku. Problematická je zde verze stahovaného Linuxu (nejnovější podverze 2.6.18), kdy není plně podporována velká množina hardwaru používaného na počítačích obsahujících například čipovou sadu i965 a novější. Větším problémem je ale nemožnost manipulace s jádrem. Například možnost spuštění make menuconfig před samotnou kompilací vyžaduje několik dosti odborných a ne vždy funkčních zásahů do instalačních scriptů.

Výhodou této instalace je její funkčnost na všech častěji používaných linuxových distribucích a její popis v dokumentaci Xenu.

3.1.3 Ruční instalace ze zdrojových kódů

Tato instalace částečně využívá předchozího způsobu. Prvním krokem je stáhnutí oficiálního instalačního balíku. Rozdíl spočívá ve spuštění příkazů `make tools` a `make xen` namísto `make world`. Tím zkompilujeme a nainstalujeme pouze hypervisor a nástroje pro administraci Xenu.

Linux samotný lze stáhnout z repositářů Debianu, případně Fedory. Zpravidla je v obou dostupná stejná verze. Rozdíl je v aplikaci patchů. V Debianu se jedná o aplikaci 3 patchů, kdežto u Fedory je jich několik desítek. Verze z repositářů Fedory se tak vyplatí pouze když je jádro patchováno právě v této distribuci, jelikož zde jsou dostupné nástroje na automatizaci tohoto procesu, pro které je celý balík s Linuxem uzpůsoben. Verze Debianu má v podadresáři `debian/binary-custom.d/xen/patchset/` soubory, které stačí podle jejich pořadových čísel aplikovat na zdrojový kód. Při kompilaci jádra je podstatné nastavit subarchitekturu na Xen. Nastavení lze najít v `menuconfigu` pod položkou:

```
Processor type and features --->
    Subarchitecture type --->
        Xen compatible
```

Tím se zpřístupní možnosti Xenu, umístěné v hlavním menu pod položkou `Xen`. Zde je důležité označit, že kompilovaný OS bude privilegovaný. Po kompilaci jsou v kořenovém adresáři jádra[1] umístěny soubory `vmlinux` a `vmlinux-stripped`. Je nutné použít jeden z těchto souborů, neboť u bzImage je problém s chybným formátem komprese a hypervisor by tak jádro nedokázal načíst. Je však možné soubor `vmlinux` zkomprimovat dodatečně, v takovém případě už se problém nenaskytne.

Když je připravené jádro i moduly k němu, je třeba přidat do konfiguračního souboru grubu záznam, kde jako `kernel` bude uvedena cesta k hypervisoru a cesta k jádru bude uvedena jako `module`. Kromě klasických parametrů jádru je možné přidat ještě několik dalších určených pro Xen. Jejich seznam je uveden v dokumentaci a na Wiki Xenu.

Po nastartování hypervisoru a privilegovaného jádra je možné nastartovat Xen démon. Ten je složen ze dvou částí. První z nich je HTTP server, který je standardně provozován na portu 8000 a přijímá XML-RPC požadavky ovládající neprivilegované systémy. Tento server je přímo napojen na knihovnu libxenctrl, přes kterou posílá požadovaná příkazy jádru. To je následně předá hypervisoru. [10] Druhou částí je sada scriptů, které pozmění nastavení v privilegovaném systému tak, aby bylo možné tam provozovat další systémy. Před spuštěním démona je však vhodné provést základní konfiguraci, kde je třeba minimálně nastavit požadovanou konfiguraci sítě.

3.2 Síť

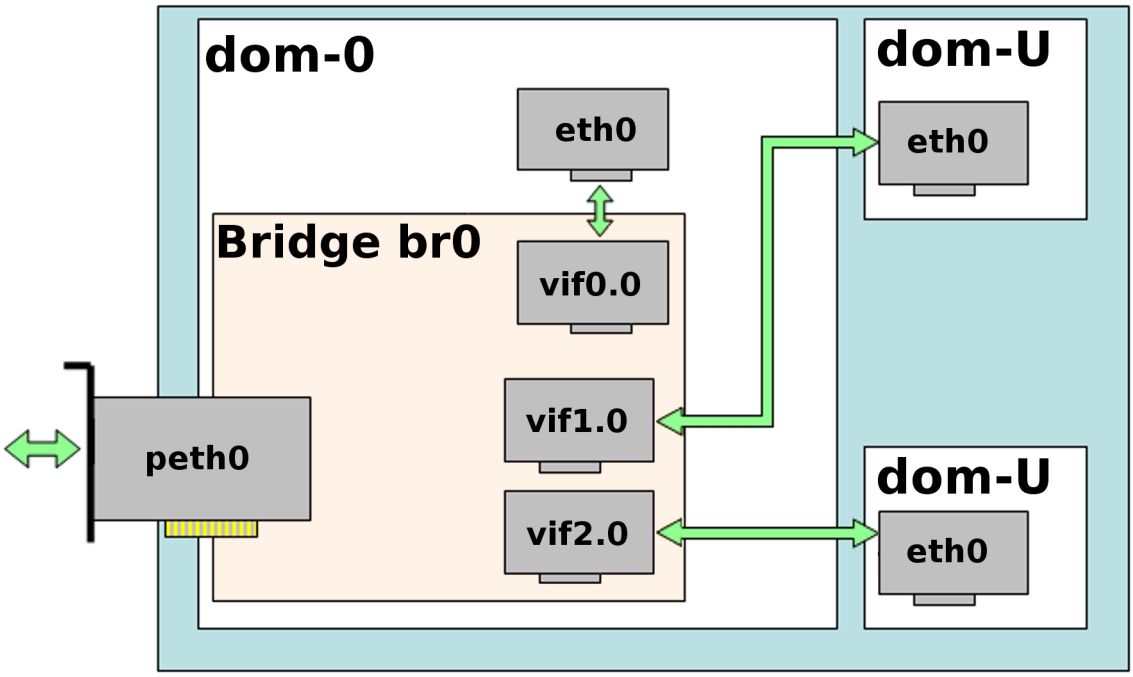
V syaby stému Xen (resp. v privilegovaném systému) jsou vytvářeny dvojice virtuálních síťových rozhraní, označené `vethx#` a `vifx.y`, kde `x` symbolizuje ID domény a `y` číslo síťového rozhraní pro danou doménu. Tyto dvojice jsou nakonfigurovány a jednotlivá rozhraní přidělena jednotlivým doménám podle režimu, na který je vnitřní síť Xenu nastavena. Systém Xen umožňuje dva režimy – *NAT* a *bridge*. Pro použití ISP není NAT vhodný, jelikož služby musí být obvykle dostupné na veřejné IP adrese. Proto tato práce bude zaměřena

[1] Adresář, do kterého byly rozbaleny zdrojové kódy Linuxu

na režim bridge. Aby bylo možné bridge provozovat, je třeba mít v systému nainstalované bridge utils a mít v jádru povolenou možnost vytvoření bridge.

Po spuštění démona Xenu dojde ke spuštění nastaveného scriptu, který pozmění stávající nastavení sítě v systému. Při bridge režimu je nejdříve vytvořen virtuální bridge. Síťové rozhraní `eth0` je přejmenováno na `peth0` a je přidáno do tohoto bridge. Pokud je v počítači více fyzických rozhraní, nebo jsou rozhraní pojmenována jinak než `eth#`, existují dvě možnosti, které z nich je použito pro komunikaci a je přidáno do bridge. Standardně je zvoleno to, přes které je v routovací tabulce nastaveno směrování na cíl `default`. Pokud je požadováno jiné, je možné jej specifikovat v nastavení démona Xenu. Přejmenování pak proběhne tak, že před název tohoto rozhraní je přidáno písmeno *p*. Původní název, IP a MAC adresu přejmenovaného rozhraní přebírá virtuální síťová karta `veth0`. Privilegovaná doména od této chvíle komunikuje se sítí přes rozhraní `veth0`.[3]

Při vytvoření neprivilegované domény je této doméně přiřazena jedna nebo více dvojic virtuálních rozhraní `vifX.Y` – `vethY`, kdy `vifX.Y` je vždy součástí bridge a `vethY` je zobrazeno neprivilegovanému systému. V tomto systému je však jeho název pouze `ethY`. Název `vethY` je pouze vnitřním názvem Xenu, aby bylo znatelné, že se jedná o virtuální rozhraní. Každé z rozhraní `vethY` může mít vlastní IP a MAC adresu. Celý systém je znázorněn na obrázku 3.1



Obrázek 3.1: síť v systému Xen

3.3 Grafický výstup

Pro práci s OS Windows je třeba, aby z neprivilegovaných domén byl umožněn grafický výstup. Xen oficiálně neumožňuje grafický výstup, kdy by neprivilegovaná doména měla přístup ke grafické kartě. Někteří lidé z komunity kolem Xenu sice pracují na experimentálních rozšířeních, kdy by bylo možné emulovat grafický hardware v případě, že by k počítači byly připojené dva displeje, ale i přes to je ono rozšíření nepoužívané, hlavně protože Xen nativně nabízí řešení jiné.

Tím řešením je VNC server. Po spuštění neprivilegované domény s Windows je automaticky spuštěn VNC server, ke kterému je možné se buď připojit na dálku po síti, nebo v případě přímého přístupu k počítači lze využít VNC klienta zabudovaného v Xenu. Vzhledem k povaze služeb ISP je pravděpodobnější připojení po síti.

3.4 Práce s disky

Každé neprivilegované doméně provozované pod Xenem je nutné do konfigurace zadat seznam diskových zařízení, které jí budou zprovozněny. Xen podporuje několik typů zdrojů mapovatelných na disková zařízení pro neprivilegované domény.

3.4.1 Disky mapované ze souboru

První možností je provoz ze souboru na souborovém systému, který je připojen v privilegované doméně. Tato možnost je však velmi pomalá zvláště protože všechny diskové operace je třeba provádět dvakrát – poprvé v neprivilegované doméně jako zápis na blokové zařízení a podruhé v privilegované doméně jako zápis do souboru.[2]

3.4.2 Disky mapované s fyzických zařízení

Druhou možností je provoz systému z normálního diskového oddílu. Zde je zpravidla doporučováno využití systému LVM2, jelikož ten podporuje řadu užitečných vlastností, jako je třeba zvětšování a zmenšování diskové oblasti.

3.4.3 Síťové disky

Poslední možností je provoz přes síťový disk. Tato možnost je zvláště výhodná z pohledu migrace virtuálních strojů – není třeba přesouvat data umístěná na disku. Také je zvláště výhodná při nutnosti sdílení nějakého diskového oddílu mezi doménami, když jedna z domén má k němu přístup pro zápis. Pokud je totiž ne-síťový diskový oddíl využíván ve více doménách, hrozí nebezpečí poškození diskového oddílu z důvodu nekonzistence metadat. Navíc Xen samotný má omezení, které nedovoluje do neprivilegované domény namapovat diskový oddíl pro zápis, pokud je již připojen v privilegované doméně, případně pokud je namapován do jiné neprivilegované domény (ať už je v ní připojen nebo ne). Podobně je zakázáno namapování oddílu, který je již připojen (resp. namapován), pro zápis. Toto omezení lze však překonat v konfiguračním souboru neprivilegované domény. Proto je doporučeno při nutnosti takového sdílení využít raději síťové disky. Podporovány jsou všechny standardní protokoly pro síťové souborové systémy.

