VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Virtualizační platforma pro dynamickou aktivaci specifikovaných virtuálních instancí s OS Linux

Virtualization Platform for the Dynamic Activation of Specified Virtual Instances with Linux OS

2010 Pavel Juška

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2010 .....

Děkuji tímto vedoucímu diplomové práce Ing. Martinu Milatovi za odbornou pomoc a konzultaci při realizaci této diplomové práce.

Abstrakt

Tato práce se zabývá výběrem vhodného virtualizačního systému, návrhem a realizací nového řešení ovládání virtuálních strojů. Požadované vlastnosti plynou z potřeb Virtuální laboratoře počítačových sítí, která bude tento systém využívat jako náhradu stávajícího, dále nevyhovujícího systému. Obsahem úvodní části práce je shrnutí teorie virtualizace v základním rozsahu nutném pro návrh a přehled současných řešení. Z analýzy požadova-ných vlastností a dostupných prostředků vyplynulo jako nejvhodnější řešení založené na virtualizačním systému KVM. Zadaný úkol se podařilo vyřešit vytvořením knihovny bash skriptů nad zvolenou virtualizační platformou. Řešení úspěšně obstálo při důklad-ném testování na serveru.

Klíčová slova: virtualizace, virtuální stroj, KVM, Virtuální laboratoř počítačových sítí

Abstract

This thesis is concerned with selection of a proper virtualization system, design and realization of a new solution of control of virtual machines. Required features follow from the needs of Virtual Networking Laboratory, which will use this system as a replacement of the recent and no longer sufficient system. The content of the first part of the thesis is the summary of theory of virtualization in basic extent necessary for design and survey of recent solutions. The analysis of required features and available resources showed the most proper solution is the one based on KVM virtualization system. The goal was achieved by creating a bash script library above the chosen virtualization platform. The solution successfully came through a thorough testing on a server.

Keywords: virtualization, virtual machine, KVM, Virtual Networking Laboratory

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Jedním z nejdůležitějších trendů v oblasti informačních technologií je v současné době zcela nepochybně virtualizace. K nasazení tohoto dnes tolik populárního řešení dochází velmi často, díky jeho širokému uplatnění v oboru informačních technologií.

Dávno se již nejedná o počáteční experimentování s novou technologií, často doprovázenou obavou, zdali nejde pouze o medializaci zajímavélo, leč v praxi nepoužitelného řešení. Čas a pevné místo virtualizace mezi IT řešeními v korporátní sféře již jasně ukázaly, že jde o vyzrálou technologii. Důvodů jejího nasazení je velmi mnoho: od těch finančních jako je úspora peněz za hardware či elektrickou energii až po ryze technické jako je možnost okamžitého vytvoření libovolných virtuálních strojů, jejich snadná správa, plnohodnotné využití hardwarového výkonu atd.

K jednomu z míst, kde našla virtualizace své uplatnění, patří i Virtuální laboratoř počítačových sítí (Virtlab) při Katedře informatiky na Fakultě elektrotechniky a informatiky Vysoké školy báňské – Technické univerzity Ostrava.

Cílem této práce je vytvořit kompletní řešení pro dynamickou aktivaci specifikovaných virtuálních instancí s OS Linux. Je třeba vybrat vhodnou virtualizační platformu pro konkrétní potřeby Virtuální laboratoře počítačových sítí, následně vytvořit kompletní řešení pro dynamickou aktivaci a ovládání specifikovaných virtuálních strojů s finálním nasazením do ostrého provozu na serveru.

1.1 Teorie virtualizace

Tato kapitola bude věnována stručnému shrnutí teorie virtualizace, která je nezbytně nutná pro pochopení dále využívané technologie. V jednotlivých kapitolách budou rozvedeny některé základní pojmy a principy tohoto odvětví.

Vzhledem k faktu, že v současné době se již jedná o tak rozsáhlou oblast IT a výklad teorie virtualizace není náplní této práce, zde bude dále uveden pouze stručný výtah, nikoli kompletní a obsáhlý popis.