[2] Na toto téma zatím nebylo provedeno mnoho měření, nicméně jedná se o obecnou zkušenost vývojářů Xenu. Jedno z podkladových měření lze nalézt na adrese `<http://lists.xensource.com/archives/html/xen-users/2006-08/msg00843.html>`

3.4.4 Řešení používaná v praxi

Jedno z vynikajících řešení prezentuje projekt zpracovaný na Západočeské univerzitě v Plzni, týkající se nasazení Xenu v univerzitním prostředí[3]. Architektura uvedeného virtualizovaného systému využívá diskové oddíly typu AFS připojené přes FiberChannel. Tato struktura umožňuje velmi jednoduchou migraci systémů ze serveru na server. Její nevýhodou je však složitější počáteční konfigurace a nezanedbatelné vstupní investice. Další řešení prezentují často využití lokálního sdíleného adresáře `/usr` připojeného pouze pro čtení. Tento systém je využívaný, jelikož výrazně šetří místo na disku.

3.4.5 Mapování odkládacích prostorů

Stejně jako běžné diskové oddíly, i SWAP je možné namapovat všemi předchozími způsoby. Práce s tímto diskovým oddílem je stejná jako s běžným odkládacím prostorem. Velký rozdíl je v umístění SWAPu na disku. Uživatel má totiž možnost namapovat jako odkládací oddíl soubor dostupný v privilegované doméně. Této možnosti je dobré se vyhnout z výkonnostních důvodů popsaných výše. Podobným problémem je využití stránkovacího souboru u systémů Windows. Tento problém se však více týká umístění systémové oblasti. Tato oblast je totiž standardně i oblastí, kde se nachází stránkovací soubor. Jednou z vlastností Windows, hlavně Windows XP, je velmi neefektivní prací s pamětí, kdy je stránkovací soubor využíván už při cca 60% zaplnění paměti. S přihlédnutím k výše uvedeným důvodům z toho vyplývá, že tak dochází k dlouhodobému snížení výkonu PC.[4]

3.5 RAM

Ve 32 bitovém systému Xen je standardně podporováno až 4GiB paměti RAM. Od verze 3.0 je však zprovozněn systém PAE, který umožňuje tuto hranici překonat. Od této verze je také umožněn provoz na 64 bitových procesorech, kde je podporováno až 1TiB paměti.

Paměťová omezení lze zadávat ve dvou rovinách. První z nich je nastavení maximální velikosti paměti pro privilegovanou doménu. Toto nastavení se provádí nastavením parametru dom0_mem jádra spouštěného jako privilegovaného.

Dále je možné nastavovat omezení paměti pro každou další spouštěnou doménu. To lze provést nastavením parametru memory v konfiguračním souboru domény. Toto nastavení je kdykoliv možné měnit za běhu domény příkazem `xm set-mem <domena> <pamet_v_MB>`. Běžící doméně lze paměť jak přidávat, tak ubírat. Uvnitř domény lze využít zápis do souboru `/proc/xen/baloon`, kterým můžeme velikost paměti pro doménu také měnit. Nastavení paměti lze shora omezit příkazem `xm maxmem`, který nastaví horní hranici velikosti paměti pro danou doménu.

[3]Projekt byl prezentován na přednášce v rámci akce BootCamp pořádané ČVUT. Záznam je dostupný na adrese `<http://www.avc-cvut.cz/avc.php?id=3590>`

[4]Testováno na systému VMWare, podobné problémy se však dají očekávat i u Xenu z důvodů popsaných v sekci 3.4.1

socket0				socket1			
core0		core1		core0		core0	
ht0	ht1	ht0	ht1	ht0	ht1	ht0	ht1
vCPU0	vCPU1	vCPU2	vCPU3	vCPU4	vCPU5	vCPU6	vCPU7

Tabulka 3.1: struktura virtuálních CPU

3.6 CPU

Xen poskytuje systém virtuálních procesorů podobný systému virtuálních síťových rozhraní. V praxi to vypadá tak, že každé doméně je možné povolit využití jednoho nebo více virtuálních procesorů. Samozřejmě je tomu třeba přizpůsobit jádro neprivilegovaného systému (např. povolit SMP, HT, aj.). Xen vytváří virtuální procesory podle skutečných procesorů dostupných na počítači, přičemž jeden virtuální procesor může být využit i více systémy. Tabulka 3.1 ukazuje, jak by Xen vytvořil virtuální procesory na dvou procesorovém systému s procesory Xeon podporujícími HyperThreading.

Systém práce s procesory je poněkud znehodnocen nefunkčností škálování výkonu procesoru.[5] Při kompilaci jádra je po zvolení architektury Xen automaticky zakázána celá sekce

```
Power management options (ACPI, APM)  --->
   Frequency Scaling
```

To samo o sobě není velký problém. Ten vyvstane až ve chvíli bootu privilegovaného systému. Pokud totiž procesor podporuje několik frekvenčních úrovní, je zvolena ta nejnižší z nich a není možné ji přenastavit. To je problém zvláště na procesorech zaměřených na použití v mobilních zařízeních, jelikož tam je minimální frekvence procesoru naprosto nevhodná pro provoz virtuálních strojů.[6].

[5] Důvody jsou uvedeny zde: <http://lists.xensource.com/archives/html/xen-devel/2006-10/msg01178.html>

[6] Bug reportován na <https://bugs.launchpad.net/ubuntu/+source/linux-source-2.6.22/+bug/137596>

Kapitola 4

Xen pro ISP

4.1 Terminologie

ISP je zkratka pro Internet Service Provider, neboli jakoukoliv firmu poskytující nějaký typ připojení k internetu

LAMP je běžně užívaná zkratka pro Linux + Apache + MySQL + PHP. Tato kombinace programů je velmi často používána na serverech, které mají sloužit alespoň částečně jako webové.

4.2 Služby poskytované ISP

Pod pojmem ISP lze rozumět několik typů firem poskytující služby přístupu na internet. První z nich je poskytovatel internetu běžným koncovým stanicím. Příkladem takové firmy může být jakýkoliv provozovatel bezdrátové sítě WiFi. Síť takového poskytovatele má obvykle jeden klíčový prvek, kterým je server sloužící jako brána do internetu. Tento v podstatě není třeba virtualizovat, proto pojednání o tomto typu ISP nebude předmětem této práce.

Podobnými prvnímu typu jsou provozovatelé sítí, na které se napojují další podnikové sítě. Až tyto sítě poskytují připojení koncovým stanicím. Tento typ ISP zpravidla nabízí svým zákazníkům doplňkové služby, které už má cenu virtualizovat – hlavně webový a emailový server. U větších zákazníků pak připadá v úvahu server pro groupware.

Naprosto typickým příkladem ISP je pak společnost poskytující webhostingové služby. Zde nachází virtualizace široké možnosti uplatnění. Nejčastějším příkladem jsou webové aplikace vyžadující vypnutá bezpečnostní opatření jako je *open_basedir* či povolení jistých nebezpečných funkcí (např. exec). Dalším častým použitím je vyžadování specifického nastavení některé z poskytovaných služeb, např. emailového serveru. Určitě také nelze opomenout poskytování virtuálních serverů samotných, jako ekonomicky výhodné alternativy k managed serverům a serverhousingům.

Posledním typem společností jsou pak ostatní ISP, kteří poskytují širokou škálu služeb, od telefonních ústředen, kde je nasazení Xenu typické v kombinaci s komunikační platformou Asterisk, až po společnosti nabízející různé druhy serverových služeb firmám, které nemají vlastní servery.

4.3 Virtualizovatelné služby

4.3.1 Virtuální webhostingy

Tento typ služeb bude pro účely této práce zastoupen virtualizovaným podáním klasického webhostingu využívající LAMP server, kde bude navíc provozován poštovní systém postfix. Existují dvě hlavní využití tohoto systému:

1. Přeprodávání služeb menším firmám, které si chtějí zřídit webhostingové služby, ale buď nechtějí ztracet čas se složitější administrací, nebo na to nemají zdroje (ať už lidské, finanční, nebo časové). V takovém případě je potřeba nabídnout hlavně kvalitní uživatelsky přívětivou administraci k celému systému, včetně nastavení jednotlivých služeb, založenou na PHP a databázi MySQL.

2. Provoz klasického webhostingu pro nějakou webovou aplikaci, která má požadavky na snížené bezpečnostní nastavení (např. vypnutý `open_basedir`). V tomto případě bude postačovat aplikace umožňující základní změny jako je admininistrace emailů a FTP účtů.

4.3.2 Služby vyžadující zvlášní nastavení

Mezi tyto služby lze zařadit většinou produkty určené primárně pro větší firmy. Tyto produkty se totiž zpravidla vyznačují tím, že nejsou připravené být provozovány v režimu webhostingu, tedy pro více domén najednou. Za celou velmi rozsáhlou skupinu lze uvést nejčastější zástupce, kterými jsou **groupware řešení**. Jsou to totiž zpravidla velmi komplexní řešení zasahující do všech částí systému. Kromě zásahů ve formě potřebné specifické konfigurace jsou také často nutné zásahy do nainstalovaných částí systému. Některé z těchto systémů lze zařadit i do následující skupiny, například *MS Echange*. Dále jako zástupce pro Linux lze uvést *Open Exchange*, *Zimbra*, či *Merak mail server*. Všechna tato řešení nabízí standardně instalaci celého systému jako celku. Další zástupci této skupiny mohou být z řad aplikačních serverů, jakými jsou *Lotus Domino* nebo *JBoss*.

4.3.3 Služby vyžadující různé OS

Tato skupina je reprezentována komerčními produkty, které poptávají středně velké firmy. Jedná se zpravidla o kombinace LAMP serveru s produkty, které nejsou portovány na Linux. Dále zde je zahrnuta kombinace Linux – NetBSD, kdy na BSD poběží databáze MySQL z důvodu lepšího výkonu a na Linuxu bude provozován zbytek služeb jako je emailový server a Apache.

4.4 Zvolený obchodní model

Aby bylo možné pokračovat dál, je nejdříve potřeba uvést obchodní model zvolený pro systém. Výše bylo uvedeno několik typů služeb, pro které je virtualizace běžně užívána. Cílem této práce bude pokusit se nastínit poněkud odlišný model, který zatím není ISP využíván. Tento model spočívá v nabízení Linuxového virtuálního serveru, kdy zákazník nebude mít přímo přístup k serveru. Tyto typy služeb již totiž existují a proto je prosazení pro nové ISP na tomto poli velmi obtížné.

Místo tohoto klasického řešení by měl systém nabízet možnost instalace modulů přes webové rozhraní. Na začátku by tedy zákazník dostal za nějaký základní poplatek virtuální

server, na němž by byla provozována pouze nějaká základní instalace www serveru Apache. Ten by umožňoval přístup k webovému administračnímu rozhraní. Přes něj by bylo možné nainstalovat například modul virtuálních serverů pro Apache, či pro Postfix, a tak provozovat virtuální webhostingy. Tento model poskytuje téměř neomezné možnosti v typu modulů, které je pro něj možné naprogramovat.

Kapitola 5

Virtualizované systémy

5.1 Terminologie

systém, operační systém bude v této kapitole představovat celek tvořený jádrem, knihovnami a dalšími programy potřebnými pro práci uživatele

hlavní doména je doménové jméno druhého řádu, které bude pro neprivilegovaný systém hlavní, to znamená, že pod ním poběží hlavní systémové aplikace, jako je webmail, PHPMyAdmin, statistiky. Standardně se bude jednat o doménu toho, kdo by virtuální server provozoval.

5.2 Příprava provozu neprivilegovaných systémů

Před začátkem tvorby neprivilegovaných systémů je potřeba připravit si jádro, na kterém tyto systémy poběží. Od verze *2.6.23* je ve vanilla kernelu implementována podpora provozu Linuxu v režimu neprivilegovaného systému. Praxe je však poněkud komplikovanější. XenLinux, původně vyvíjený pro provoz na Xenu, je, jak již bylo uvedeno výše, založený na *samostatné subarchitektuře*. Tento princip převzali i distribuce Debian a Red Hat. Oproti tomu vanilla kernel se snaží stejnou funkčnost implementovat přes *paravirt-ops*. Proto je potřeba při použití tohoto jádra provést poněkud odlišnou a ne úplně triviální konfiguraci. I po zprovoznění však následují další komplikace. Důsledkem výše uvedeného rozdílu je nepřipravenost jádra pro plnou podporu funkcionality Xenu a tak nefungují některé vlastnosti tohoto systému, např. *live migrace* a *uložení / pokračování*. Na odstranění těchto problémů v současnosti pracuje hlavně firma Red Hat a brzy by se měly dostat do hlavního stromu jádra patche umožňující funkčnost co nejvíce podobnou XenLinuxu, avšak založenou na *paravirt-ops*. I přes to je v současném stavu Vanilla kernel nepoužitelný. Z toho důvodu je dále využito stejné jádro, jako pro privilegovaný systém.[12]

Jak je uvedeno v kapitole 3.1.3, nejvýhodnějším způsobem instalace privilegovaného systému je manuální kompilace zdrojových kódů. Stejný postup je zvolen i zde. Z toho důvodu je vhodné nejdříve vytvořit kopii zdrojového kódu privilegovaného jádra. V konfiguraci tohoto upraveného jádra je potřeba vypnout následující položku:

```
XEN --->
    Privileged Guest (domain 0)
```

Po opětovné kompilaci je třeba si přejmenovat aktuální adresář s moduly jádra před tím, než nainstalujeme nové moduly.

```
mv /lib/modules/2.6.22.9 /lib/modules/2.6.22.9-dom0
make modules_install
mv /lib/modules/2.6.22.9 /lib/modules/2.6.22.9-domU
mv /lib/modules/2.6.22.9-dom0 /lib/modules/2.6.22.9
```

V adresáři `/lib/modules/2.6.22.9-domU` nyní máme připraveny moduly pro neprivilegované systémy. Jejich využití bude popsáno později. Dále je potřeba zkopírovat jádro do adresáře `/boot`. Stejně jako v případě privilegovaného systému je třeba využít nekomprimovaný obraz. Detaily jsou uvedeny v sekci 3.1.3.

5.3 Obsluha neprivilegovaných systémů

Základním nástrojem pro obsluhu neprivilegovaných systémů je příkaz `xm`. Ten slouží pro posílání příkazů Xenu. Základními příkazy potřebnými v kontextu této práce jsou:

- `create` Tento příkaz vytvoří a spustí neprivilegovaný systém buď podle konfiguračního souboru, nebo podle zadaných parametrů
- `destroy` okamžitě ukončí provoz systému a smaže ho ze seznamu provozovaných systémů
- `reboot` restartuje systém
- `shutdown` ukončí provoz systému
- `start` zapne systém, jehož provoz byl dříve ukončen příkazem `shutdown`
- `list` vypíše seznam provozovaných systémů
- `console` Tento příkaz se připojí na konzoli k neprivilegovanému systému. To může být potřebné při opomenutí konfigurace sítě, případně při chybě během práce se sítí.

Přesný popis příkazů lze najít v [5].

Jak je uvedeno výše, Xen se při vytváření neprivilegovaných systémů řídí primárně konfiguračními soubory. Při zadání příkazu `xm create <system>` je nejdříve prohledáván aktuální adresář. Pokud se v něm vyskytuje soubor, jehož název je shodný s parametrem `<system>`, je považován za konfigurační a Xen načítá informace o spouštěném systému z něj. Pokud daný soubor neexistuje v aktuálním adresáři, je prohledáván adresář `/etc/xen`. Pokud soubor s daným jménem není nalezen ani tam, je zaklášena chyba. Pokud konfigurační soubor není vytvořen, je možné využít `/dev/null`. V takovém případě ale musí být kompletní konfigurace uvedena v dalších parametrech příkazu. Pro funkční systém je potřeba nastavit následující parametry systému:

- **cesta k jádru** Je potřeba, aby se vůbec mohl systém nastartovat. Soubor s jádrem je v adresářovém stromu privilegovaného systému
- **omezení paměťi** Jedná se o horní omezení, které nesmí systém přesáhnout. Lze ho měnit i za běhu systému příkazem `xm max-mem`

- **jméno systému** Také by se dalo říct, že je to identifikátor, který slouží k práci se systémem
- **síťová rozhraní** Zde jsou zadána nastavení všech rozhraní, které má mít systém. Tato rozhraní jsou při startu systému přidána do bridge a propojena se systémem
- **disky** Podobně jako u síťových rozhraní, zde se jedná o virtuální bloková zařízení, která jsou neprivilegovanému systému zviditelněna.
- **kořenový filesystém** je předán jako parametr spouštěnému jádru, aby vědělo, na kterém zařízení jej má hledat
- **definice konzole** je třeba uvést, aby bylo možné se připojit na konzoli daného systému příkazem `xm console`.

Přesný popis parametrů lze najít v [5].

5.4 Šablony neprivilegovaného OS

V souladu s výše uvedenou specifikací služeb provozovaných na neprivilegovaném systému bude operačním systémem GNU/Linux, přesněji distribuce *Ubuntu*. Ta byla zvolena z několika důvodů. Je založena na populární serverové distribuci Debian, takže je pro ní dostupný dobře vyladěný software. Také je možné ji instalovat pomocí softwaru *Debootstrap*, který celý proces výrazně zjednodušuje. Hlavním důvodem však je Enterprise verze tohoto systému, ke které je nabízena profesionální podpora od společnosti Canonical. Navíc se jedná o velmi oblíbenou distribuci, jejíž popularita stále roste, takže má i bohatou uživatelskou základnu. To může být užitečné zvláště když je potřeba technická podpora pro systém, který ji nemá zakoupenou.

Jak již bylo řečeno, šablona bude vytvořena pomocí systému Deboostrap, který umožňuje nainstalovat jakýkoliv systém založený na Debianu do libovolného adresáře. Instalace bude prováděna na samostatný diskový oddíl typu ext3. Předpokládejme, že tento oddíl je označen `/dev/sda3` a je připojen na `/mnt/domu`. Instalaci systému provedeme příkazem:

```
debootstrap gutsy /mnt/domu http://archive.ubuntu.com/ubuntu
```

Dále je třeba provést základní konfiguraci systému:

- Nakopírovat adresář s moduly do neprivilegovaného systému:

  ```
  cp -R /lib/modules/2.6.22.9-domU /mnt/domu/lib/modules/2.6.22.9
  ```

  Přitom předpokládáme, že v adresáři `/lib/modules/2.6.22.9-domU` jsou připraveny moduly pro neprivilegované jádro.

- `chroot /mnt/domu`, následně pak změnit heslo uživatele *root* příkazem `passwd`
- Do souboru `/etc/fstab` přidat disky, které budou neprivilegované doméně přístupné (viz. sekce 3.4), dále je třeba také přidat řádek `proc /proc proc defaults 0 0` pro připojování adresáře se zařízeními.

  Pokud se počítá s využitím swapu, je možné jej připojit pomocí řádku

```
/dev/sda4 none swap sw 0 0
```

Tento zápis předpokládá, že swap byl v konfiguračním souboru domény namapován na `/dev/sda4`.

- Do souboru `/etc/hosts` přidat IP adresy známých systémů
- Do souboru `/etc/resolv.conf` vyplnit adresy nameserverů. Při daném nasazení lze předpokládat, že na síti není provozován DHCP server, který by toto nastavení určil.
- Vyplnit nastavení sítě v `/etc/networking/interfaces` (více o sítích v sekci 3.2)
- Vyplnit nastavení repozitářů podle stránky `<http://ubuntuguide.org/wiki/Ubuntu:Gutsy>`

Aby bylo možné provozovat administraci systému, je potřeba nainstalovat www server Apache a interpret jazyka PHP. Jelikož zvolíme jejich instalaci ze zdrojových kódů, jsou jejich prerekvizitami základní nástroje pro kompilaci, které nainstalujeme pomocí příkazu

```
aptitude install libssl-dev gcc make libxml2-dev libxml2
libcurl4-openssl-dev libcurl4-openssl-dev libjpeg62 libjpeg62-dev
libpng12-0 libpng12-dev libfreetype6 libfreetype6-dev libmhash2
libmhash-dev libmysqlclient15-dev libmysqlclient15off libxslt1.1
libxslt1-dev
```

5.5 Instalace Apache a PHP

5.5.1 Kompilace Apache

Přestože balíčkovací systém APT obsahuje v repozitářích Ubuntu server Apache i interpret PHP, zvolíme za instalační metodu kompilaci ze zdrojových kódů. Důvodem jsou větší možnosti ve výběru povolených a zakázaných součástí obou programů. Zejména některá méně používaná, nebo novější rozšíření není možné jinak než kompilací nainstalovat. Zdrojové kódy Apache lze získat ze stránek `<http://httpd.apache.org/download.cgi>`. Pomocí následující série příkazů archiv stáhneme, rozbalíme a nainstalujeme

```
cd /root
wget http://apache.mirror.superhosting.cz/httpd/httpd-2.2.8.tar.gz
mkdir install
mv httpd-2.2.8.tar.gz install
cd install
tar -xf httpd-2.2.8.tar.gz
cd httpd-2.2.8
./configure <konfiguracni_parametry>
make
sudo make install
```

Příkaz `./configure`, který provádí konfiguraci, má velké množství parametrů, kterými lze ovlivnit možnosti instalace. Zde bude výčet a vysvětlení těch nejdůležitějších z nich:

- `--prefix=<cesta>` udává instalační cestu. Tu nastavíme na `/usr/local/apache2`. Tímto způsobem bude v budoucnu velmi jednoduché provádět aktualizace, kdy stačí nové verzi dát jiný instalační adresář, starou verzi vypnout a novou zapnout.
- `--enable-authn-dbd` povoluje autorizaci pomocí SQL – to se může hodit na zabezpečení některých aplikací, například statistik AWStats
- `--enable-ssl` umožňuje komunikaci přes SSL/TLS
- `--enable-dav` umožňuje provozovat WebDAV server, který je vhodný například pro sdílení kalendářů jako alternativa komerčních technologií
- `--enable-rewrite` povoluje modul pro přepisování URL stránek. Jeho využívání je v posledních letech trendem hlavně při optimalizaci stránek, proto je webmastery vyžadována jeho přítomnost
- `--enable-so` toto je důležitá direktiva, která umožňuje nahrávání externích dynamických knihoven jako modulů Apache. To umožňuje provoz PHP jako modulu www serveru.