1.1.1 Virtualizace

Virtualizace[1] se obvykle definuje jako abstrakce výpočetních zdrojů. V rámci informačních technologií se pod pojmem virtualizace rozumí techniky a postupy, jež pracují s dostupnými zdroji odlišným způsobem, než jak odpovídá jejich fyzické struktuře. Pomocí tohoto virtualizovaného prostředí můžeme dosáhnout libovolného, námi požadovaného uspořádnání virtuálních prvků, nezávisle na dostupném hardwaru. To s sebou nese řadu výhod v podobě možnosti přizpůsobení měnícím se potřebám uživatelů, úspory financí z důvodu agregace zdrojů, úspory místa pro fyzický hardware, bezpečnosti atd.

V současné době můžeme virtualizovat takřka vše od kompletní infrastruktury hardwaru až po jednotlivé části systému. Takto lze simulovat celé počítače (tzv. virtuální stroje), dílčí hardwarové komponenty (například virtuální paměť, virtuální procesor či virtuální disk atd.), případně pouze softwarové prostředí v podobě virtualizace operečního systému apod.

Strukturovat pojem virtualizace v rámci IT můžeme následovně:

- Virtuální stroj (Virtual machine, VM)
  - Virtualizace platformy (Platform virtualization)
    - Plná virtualizace (Full virtualization)
    - Hardwarově asistovaná virtualizace (Hardware-assisted virtualization)
    - Částečná virtualizace (Partial virtualization)
    - Paravirtualizace (Paravirtualization)
    - Virtualizace na úrovni operačního systému (Operating system-level virtualization)
  - Aplikační virtualizace (Application virtualization)
    - Přenosné aplikace (Portable application)
    - Meziplatformní virtualizace (Cross-platform virtualization)
    - Virtualizační aparát (Virtual appliance)
    - Emulace nebo Simulace (Emulation or simulation)
- Virtuální paměť (Virtual memory)
- Virtualizace úložišť (Storage virtualization)
- Paměťová virtualizace (Memory virtualization)
- Síťová virtualizace (Network virtualization)
  - Virtuální privátní sítě (Virtual private network, VPN)
- Desktopová virtualizace (Desktop virtualization)
- Datová virtualizace (Data virtualization)
- Databázová virtualizace (Database virtualization)
- Virtuální účtování (Virtual Chargeback)

Předcházející pojmy jako takové nejsou nikde univerzálně uznané či standardizované apod. Principy se strukturou na níž rozdělení stojí však lze nalézt ve většině literatury.

Vzhledem k zaměření této práce zde dále bude podrobněji rozepsána pouze část týkající se virtualizace platformy.

1.1.2 Virtualizace platformy

Historický význam termínu virtualizace[2] původně pochází již z 60. let 20. století a označoval vytváření virtuálních strojů při použití kombinace softwaru a hardwaru. Toto se dnes nazývá virtualizací platformy. Pokusný stránkovací mechanismus systému IBM M44/44X dal zase vniknout pojmu virtuální stroj. V současné době mají oba tyto termíny již své další významy a jejich chápání se může proto odlišovat.

Virtualizace platformy je realizována na fyzické hardwarové platformě formou softwarového hostitele, jež simuluje virtuální počítačové prostředí neboli vytváří virtuální stroj pro hostovaný software. Tím vznikne spolupracující dvojice hostitel (reálný stroj) – host (virtuální stroj). Software hosta pak běží v takto virtuálně simulovaném prostředí stejně, jako by byl nainstalován na skutečné hardwarové platformě. Typicky je takto simulováno několik virtuálních strojů na jednom stroji fyzickém.

Parametry těchto simulovaných virtuálních strojů je možné konfigurovat dle uživatelských potřeb a možností reálného hostitelského hardwaru.

1.1.2.1 Plná virtualizace

Plná virtualizace[3] představuje techniku, jež provádí kompletní simulaci hardwaru pro virtuální stroj. To znamená, že z hlediska hostovaného softwaru se jedná o plnohodnotný počítač. Na takovémto virtuálním stroji tedy můžeme provozovat libovolný software, zejména libovolný operační systém, jenž by běžel i na reálném hardwaru se stejným druhem procesoru (stejnou instrukční sadou jako hostitelský počítač).

Toto je hlavní rozdíl oproti jiným formám virtualizace, kde je umožněn běh pouze některých či speciálně upravených programů. Z hlediska uživatele plné virtualizace je simulovaný stroj k nerozeznání od fyzického.