Jelikož byl Apache instalován do nestandardního adresáře, je vhodné umístit odkaz na jeho spouštěcí soubor do některého z adresářů v proměnné $PATH. Toho docílíme například příkazem:

```
ln -s /usr/local/apache2/bin/apachectl /usr/local/bin/apachectl
```

5.5.2 Kompilace PHP

Stejně jako u Apache, zvolíme i zde kompilaci ze zdrojových kódů. Hlavním důvodem je nepřítomnost rošíření *mysqli* v repozitářích systému. Budeme tedy instalovat nejnovější verzi (5.2.5), která je dostupná na stránce `<http://www.php.net/downloads.php>`. PHP nainstalujeme pomocí následující sady příkazů:

```
cd ~/install
wget http://cz.php.net/get/php-5.2.5.tar.bz2/from/this/mirror
tar -xf php-5.2.5.tar.bz2
cd php-5.2.5
./configure --prefix=/usr/local/apache2/php
--with-apxs2=/usr/local/apache2/bin/apxs --with-openssl --with-curl
--enable-exif --enable-ftp --with-gd --with-gettext --with-mhash
--with-mysqli --with-mysql --with-xsl --with-jpeg-dir --with-png-dir
--with-freetype-dir
make
make install
```

Nyní vysvětlení ke konfiguračním parametrům PHP:

- `--prefix` udává cestu, kam se PHP nainstaluje. Zde je instalováno do podadresáře `apache2` z čistě praktických důvodů. Celý server je tak na jednom místě, což vede k lepšímu provádění případných aktualizací

- `--with-apxs2` udává cestu k programu, který se stará o obsluhu externích modulů Apache
- `--with-openssl --with-curl --enable-exif --enable-ftp --with-gd --with-gettext --with-mhash --with-mysqli --with-mysql --with-xsl`

  aktivují některá často používaná rozšíření PHP, která nejsou aktivována standardně
- `--with-jpeg-dir`, `--with-png-dir`, `--with-freetype-dir` aktivují některá rozšíření knihovny GD, která nejsou instalována standardně

Mezi příkazy `make` a `make install` lze provést ještě příkaz `make test`, který provede několik tisíc testů, jestli PHP funguje správně. Pokud najde chybu, je možné ji odeslat programátorům, aby mohla být odsraněna.

5.5.3 Předpoklady konfigurace

Konfigurace Apache se dělí na dvě hlavní části. Základní částí je hlavní webový server, který je nastaven vždy. To znamená, že Apache obsluhuje požadavky přicházející na IP adresu, resp. IP adresy počítače a směruje je na jedno místo. Pokud chceme, aby některé adresy, ať už doménové nebo IP, byly směrovány jinam, je potřeba založit tzv. *virtuální server*, k čemuž lze využít konfigurační direktivu `VirtualHost`. Důležitou vlastností virtuálního serveru je, že dědí konfiguraci hlavního webového serveru. Jednotlivé části konfigurace lze přepsat pro každý server zvlášť.[7]

Nejdříve je třeba si určit některá pravidla, která budou využita při další konfiguraci:

- předpokládáme existenci jediné IP adresy, která je v systému dostupná
- z důvodu lepší konfigurovatelnosti bude www server provozován v režimu *jmenných virtuálních serverů*
- jelikož budou na systému provozovány jmenné virtuální servery nebude standardně provozována SSL komunikace
- každý zprovozněný virtuální server bude mít svůj konfigurační soubor generovaný z databáze a umístěný ve výše zmíněném adresáři `vhosts`
- administrace systému bude umístěna v adresáři `/var/www`

5.5.4 Konfigurace

Jelikož Apache bude provozován v režimu jmenných virtuálních serverů, odpadá část konfigurace hlavního serveru. Úlohu hlavního serveru přebírá virtuální server uvedený v konfiguraci jako první v pořadí.[7] Nastavení tohoto serveru se budeme věnovat později. Zbývá nám tedy nakonfigurovat část hlavního serveru, která je všeobecná. Při této konfiguraci využijeme existujícího souboru `httpd.conf`. V něm můžeme velkou část originálního nastavení nechat, zde se budeme věnovat jen těm položkám, které je třeba změnit. Dále je třeba smazat kontejner označený jako `<Directory /usr/local/apache2/htdocs>`, který se stává zbytečným díky použití virtuálních serverů. Dále je možné odstranit scriptalias na `cgi-bin` a kontejner `<Directory /usr/local/apache2/cgi-bin>`. Tyto se věnují CGI scriptům, které nebudou na serveru provozovány. Nyní zbývá na konec souboru přidat direktivy pro zprovoznění výše uvedeného modelu virtuálních serverů:

Parametr	Hodnota	Pozn.
`ServerAdmin`	`jmeno@hlavnidomena.tld`	zde bude uvedená emailová adresa provozovatele systému, jakožto hlavního administrátora
`User`	`www`	uživatel, pod kterým www server poběží
`Group`	`www`	skupina, pod kterou www server poběží
`DirectoryIndex`	`index.html index.php`	Pokud nebude v HTTP požadavku uvedeno jméno souboru v požadovaném adresáři, použije se jedno z těchto (přednost má dříve uvedené).
AddType	`application/ x-httpd-php .php`	Souborům s příponou `.php` přiřadíme typ *PHP aplikace*. Tím se stanou zpracovatelné intepretem PHP při příchodu HTTP požadavku na ně. Tuto direktivu je třeba přidat do kontejneru `<IfModule mime_module>`

Tabulka 5.1: nastavení hlavního serveru

```
NameVirtualHost *
Include conf/main_domain.conf
Include conf/vhosts/[^.#]*
```

První direktiva zapíná režim virtuálních serverů, další includuje soubor `main-domain.conf`, kde bude uvedena konfigurace virtuálních serverů hlavní domény. Poslední direktiva postupně includuje všechny soubory z adresáře `vhosts`, tzn. všechny soubory s virtuálními servery. To je příprava na budoucí možnost rozšíření funkčnosti Apache.

Provoz administrace

Administrace bude provozována pod standarním virtuálním serverem. Je však potřeba pamatovat na to, že můsí být umístěna tak, aby bylo možné umístit v pořadí „před ni“ konfiguraci hlavní domény. Webová prezentace hlavní domény bude dostupná jako modul systému, to znamená, že nebude standardně dostupná. Konfigurace virtuálního serveru pro provoz administrace bude vypadat podle následujícího příkladu. V příkladu se předpokládá provoz serveru na adrese `192.168.0.201`, administrace umístěná na adrese `http://server1.zelenyweb.cz` a email provozovatele serveru `zakaznikuv@email.cz`.

```
<VirtualHost 192.168.0.201>
    # povinne nastaveni
    ServerAdmin     zakaznikuv@email.cz
    ServerName      server1.zelenyweb.cz
    DocumentRoot    /var/www
    <Directory /var/www>
        AllowOverride All
    </Directory>
</VirtualHost>
```

První část je povinná pro každý virtuální server. Následuje kontejner `Directory`, který říká, že v souborech `.htaccess` bude možné měnit konfiguraci virtuálního serveru pomocí všech konfiguračních direktiv povolených serverem.

Kapitola 6

Modulární aplikace pro administraci systému

6.1 Terminologie

APT neboli Advanced Packaging Tool je nástroj pro správu software nejčastěji používaný v linuxových derivátech distribuce Debian. Nástroj je nástavbou balíčkovacího systému *dpkg*. Umožňuje balíčky automaticky stahovat z centrálních úložišť, zvaných *repozitáře*.Při instalaci kontroluje vztahy mezi balíčky a na základě těchto vztahů upravuje proces instalace tak, aby nedošlo k nekonzistenci systému.

repozitář označuje v následujícím kontextu komplex úložiště balíčků skládající se ze dvou částí, popsaných v kapitole 6.4.1.

balíček v kontextu navrhovaného systému tento výraz reprezentuje modul administrace systému

souborová část repozitáře , dále označovaná jako SČR, je strom adresářů a souborů uvnitř souborového systému v privilegovaném systému. Tento strom je dostupný administračním aplikacím provozovaným na neprivilegovaných systémech

dependency hell je neoficiální termín pro stav, kdy jsou mezi sebou balíčky v systému závislé tak, že není možné provést požadovanou operaci, protože by se tím poškodily závislosti a bylo by velmi obtížné (pokud vůbec možné) tyto závislosti dát do pořádku. Dalším případem tohoto problému je kruhová závislost balíčků.

UMSC je zkratka pro anglický termín *Universal Modular System Core*, tedy pro jádro univerzálního modulárního systému. Touto zkratkou bude v dalším textu označováno (resp. pojmenováno) jádro aplikace pro administraci systému.

Aplikace, administrace pokud nebude uvedeno jinak, tyto dva pojmy označují jádro UMSC doplněné o moduly. Tato aplikace tak může ovlivnit systém, aby jej bylo možné používat pro určitý účel.

6.2 Princip činnosti

Administrace bude provozována na každém neprivilegovaném systému zvlášť. Jelikož uživatel nebude mít přímý přístup k systému, není potřeba aplikaci speciálně zabezpečovat proti vniknutí do zdrojových kódů. Do administrace bude mít provozovatel systému přístup přes subdoménu poskytovatele. Takovou subdoménou může být například *vserver1.zelenyweb.cz*. V základním provedení bude administrace v podstatě jen webové rozhraní k UMSC, které umožní instalovat další moduly. Instalováním modulu přibude do menu administrace položka, pod kterou bude dostupné nastavení modulu. Do administrace bude standardně jedna sada přístupových údajů, bude však možné vytvořit modul umožňující přidání dalších uživatelů, kteří budou mít kontrolu nad částí systému, případně nad celým systémem.

UMSC se bude hlavním principem podobat balíčkovacímu nástroji APT známému hlavně ze systémů odvozených z linuxové distribuce Debian. Jednotlivé moduly tak budou dostupné ve formě balíčků, které budou stahovány z privilegovaného systému. Na něm také bude provozována systémová databáze. Tato databáze by měla převzít roli repozitáře z APT. Repozitář v podání této databáze by však měl rozšířené možnosti. Balíčky by byly samozřejmě jeho nedílnou, avšak nikoliv jedinou součástí. Částečně by přebíral i funkci lokální databáze balíčků. Další jeho podstatná úloha by spočívala v uložení informace o tom, zda má uživatel možnost daný modul koupit a provozovat. S tím by souvisela celá řada úkolů, která však nebude předmětem této práce. Jedná se totiž vesměs o úkoly administrativní. V této práci bude věnována pozornost hlavně technické stránce. O té bude pojednávat sekce 6.4.1.