Tato forma virtualizace se postupem času ukázala jako velice úspěšná díky své komplexnosti a nabízeným možnostem.

1.1.2.2 Hardwarově asistovaná virtualizace

Hardwarově asistovaná virtualizace[3] představuje evoluční vylepšení předešlé plné virtualizace za pomocí speciálních instrukcí procesoru. Do procesorů podporujících tuto formu virtualizace byly přidány speciální sady instrukcí umožňující dosažení lepších výsledků zejména v oblasti výkonu při využití těchto procesorů pro běh virtualizace. Zavedením těchto instrukcí dochází také ke snížení množství změn nutných pro běh virtualizace v hostitelském operačním systému.

V současné době jde o velmi rozšířený model, který využívají všechny hlavní virtualizační systémy.

1.1.2.3 Částečná virtualizace

Částečná virtualizace, jak již název napovídá, nesimuluje kompletní virtuální stroj, jak jej chápeme například z principů plné virtualizace, ale pouze některé části fyzického hardwaru hostitele. Toto prostředí již neumožňuje běh libovolného softwaru, i když v mnohých případech je postačující pouze jeho úprava. Obvykle zde také nelze spustit celý operační systém. Stěžejní částí této formy virtualizace je nezávislý adresní prostor pro běžící programy. Dnes je tato technika součástí většiny moderních operačních systému a neřadí se již mezi klasické formy virtualizace v dnešním smyslu chápání tohoto termínu. Tato virtualizace byla ovšem důležitým historickým mezníkem na cestě k plné virtualizaci, pro jejíž vývoj byly využity zkušenosti s provozem částečné virtualizace.

1.1.2.4 Paravirtualizace

Paravirtualizace[3] představuje zvláštní formu virtualizace, kdy nesimulujeme hardware virtuálního stroje tak, jak jej chápeme například z principů plné virtualizace. Hostitel zde nabízí pouze softwarové rozhraní formou zvláštního API pro speciálně upravený operační systém hosta. Takto uzpůsobený opereční systém tak nepracuje s hardwarem klasickým způsobem, ale skrze volání API dané virtualizace hostitele nazývané hypervizor. Výhodou tak je přímé vykonávání instrukcí virtuálního stroje fyzickým procesorem.

Na druhou stranu mezi jeho hlavní nevýhody patří nutnost speciální úpravy hostovaného operačního systému, což nelze provést u všech systémů. Tento nedostatek se výrobci procesorů snaží odstranit zavedením speciálních instrukcí procestoru, které umožňují běh operačních systémů bez nutnosti jejich modifikace.

Tento způsob je dnes spolu s hardwarově asistovanou virtualizací nejpoužívanější.

1.1.2.5 Virtualizace na úrovni operačního systému

Klíčovou myšlenkou virtualizace na úrovni operačního systému je simulace běhu vícero instancí konkrétního operačního systému uvnitř jedné jeho skutečné instance.

Host tak sdílí stejný operační systém jako hostitel, to znamená použití stejného jádra v hostiteli i hostu. Z hlediska běžících programů hosta je však operační systém chápán jako samostatný. Na takto použité jádro jsou ovšem kladeny specifické nároky plynoucí z využívání hardwarových zdrojů více navzájem nezávislými instancemi.

Tato forma virtualizace se nejčastěji používá u virtuálních hostingů, kde dochází k přidělování omezených hardwarových zdrojů velkému počtu vzájemně nedůvěryhodných uživatelů.

1.2 Současné existující systémy

V současné době již na trhu existuje několik desítek hotových řešení zabývajících se tímto odvětvím. Liší se především rozsahem (jak velké množství funkcí pro práci s virtuálními stroji daný systém nabízí) a kvalitou řešení, univerzálností použití (zda se jedná o systém využitelný například pouze pro konkrétní účely či je zcela univerzální), použitou platformou, typem licence (tedy i cenou) apod.

Nelze tedy srovnávat rozsáhlé systémy vyvíjené pro potřeby velkých korporací s těmi malými často tvořenými jedním vývojářem v rámci open source projektu. V následujícím výčtu jsem se zaměřil na systémy umožňující plnohodnotnou a univerzální virtualizaci se standardní sadou funkcí pro práci s virtuálními stroji. Jelikož i systémů s těmito parametry existuje mnoho, snažil jsem se vybrat ty nejrozšířenější z nich. To by mělo být dostatečnou zárukou pro kvalitní dokumentaci, dlouhodobou podporu, budoucí vývoj, přiměřené množství chyb atd.