6.3 Moduly systému

6.3.1 Modul jako balíček

V předchozím vysvětlení bylo nastíněno, jak budou vypadat moduly administrace. Nebyla však vysvětlena podstata principu činnosti, tedy podstata existence a využití balíčků. Balíček si lze představit jako rozšíření administrace o možnost konfigurace nějaké části systému. Zkráceně tedy lze říci, že platí **balíček = modul administrace**. Aby tento modul fungoval, musí obsahovat části popsané v kapitole 6.3.2.

6.3.2 Činnosti pokryté balíčkem

Jelikož by část systému, kterou modul ovládá, byla zbytečná právě bez možnosti její konfigurace, obsahuje balíček (resp. UMSC) i nástroje pro instalaci (a samozřejmě také odstranění) dané části systému. Zde je v návrhu využita spolupráce s balíčkovacími nástroji systému, tedy v našem případě s APT. Tím je zúžena množina činností, které musí navrhovaný systém ovládat – není třeba se starat o instalaci programů samotných. Hlavním účelem balíčků tak zůstává rozšíření administrace o možnost konfigurace konkrétní služby.

Aby bylo možné přidat do administrace možnost konfigurace dalšího modulu, je potřeba vyřešit několik základních problémů. Při instalaci balíčku je nejdříve potřeba nainstalovat službu, která má být balíčkem ovládána (pokud již není nainstalována). Tento problém je, jak už bylo uvedeno, vyřešen významnou spoluprácí s balíčkovacími nástroji systému, kdy budou při instalaci (resp. odstranění) balíčku pouze volány příkazy typu `aptitude install` a `aptitude purge`.

Dále je potřeba připravit nějakou základní konfiguraci pro tuto službu, jelikož standardní konfigurace zpravidla nevyhovuje potřebám zákazníka. Tuto základní konfiguraci budou do

souborů zapisovat scripty, které budou také volány při instalaci PHP funkcí `exec`. Tyto scripty mohou být v libovolném jazyce. Důležité je, aby interpret tohoto jazyka byl instalován v daném systému. Standardně by například bylo vhodné používat PHP, jalikož jeho přítomnost v systému je v podstatě jistá.

Jednou z nejdůležitějších částí je uložení konfigurace, kterou si uživatel přes webové rozhraní nastaví. Nejvýhodnějším úložištěm z hlediska jednoduchosti manipulace s daty je databáze, v našem případě MySQL. Aby bylo možné konfiguraci ukládat, je pravděpodobně potřeba, aby při instalaci balíčku byly vytvořeny určité databázové struktury. Ty by měly být po odebrání balíčku zase odstraněny, aby data nebyla na disku umístěna zbytečně.

Jelikož ne všechny systémové služby podporují načítání konfigurace z databáze, měl by balíček obsahovat nástroje, které po instalaci budou umožňovat generování konfiguračních souborů z dat uložených v MySQL. Bude se opět jednat o scripty podobné těm výše popsaným.

Hlavní částí balíčku však bude rozšíření webové administrace o možnosti nastavení dané služby. Takové rozšíření se bude skládat ze scriptů, které budou pravděpodobně obsahovat definice tříd a funkcí umožňující nějakou činnost. Typicky touto činností bude modifikace dat uložených v databázi. Samozřejmě nesmí také chybět uživatelské rozraní k takovému modulu. Poslední součástí je přidání položky menu do existující administrace.

6.3.3 Existující systémy

Když je řeč o balíčcích, nabízí se otázka, proč nevyužít již existující balíčkovací systémy a nástroje. Důvodů je hned několik. Prvním důvodem je to, že ačkoliv se v této práci mluví o balíčcích, jsou to stále jen moduly webové aplikace. Do principu těchto modulů byly pouze zavedeny některé prvky známé z balíčkovacích systémů, aby byl systém efektivnější. Balíčkovací nástroje typu *dpkg* tak nepředstavují úplně ideální řešení, jak moduly webové administrace instalovat. Proto jsou pouze využity jako nástroj využívaný jádrem UMSC.

Druhým důvodem je absence jistých omezení. Systém je navržen za účelem budoucího komerčního využití, kdy si provozovatel, tedy ISP, bude určovat, za které služby se bude platit. Implementace vlastního systému tak lépe integruje tyto omezení. Samozřejmě by bylo možné omezení implementovat i při použití externích nástrojů, nicméně chyběly by některé možnosti, jakými je platnost zakoupené licence pouze na některé verze programu. Ty by se řešily jen velmi obtížně.

Posledním důvodem jsou ne zcela vyhovující vlastnosti nástroje *dpkg*. Ten podporuje spuštění dvou scriptů během provádění operace s balíčkem – script provedený před akcí a script provedení po ní. V navrhovaném systému však potenciálně bude třeba řešit komplexnější problémy, kde je nutné využití jemnějšího dělení. Také je pravděpodobné, že instalace modulu administrace bude potřebovat více scriptů než jen dva.

6.3.4 Typy balíčků

Balíčky se dají dělit několika způsoby. Jeden ze základních pro pochopení balíčků jako takových je dělení podle jejich obsahu a účelu. Těmi nejpodstatnějšími jsou *moduly administrace*. Ty si lze představit jako sadu scriptů přidávající do administrace možnost upravovat nastavení některé ze systémových služeb. Systém dokonce umožňuje situaci, kdy už žádné další balíčky nejsou potřeba – v takovém případě by například v tomto balíčku měl být i script instalující službu, která je modulem ovládána. Dále by také měly být přítomny například SQL scripty, nebo různé komunikační a monitorovací scripty.

Výhodnější však je, když výše popsaný celek rozdělíme do více částí, které budou instalovány samostatně. Tím využijeme potenciál navrženého systému. Základní typy balíčků tak mohou být tyto:

Modul administrace Je hlavním typem balíčku. Tento balíček obsahuje část rozhraní administrace, která má být přidána. Při aplikaci požadavků uživatele přitom využívá volání, které poskytují scripty v jednotlivých knihovnách. Nezbytnou součástí balíčku je také přidání položky do stávajícího menu administrace.

Knihovna UMSC Jedná se o PHP script poskytující nějaké služby. Těmi může být například zápis statistických dat, kdy je script přidán do cronu a při každém spuštění zapíše do databáze údaje, ze kterých později může být generován graf. Druhým, zcela odlišným příkladem, je knihovna pro abstrakci připojení do databáze MySQL, která nabízí abstrakci rozhraní mysql a mysqli.

Obsluha služby Reprezentuje nějakou část konfigurace dané služby. Například pro službu FTP server se může jednat o možnost načítání uživatelů z databáze, nebo systémová kontrola uživatelských kvót. Balíček tohoto typu tak při instalaci typicky upraví konfigurační soubory, nainstaluje databázové struktury a připraví scripty pro generování konfiguračních souborů z databáze. Konkrétně vždy záleží na typu služby.

Systémová služba je v podstatě balíček, jehož instalace je v podstatě jen zavolání příkazu pro instalaci systémové služby (např. FTP server, emailový server, ...)

Příklady modulů jednotlivých kategorií budou uvedeny v kapitole 8.

6.4 Repozitář balíčků

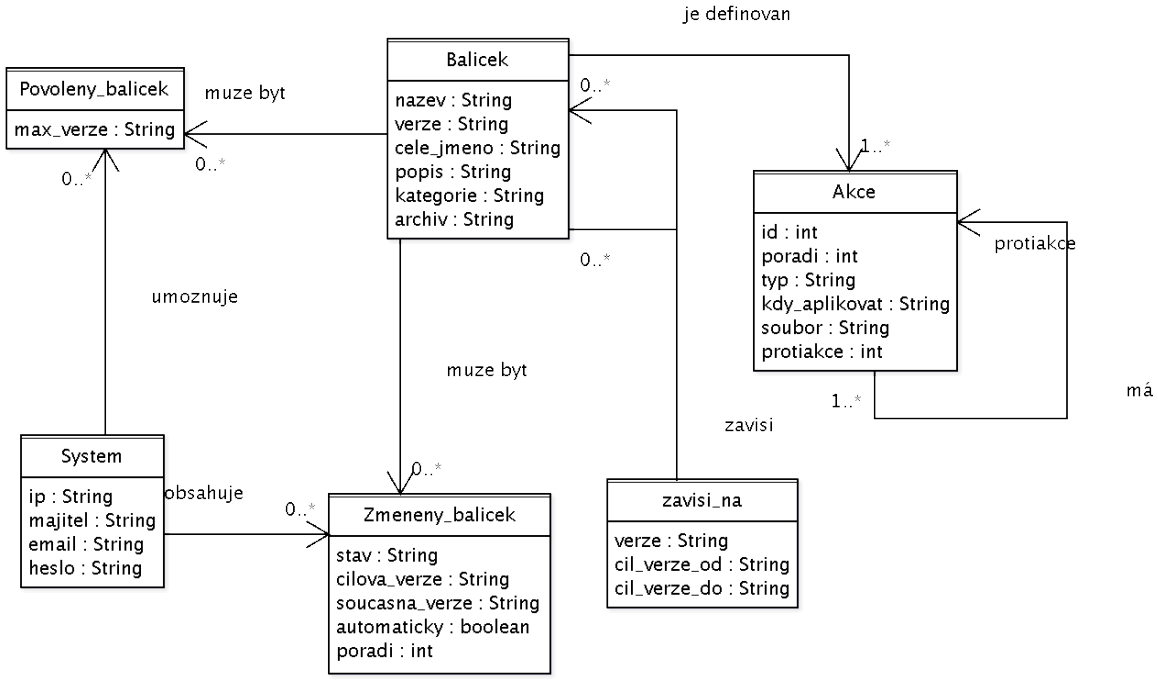
6.4.1 Funkce

Jak již bylo uvedeno, repozitář balíčků bude zastupován databází provozovanou na privilegovaném systému. Tato databáze v podstatě může být provozována na jakémkoliv relačním databázovém systému, v našem případě bude využit databázový server *MySQL* a v něm ukládací engine *InnoDB*, který podporuje netriviální vlastnosti SQL, jakými jsou např. vzdálené klíče a databázové triggery.

Kromě databáze bude mít repozitář ještě jednu část a sice úložiště systémových balíčků (SČR). Požadavky na toto úložiště jsou dva – zabezpečení přístupu, aby nebylo možné si data stáhnout odkudkoliv, a snadný přístup k datům pomocí PHP scriptování. Hlavní možnostmi tak je přístup přes webové rozhraní (resp. přes PHP script přístupný přes toto rozhraní) a přes FTP. Zvolen bude první přístup. Jeho jedinou nevýhodou je nutnost provozovat www server na privilegované doméně. To lze však předpokládat, jelikož na ruční administraci repozitáře bude vhodné použít aplikaci *PHPMyAdmin*, na jejíž provoz bude www server také potřeba. Nespornou výhodou webového rozhraní oproti FTP jsou obrovské možnosti, jakým způsobem kontrolovat autorizaci přístupu. V tomto případě bude nasazena kontrola přes IP adresu systému, ze kterého se o balíček žádá. Pokud IP adresa bude mezi známými systémy, bude balíček poskytnut. Funkce SČR bude založena na již zmiňovaném scriptu. Ten bude umístěn v kořenovém adresáři SČR. Při požadavku na nějaký balíček zkontroluje v databázi repozitáře, zda požadavek přišel z autorizovaného systému. Pokud tomu tak je, získá script z této databáze ještě informaci o tom, kde, vzhledem ke svému

umístění, se nachází soubor tohoto balíčku. Tento soubor poté pošle jako odpověď žadateli o balíček.

Hlavní část repozitáře bude v databázových strukturách. Na obrázku 6.1 je znázorněn relativně jednoduchý ER diagram modulárního systému. Jednotlivé části tohoto diagramu jsou popsány v následujících sekcích.



Obrázek 6.1: ER diagram repozitáře

6.4.2 Struktura balíčku

Struktura balíčku obsahuje dvě části. První z nich je databázová struktura zastupující balíček. Ta je vždy určena názvem balíčku a jeho verzí. U té je předpokládán standardní formát používaný u verzování programů, tedy `xx.xx.xx`, kdy `x` reprezentuje číslici od 0 do 9. Balíček může být členem jedné z kategorií. Tyto kategorie slouží jednak k lepší orientaci a jednak k určitému omezení, které je vysvětleno v tabulce 6.4.2. U každého balíčku je také uvedena cesta k souboru, ve kterém je daný balíček umístěn. Cesta je uvedena relativně ke kořenovému adresáři SČR.

Druhou částí je archiv obsahující soubory potřebné k instalaci balíčku. Archiv balíčku bude v souboru ZIP, jelikož práce s ním je dobře podporována v PHP. Struktura tohoto archivu bude popsána v části 6.5.1.

6.4.3 Vztahy mezi balíčky

Vztahy mezi balíčky reprezentuje na ER diagramu 6.1 vztah *závisí*, který je rozšířený na entitní skupinu, jelikož reprezentuje relaci *M:N*. Ačkoliv systém APT nabízí mnohem více vztahů mezi balíčky, UMSC podporuje právě jen kolize. Původní návrh počítal ještě se

Kategorie	Účel
sql	Balíčky obsahující hlavně SQL patche. Tyto patche budou připravovat databázové struktury pro různé moduly.
library	Balíčky podpůrných PHP scriptů a konfiguračních souborů. Příkladem zde mohou být soubory obsahující třídy, které jsou využity dalšími scripty.
script	Systémové scripty, které mohou být využity například pro generování konfiguračních souborů, restart provozovaných služeb, nebo jednorázové úpravy v systému
free	Moduly do systému, které nebudou zpoplatněny
payed	Zpoplatněné moduly systému – při instalaci balíčků této kategorie bude prováděna kontrola, zda je na daném systému balíček povolen.

Tabulka 6.1: přehled kategorií balíčků

vztahy *závislost*, *kolize* a *poskytování*. Ty však byly vyřazeny, protože jejich přítomnost řádově zvyšovala složitost systému a přitom jejich plánované využití bylo minimální.

Vztah *závislost* je nejjednodušším vztahem, který vyjadřuje, že součásti daného balíčku potřebují ke svému provozu nějaký prvek dodávaný jiným balíčkem. Systém proto musí umět rozpoznat:

- zda jsou k dispozici balíčky, na kterých je daný balíček závislý. Přitom je potřeba kontrolovat potřebné verze balíčků
- zda balíčky nezávisí na jiných balíčcích – tuto kontrolu je potřeba provádět rekurzivně, dokud není splněna podmínka, že žádný balíček není závislý na nějakých dalších, které nejsou určeny k instalaci

Podstatným problémem závislosti je tzv. *dependency hell*. Tento problém je třeba vyřešit, aby nedocházelo k problémům popsaným v části 6.1, které jsou typické například ve vývojových verzích systému Debian. Jelikož je UMSC podstatně jednodušší (už vzhledem k povaze instalovaných částí systému – jedná se převážně o PHP scripty), omezuje se tento problém jen na kolize mezi požadovanými verzemi balíčku. Nejjednodušším řešením, které také bude v systému použito, využívá lidský faktor. Spočívá v důsledné kontrole závislostí vývojovým týmem, než nové verze balíčků nahraje do repozitáře. Toto řešení maximálně zjednodušuje systém. Pro pohodlí vývojářů balíčků by však také bylo vhodné navrhnout systém, který by umožňoval například koexistenci různých verzí jednoho balíčku. Takovému systému se bude věnovat kapitola **??**.

Vztah *kolize* byl odstraněn, jelikož ke kolizím v systému zpravidla dochází ze třech důvodů:

1. Balíček je závislý na jiném, ale s určitou jeho verzí koliduje.
2. Dva balíčky mají jeden společný soubor, ovšem každý z nich má na soubor jiné požadavky.
3. Dynamicky linkované knihovny nabízejí stejné funkce. Tím by potenciálně mohlo dojít k problémům při spuštění programu ve chvíli, kdy by náhodou programu byly nabídnuty špatné funkce

První problém se dá částečně odstranit závislostí na určitých verzích balíčku. Zůstane pouze případ, kdy je program závislý na určitém rozsahu verzí, ale z tohoto rozsahu je například vyjmuta jedna verze, která měla oproti předchozím i následujícím určitou nevyhovující úpravu. Tento případ však lze považovat za velmi vzácný. Druhů problém lze eliminovat při programování balíčku, stačí, aby autor sledoval, zda k nějaké kolizi nedojde. Druhou možností je určit systém pojmenovávání, případně umísťování souborů patřících do balíčku. Třetí případ se netýká tohoto UMSC, jelikož knihovny nejsou dynamicky linkované.

Vztah *poskytování* měl sloužit jako doplněk doplněk, jehož úkolem je do jisté míry zabránit výše popisovanému problému *dependency hell*. Vztah poskytování říká, že všechny funkce daného balíčku poskytuje ještě balíček jiný. Tím v podstatě vzniká alternativa poskytovaného balíčku, kterou lze využít při například při kolizi s jiným balíčkem. Jelikož však byl odstraněny kolize, není tento balíček dále potřeba.

6.4.4 Operace s balíčky

Podstatnou částí definice balíčkou jsou akce, které se mají vykonávat. Rozlišujeme čtyři operace s balíčkem: *instalace*, *odebrání*, *upgrade*, *downgrade* a *protiakce*. Protiakce budou vysvětleny níže. Instalace a odebrání mají smysl jasně definovaný. Poněkud jinak je tomu u akcí upgrade a downgrade. Každá z těchto akcí provede změnu jen do nejbližší verze balíčku, která je dostupná. Pokud tedy chceme provést například upgrade na nejnovější verzi nějaké knihovny a mezi nainstalovanou a aktuální verzí existuje ještě několik dalších verzí, je potřeba pro každou z nich provést aktualizační mezikrok.

Pro každou operaci s balíčkem lze definovat sérii akcí, které se při ní mají provést. U jednotlivých akcí lze přitom určit pořadí jejich provedení. Existují základní typy akcí: *provedení sql scriptu*, *spuštění systémového scriptu*, *instalace souboru*, *smazání souboru* a *aplikace patche na soubor*. U každé akce je možné uvést cestu k souboru s ní souvisejícímu – u instalace souboru je soubor umístěn v předem známém adresáři, cesta jen určuje, kam soubor nekopírovat. U provedení SQL scriptu je uveden soubor obsahující daný script, stejně tak u spuštění scriptu. Při aplikaci patche na soubor je uvedena cesta k souboru, na který je potřeba patch aplikovat. Soubor s patchem bude mít předem stanovený tvar `patchX.txt`, kde `X` je pořadové číslo vykonávané operace.

Důležitým atributem každé akce je její protiakce. Tento atribut je neocenitelný v případech, kdy selže nějaké operace s balíčkem, například jeho aktualizace. V takovém případě se lze vrátit k původní verzi prostým provedením protiakcí ke všem akcím, které byly dosud provedeny. Zavedení protiakcí systém neučiní o mnoho složitějším a je to neocenitelné pro programátory modulů, kterým se tak ušetří programování každé akce v balíčku tak, aby testovala, zda není zrovna aplikována při zotavení z chyby. Tento postup by totiž byl druhý možný – kdyby například selhala instalace, byla by automaticky zavolána deinstalace a všechny její akce. Protiakce je v případě tohoto systému reprezentována číselným ID akce, která protiakci tvoří. Pro tyto akce je také vytvořena operace protiakce, která shromažďuje akce, které jsou použity jako protiakce, ale do žádné jiné operace nepatří. Ačkoliv by k něčemu takovému nemělo dojít, může se vyskytnout speciální případ, kdy bude třeba po tomto řešení sáhnout. Akce tak bude umístěna do neutrální operace a nebude tak hrozit její nechtěná aplikace při operaci s balíčkem. U protiakcí, které nejsou využity v žádné jiné operaci, nebude třeba protiakci uvádět.

6.4.5 Stavy balíčků

Tato problematika částečně souvisí s předchozí. Systém APT si pamatuje dva stavy balíčků – současný a nastávající. Za klidového stavu je nastávající stav stejný jako současný. Ve chvíli, kdy administrátor systému označí nějaký balíček například k instalaci, je změněn jeho nastávající stav na instalovaný. UMSC podporuje informace jak o současném, tak i nastávajícím stavu. Všechny informace o stavech balíčků jsou obsaženy v tabulce reprezentované entitní množinou *změněný balíček*. Stavy balíčku mohou být následující:

instalován není naplánována žádná akce

instalovat není instalován, ale je na instalaci naplánován

odstranit je instalován, je plánováno jeho odstranění

aktualizovat je instalován, je plánováno jej aktualizovat

downgradovat je instalován, je plánováno jej downgradovat

Ve výčtu chybí stav *neinstalován*. Tento stav je považován za výchozí, proto není uváděn, pokud není pozměněn nebo není na změnu naplánován. U stavů je navíc uvedena informace o současné a plánované verzi. Současná verze balíčku je při stavu *instalovat* nulová. Nová verze je nulová u stavu *odstranit*. K informaci o stavu patří také informace, zda je balíček pouze závislostí, nebo byl instalován uživatelem. Tato informace se může hodit pro přehlednost systému. Pokud by byly smazány všechny balíčky závisející na automaticky instalované součásti systému, mohla by být tato také smazána.

6.4.6 Akční řetězec

Při operaci s balíčky, konkrétně v situaci, kdy je nějaký balíček instalován pro splnění závislostí, se může stát, že tento balíček je instalován, protože závisející balíček bude při požadované operaci potřebovat spustit systémový script poskytovaní tímto balíčkem. Tato situace by mohla být problematická, kdyby byl zavisející balíček obsloužen dříve, než jeho závislosti. Proto je vhodné si uložit řetězec operací, který reprezentuje informaci o pořadí, v jakém musí být balíčky obslouženy, aby k popisovanému problému nedošlo. Aby nedocházelo ke zbytečné duplicitě některých dat, je informace o pořadí atributem entitní množiny *změněný balíček*. Pole s pořadím by mělo být vyplněné u každého balíčku, jehož stav má být změněn. Při pokynu k aplikaci požadovaných operací provádí instalátor operace podle jejich číselné hodnoty od nejnižší, resp. nejvyšší (záleží na implementaci).

Problém popsaný v předchozím odstavci s sebou přináší další omezení. Při upgradu (resp. downgradu) balíčku je potřeba myslet na to, že nebudou dostupné závislosti mezistupňů mezi počátečním a koncovým stavem. To omezuje programátora v tom, že nemůže spoléhat na zavolání systémových scriptů, které nejsou součástí balíčků, na kterých je závislá konečná (tedy požadovaná) verze aktualizovaného balíčku.

Akční řetězec zanáší do systému další riziko, na které by měl správce repozitáře, resp. vývojář balíčků dát pozor. Tím rizikem je kruhová závislost balíčků. Kruhovou závislost lze detekovat při instalaci balíčků, záleží však na implementaci. V případě dobrého návrhu lze s využitím právě pořadového čísla pro obsloužení balíčku naprogramovat algoritmus, který zkusí zpětnou závislost najít. Této závislosti se však dá velice efektivně předcházet již při sestavování balíčků. Mohou nastat dva případy, proč je balíček zpětně závislý na některém, jehož je sám závislostí:

1. zpětná závislost existuje, aby balíček bylo možné instalovat
2. změtná závislost existuje, aby balíček bylo možné provozovat

Prvnímu případu se dá předejít vytvořením virtuálního balíčku, který provede to, co je potřeba, aby bylo možné závislý balík instalovat. V tom případě je ale potřeba, aby změny, které provede, nekolidovaly s těmi, které má provést původní balíček (např. aby nakopírovaný soubor bylo možné přepsat stálým, který patří původnímu balíčku). Druhý případ je možné ignorovat, jelikož používat se oba balíčky budou až po jejich instalaci, kdy už bude vše v pořádku.

6.5 Klientská strana

6.5.1 Struktura balíčku

Po stáhnutí z repozitáře je balíček rozbalen do adresáře `/home/packages/package`. Obsahuje základní adresářovou strukturu, kterou tvoří tři adresáře. První z nich je adresář `files` obsahující potřebné soubory balíčku. Tyto soubory jsou uspořádány do stromové struktury tak, aby jejich umístění vzhledem k adresáři `files` bylo stejné, jako jejich umístění vzhledem ke kořenovému adresáři po instalaci na jejich pozici.

Druhým adresářem je `sql` tento obsahuje soubory s SQL scripty, které budou při nějaké operaci s balíčkem potřeba. Znalost adresáře, kde budou scripty uložené, je možné využít, takže do cesty v databázi akce je třeba uvést jen název souboru. Patch bude aplikován pomocí příkazu `mysql --user=system --password=<heslo> < <cesta_k_souboru>`. První příkaz ve scriptu by tedy měl být `USE <nazev-databaze>`. Heslo pro uživatele system je pro všechny systémy stejné a je pevně uložené v klientu balíčkovacího systému. Cesta k souboru bude uvedena v parametru akce, jež má SQL script vykonat.

Poslední složkou archivu je adresář `control`. Ten obsahuje servisní soubory, jakými mohou být například patche a scripty, které během instalace může být třeba spustit. Některé akce při instalaci mohou využít známé pozice těchto scriptů k tomu, aby je spouštěly v případě, že budou potřebovat vykonat nějakou operaci například s právy superuživatele. K tomu je vhodné určit jeden soubor, který bude má povolené spouštění přes `sudo`. Určeme tedy cestu k takovému scriptu jako `/home/packages/package/control/su-script`. Tomu je tedy potřeba v souboru `/etc/sudoers` přidat možnost spouštění jako superuživatel.

6.5.2 Podpůrné scripty

Kromě administrace samotné a scriptů dodaných v balíčcích je v systému potřeba několik velmi jednoduchých podpůrných scriptů. Tyto scripty mají za úkol dělat prostředníka mezi administrací a systémem. Prostředník je potřeba hlavně z důvodu omezení uživatelských práv. Administrace je provozována přes webové rozhraní, takže automaticky přebírá oprávnění uživatele, pod kterým je spuštěna, tzn. uživatele *www-data*, což je standardní uživatel vytvořený pro provoz www serveru Apache. Oproti tomu v systému budeme potřebovat často provádět akce, které vyžadují oprávnění superuživatele, případně i jiného systémového uživatele. Z toho důvodu by měl být přístupný script, který dané akce umožní vykonat tím, že jej zavoláme přes `sudo`.

Kapitola 7

Implementace systému

7.1 Podpůrné scripty

V systému jsou dostupné čtyři základní podpůrné scripty. Jedná se pouze o základní scripty, jejichž využití se dá předpokládat ve velkém množství balíčků. První z těchto scriptů slouží k instalaci souboru. Jedná se o vylepšenou verzi programu install, který je standardní součástí většiny Linuxových distribucí. Tento umí navíc vytvořit adresářovou strukturu pro soubor. Vytvářené adresáře a kopírované soubory automaticky dědí vlastníka podle nadřezaných adresářů. Ve scriptu probíhá kontrola, zda se scriptu nesnaží někdo na vstup poslat soubor, který není součástí rozbaleného balíčku. Tato kontrola, spolu se schopností vytvářet adresáře, byla důvodem pro vznik tohoto scriptu.

Druhý je script na odstranění instalovaného souboru. Zde už žádná kontrola neprobíhá, ovšem script maže adresáře, jestliže v nich smazal poslední položku.

Třetí script umožňuje modifikovat soubor `/etc/sudoers`. Pro jeho úpravu je totiž nejdříve potřeba změna přístupových práv. Stejně tak je potřeba změna práv na původní po úpravě souboru. Pro tento script je důležité zavést silné omezení, jelikož útokem na něj by potenciální útočník mohl získat neomezenou kontrolu nad systémem. Jako omezení by mělo stačit možnost, zda soubor, pro který je požadováno jeho spuštění přes `sudo`, existuje v daném balíčku. To ověříme jeho existencí daného souboru v adresáři `/home/packages/package`.

7.2 Implementace závislostí

Systém závislostí je z celé implementace systému tou nejzajímavější částí, jelikož problematika závislostí a korektní instalace je poměrně komplexní vzhledem k ostatním částem systému. Po zjednodušení návrhu, kdy byly odstraněny kolize a alternativy balíčků, je možné implementovat závislosti jako N-nární strom. Jako kořeny tohoto stomu slouží balíčky, které nejsou instalovány jako závislosti. Listy stromu tvoří balíčky, které nemají žádné další závislosti. Typicky se bude jednat o knihovny, případně balíčky instalující součásti systému.

Aby byla splněna podmínka stromu, bude mít tento jeden virtuální uzel, který bude sloužit jako kořen a bude spojovat všechny nezávislé balíčky. Tato úprava následně také vede ke zjednodušení práce s tímto stromem. Pro další zefektivnění práce bude vytvořeno pole odkazů na objekty existující ve stromě. To vede k několika vylepšením. Prvním z nich je mnohem rychlejší vyhledání, zda je uzel ve stromu a nalezení odkazu na tento uzel. To by jinak bylo potřeba řešit rekurzivním průchodem, který však může být velmi neefektivní.

Při vhodném využití těchto indexů lze také detekovat, zda při slučování stromů už byl daný uzel (a jeho závislosti) navštíven. To se, vzhledem k tomu, že každý uzel může mít potenciálně několik rodičů, může stát a když nedochází k této detekci, byly by některé větve procházeny dvakrát. Všeobecně lze tedy říci, že hlavní úlohou indexového pole je zabránění ve vícenásobném rekurzivním průchodu některých větví stromu, ke kterému by jinak docházelo u uzlů, které mají více rodičů.

7.3 Instalace balíčku

Tato operace má dvě úskalí, které UMSC řeší. Prvním je Dočasný strom. Ten je využit v případě výskytu chyby. Není sice nutně potřeba, nicméně v případě výskytu chyby v závislostech výrazně zjednodušuje zotavení z této chyby. Strom závislostí balíčků je v takovém případě totiž konzistentní, chyba se vyskatuje na dočasném stromu. Stačí tak dočasný strom smazat a vrátit chybový kód.

Druhý problém vzniká kvůli vlastnostem jazyka PHP. Při přiřazování objektů je totiž přiřazena reference na tento objekt. Objekt přitom zůstává v paměti, pokud na něj jsou nějaké reference. Proto je třeba po spojení dočasného a stálého stromu balíčků přerušit všechny reference mezi objekty v dočasném stromu. Toto rušení probíhá už při spojování stromů, neboť by se mohlo stát, že některá větev stromu by zůstala oddělená od kořene stromu (díky algoritmu, který je použit na spojování stromů) a nemohla by tak být rozpojena patřičnou funkcí.

7.4 Odstranění balíčku

Aby odstanění balíčku mělo stejnou možnost zotavení z chyby, jakou má instalace balíčku, je potřeba operaci samotnou provádět v prostoru odděleném od hlavního stromu balíčků. Princip bude poněkud jiný, než tomu bylo u instalace, jelikož nyní není třeba uchovávat dodatečné informace, potřebujeme pouze vědět, které balíčky odstranit. Je tedy využito pole odkazů na balíčky, které mají být odstraněny. Odstraněn bude balíček, který uživatel označil a všechny jeho závislosti, které byly instalovány automaticky (tzn. mají zapnutý příznak, že byly instalovány automaticky jako závislost balíčku). U všech odstraňovaných (jak u závislostí, tak u balíčku, který k odstranění označil uživatel) je navíc kontrolováno, zda nejsou závislostí jiného balíčku. Pokud ano, není takový balíček, ani jeho závislosti dále vyšetřován.

7.5 Grafické rozhraní

Vzhled grafického rozhraní není v současném provedení podstatný, proto se zaměřmě raději na to, co nabízí a jaké jsou jeho možnosti. Grafické rozhraní je v základním stavu představováno možnostmi nainstalovat modul a odebrat modul. Zobrazené by přitom měly být pouze moduly administrace, nikoliv všechny balíčky – to by pouze mátlo uživatele a nemělo by to žádné významné pozitivum. Po instalaci nějakého modulu do administrace přibude v menu položka, která bude tento modul zastupovat. Zobrazené možnosti nastavení modulů závisí na těchto konkrétních modulech, návrh stránky s nastavením bude vždy záležitostí tvůrce modulu. Položky do menu jsou přidávány includem souborů v podadresáři administrace. V každém takovém souboru by měl být HTML kód jedné položky menu. Vzor si lze prohlédnout na přiloženém CD.

Kapitola 8

Návrh a příklady modulů

8.1 FTP server

8.1.1 Účel

Balíček patří do skupiny balíčků, která zastupuje pouze instalaci služby. Tento balíček je tedy pouze závislostí modulů,které modifikují nastavení daného FTP serveru.

8.1.2 Součásti balíčku

Hlavní součástí balíčku je script, který provede příkaz

```
aptitude install proftpd proftpd-mysql
```

Tím máme v podstatě nainstalovaný FTP server a úloha tohoto balíčku končí. Pro spuštění scriptu samozřejmě musí být v repozitáři uvedená akce. Všechny náležitosti tohoto balíčku jsou umístěny na přiloženém CD.

8.2 Nastavení systémových kvót

8.2.1 Účel

Tento balíček by byl závislostí modulu administrace, který by nastavoval systémové kvóty. Jeho účelem by byla příprava systému pro funkčnost systémových kvót. Tzn. hlavním cílem by bylo při instalaci systémové kvóty zavést a při odinstalaci je vypnout. Prerekvizitou tohoto balíčku by byl provoz virtuálních webhostingů. Tato prerekvizita je spíše logická, než technická. Zavádět systémové kvóty je až na vyjímky zbytečné, když systém využívá jediná osoba.

8.2.2 Součásti balíčku

Balíček by měl několik částí, které by bylo potřeba vykonat. Nejdříve by bylo třeba vykonat SQL script, který by do tabulky s provozovanými virtuálními hostingy přidal sloupec pro nastavení jejich kvóty. Dále by bylo potřeba nainstalovat script, který by data z databáze převáděl do systému. Za předpokladu, že v systému by jako prerekvizita již platilo, že pro jednu hostovanou doménu existuje jeden systémový uživatel, bylo by dále třeba jen do souboru `/etc/sudoers` přidat řádek, aby Apache mohl spouštět nainstalovaný script

pro transformaci dat z databáze do systému. Dále by pravděpodobně bylo vhodné nastavit všem výchozí kvóty, kvóty aktivovat v souboru `/etc/fstab` a restartovat systém, aby změny začaly platit.

8.3 Modul administrace - kvóty

8.3.1 Popis

Modul by vycházel z následujích prerekvizit: nainstalovaný FTP server, načítání uživatelů z databáze, provoz virtuálních webhostingů, zprovozněné systémové kvóty. Po instalaci modulu by do administrace přibyla stránka, kde by bylo možné nastavit každé hostované doméně diskovou kvótu. Po potvrzení kvót by tyto byly uloženy do databáze a z nich by byly vygenerovány nové hodnoty kvót pro systémové uživatele. Přitom by modul využíval právě možností, které mu dávají prerekvizitně instalované balíčky (např. volání scriptu pro zápis dat systémovým uživatelům).

Kapitola 9

Možnosti vylepšení

9.1 Koexistence verzí balíčků

Tento problém by bylo možné vyřešit například pomocí nástroje, který by se dal nazvat linkerem. Jeho účel by spočíval v procházení souborů se scripty a nacházení, kde jednotlivé scripty využívají nějaká externí volání (`include` v PHP, `os.popen2` v Pythonu). Cesty k těmto externím voláním by pak nahrazoval skutečnou cestou, kde soubor je umístěn.

Problém by však bylo sestavení parseru pro různé programovací jazyky a různá volání. Celý systém by tak bylo potřeba zjednodušit například tak, že tato volání jiných částí systému by byla uložená v konstantách (pouze však cesty v souborovém systému). Tyto konstanty by byly umístěné na označeném místě v souboru: Zároveň by také byly označeny, která cesta patří k jakému balíčku. Označení by bylo provedené například komentářem. Podobný systém je použit pro balíčkovací systém APT, když má provádět nějaké změny v konfiguračních souborech.

9.2 Pokročilé řešení kruhových závislostí

Současné řešení kruhových závislostí využívá lidský faktor, kdy správce repozitáře musí hlídat, aby smyčka nevznikla. Má k dipozici metody, jak případnou smyčku rozpojit, tyto metody byly popsány v sekci 6.4.6. Pokročilé řešení by mohlo využít dodatečných informací o tom, proč konkrétně existuje zpětná závislost. Jelikož se každá operace skládá z jednotlivých akcí, bylo by možné akční řetězec nesestavovat jako pořadí operací, ale pořadí jednotlivých akcí.

Tato úprava by mohla mít dvě formy. První forma požaduje rozsáhlé změny v systému – spočívala by totiž v tom, že by byly nejprve staženy a rozbaleny všechny balíčky a místo N-nárního stromu, který v současnosti slouží jako základ pro instalační řetězec by musela být využita struktura, kde by bylo definováno pořadí akcí z jednotlivých balíčků. To by samozřejmě také požadovalo rozsáhlé změny v databázi. Po těchto změnách by se podstatně zvětšilo množtví dat ukládaných do databáze – bylo by potřeba ukládat pořadí akce, balíček a operaci, ke které akce patří a číslo akce pro danou operaci. Tato data by musela být přitom generována systémem podle toho, jaké balíčky by zrovna měly být pozměněny. Toto řešení je však pro účely současného systému zbytečně komplexní, proto nebylo využito v první verzi, která je součástí této práce.

9.3 Základní modulové vybavení

Aby byl systém dobře prodejný, je potřebné, aby se nejednalo pouze o jádro systému, ale aby k němu existovalo určité množství základních modulů. Jako téměř nutný základ můžeme považovat *FTP server*, *www stránky hlavní domény*, *emaily hlavní domény*, *virtuální hosting emailů a webů*.

Jako doplňkové služby by mohlo být vhodné implementovat *SSL komunikaci* pro web i emaily. Jako součástí všech zmiňovaných balíčků by byla i část, která by implementovala oddělenou administraci, například pro zmiňované virtuální hostingy. Pro případy, kdy neprivilegovaný systém využívá pouze zákazník sám a služby dále nepronajímá bude stačit vždy jen rozšíření administrace systému (tj. ta, kde lze instalovat nové moduly).

Jako „systémové moduly“ by bylo možné nabízet např. *IP adresy navíc*, *vytvoření systémového uživatele*, za kterého bude možné se přihlásit přes SSH. Dále, jelikož je celý systém provozován na Xenu, je možné využít možnosti právě tohoto systému. Tak by se dal naprogramovat modul *monitorování* a *automatické migrace* v případě přetížení hostitelského počítače.

Kapitola 10

Závěr

Jelikož návrh samotného systému byl časově řešen ještě v době, kdy představený obchodní model nebyl ještě zcela promyšlený, byl systém administrace navržen tak, aby byl pokud možno co nejuniverzálnější. Pozdější upřesnění podmínek jeho provozu dovolilo dosti silně zjednodušit bezpečnostní opatření, což se ve výsledném efektu ukázalo jako obrovská časová úspora, jelikož potřeba návrhu zabezpečení by mohla tuto práci zavést do poněkud odlišné problematiky, než která byla původním cílem.

Systém balíčků byl navržen daným způsobem z důvodu zabránění redundance dat. Jednoduchý modulární systém by jako modul předložil samostatnou sadu scriptů, která nezávisí na žádném jiném modulu. To je jistě velmi jednoduché a svým způsobem i návrhově a implementačně efektivní řešení. Má však jednu nevýhodu, kterou je nemožnost využití některých částí jinými. Může tak docházet k duplicitě kódu. Proto byly zavedeny závislosti mezi jednotlivými balíčky – je možné programovat knihovny, které samy o sobě nebudou moduly aplikace, ale budou sloužit jako spojka mezi těmito moduly a službou, kterou ovládají.

Systém administrace má velice silný potenciál na budoucí využití, jelikož je naprosto univerzální – nebude jej tak třeba provozovat na virtualizovaném systému. Systému by jistě také pomohlo vydání některé budoucí verze pod licencí GPL. K projektu by se mohla dostat širší skupina vývojářů a mohly by tak přibývat nové moduly.

Pokud jde o virtualizaci a její nasazení pro ISP, určitě jsou možnosti, jak lze virtuální systémy využít. Nevýhodou však je vysoká náročnost na hardware. Aby byly využity všechny výhody virtualizovaných systémů, je potřeba vybavení za cenu, při které spousta ISP začíná uvažovat, jestli to má vůbec cenu. Druhým problémem je, že Xen je poměrně mladý projekt, který ještě není stabilní tolik, jako například konkurenční VMWare. Na projektu však pokračuje neustálý vývoj a každá verze nese známky výrazného pokroku. Obrovským krokem kupředu by bylo zařazení subsystémů pro HyperCally do vanilla kernelu.

Přínosem této práce je položený základ aplikaci s modularitou, která nabízí dobrý kompromis mezi komplexností balíčkovacího nástroje *dpkg* a jednoduchostí obvyklých provedení modulů webových aplikací. Za další přínos lze považovat obchodní model navržený pro provoz virtualizovaných systémů, který by mohl pokrýt novou oblast trhu. Tento model však ještě není prověřen praxí, proto se o velikosti jeho přínosu dá zatím jen spekulovat.

Literatura

[1] NetBSD/xen Howto.
URL <http://www.netbsd.org/ports/xen/howto.html>

[2] OpenSolaris Community: Xen.
URL <http://www.opensolaris.org/os/community/xen/>

[3] XenNetworking.
URL <http://wiki.xensource.com/xenwiki/XenNetworking>

[4] BOURKE, T.: Comparing MySQL performance.
URL <http://www.linux.com/feature/41348>

[5] DAGUE, S.; STEKLOFF, D.; SAILER, R.; aj.: *xm - Xen management user interface*. 1 2008.

[6] DALOUCHE, S.: Xen Disk I/O performance vs native performance.
URL <http://www.nabble.com/Xen-Disk-I-O-performance-vs-native-performance-td15076789i20.html>

[7] KABIR, M. J.: *Apache Server 2 – Kompletní příručka administrátora*. Brno: Computer Press, 2004, ISBN 80-251-0319-6.

[8] KUPARINEN, M.: Xen Disk I/O Benchmarking: NetBSD dom0 vs Linux dom0.
URL <http://users.piuha.net/martti/comp/xendom0/xendom0.html>

[9] ROSEN, R.: Linux journal - Introduction to the Xen Virtual Machine.
URL <http://www.linuxjournal.com/article/8540>

[10] STEPHENS, P.: XenArchitecture.
URL <http://wiki.xensource.com/xenwiki/XenArchitecture?action=AttachFile\&do=get\&target=Xen+Architecture_Q1+2008.pdf>

[11] Wikipedia: Ring (computer security) — Wikipedia, The Free Encyclopedia. 2008, [Online; accessed 16-April-2008].
URL <http://en.wikipedia.org/wiki/Ring_(computer_security)>

[12] WILLIAMSON, M.: vanilla kernel and xen (in general), vanilla 2.6.24 and xen.
URL <http://www.nabble.com/Re%3A-vanilla-kernel-and-xen-%28in-general%29%2C-%09vanilla-2.6.24-and-xen-p15250397.html>

Seznam příloh