Podrobný popis jednotlivých systémů není náplní této práce, a proto následující shrnutí obsahuje pouze stručnou charakteristiku těchto řešení a jejich pozici na trhu. V případě zájmu o podrobnější popis jednotlivých systémů doporučuji navštívit domovské stránky, které budou dále uvedeny.

1.2.1 KVM

Jedním ze současně nejrozšířenějších systémů využívaných pro virtualizaci je zcela nepochybně KVM[5] neboli Kernel-based Virtual Machine. Pro svůj chod využívá hardwarové podpory speciálních instrukcí procesorů v podobě technologií Intel VT nebo AMD-V. Základní plná virtualizace systému Qemu[6] je dále rozšířena o nadstavbu KVM a stává se tak hardwarově asistovanou virtualizací. Celý systém je vyvíjen v rámci open source na platformě Linux a od verze jádra 2.6.20 se stal v únoru roku 2007 jeho součástí. Jednotlivé komponenty celého systému jsou vydávány pod různými variantami GNU licencí.

Domovskou stránku tohoto systému s komplexnějším popisem nalezneme na adrese: http://www.linux-kvm.org/

1.2.2 Windows Virtual PC

Windows Virtual PC, dříve označovaný jako Microsoft Virtual PC či Connectix Virtual PC je virtualizační nástroj pro platformu Microsoft Windows. Oficiálně podporuje pouze běh hosta s tímto operačním systémem. Stejně jako většina moderních virtualizačních systémů taktéž využívá virtualizační podpory hardwaru, čímž se řadí mezi hardwarově asistované virtualizace.

Poslední verze Windows Virtual PC na rozdíl od předchůdce Microsoft Virtual PC běží již pouze pod nejnovějším operačním systémem hostitele, tedy Windows 7.

K dispozici je zde také speciální Windows XP Mode, který slouží pro běh aplikací vyžadujících starší operační systém Windows XP. V tomto módu běží aplikace v Terminal Services session ve virtualizovaném hostu a z hostitele jsou přístupné skrze Remote Desktop Protocol. S hostitelem tak sdílí například nabídku Start, plochu atd.

Windows Virtual PC je komerční nástroj šířený pod proprietární licencí.

Domovskou stránku tohoto systému s komplexnějším popisem nalezneme na adrese: http://www.microsoft.com/windows/virtual-pc/

1.2.3 VirtualBox

Mezi rozšířenější virtualizační nástroje můžeme řadit také VirtualBox od společnosti Oracle, dříve společnosti Innotek a následně Sun. Na rozdíl od předchozích dvou se jedná o multiplatformní systém a to jak na straně hosta, tak i hostitele. Část kódu systému je původně převzata z Qemu. Je nabízen ve dvou variantách, které se liší druhem licence a v návaznosti na to i svojí funkčností: základní VirtualBox Open Source Edition s licencí GNU nabízený v podobě zdrojových kódů a VirtualBox Personal Use and Evaluation License s rozšířenou funkcionalitou a proprietární licencí nabízený v podobě předkompilovaných binárních kódů. I tento systém využívá hardwarové podpory a řadí se tudíž taktéž mezi hardwarově asistované virtualizace.

Domovskou stránku tohoto systému s komplexnějším popisem nalezneme na adrese: http://www.virtualbox.org/

1.2.4 VMware

Nejsofistikovanějším a nejvyužívanějším komerčním nástrojem či spíše nástroji pro virtualizaci je VMware od firmy VMware, Inc. Tato firma nabízí celou škálu produktů pro virtualizaci od základních systémů jako je VMware Player až po rozsáhlé řešení určené pro firemní infrastrukturu jako VMware ESX Server. Jejich produkty jsou nabízeny pod různými proprietárními licencemi. Některé nástroje jsou poskytovány zdarma, jiné pouze v rámci placené licence. Vzhledem k tomu, že se jedná o špičku v oblasti virtualizace, najdeme v jejich nabídce vše co se virtualizace týká. Nabídka této firmy tak neuspokojí pouze uživatele, jež mají zájem o open source či bezplatné řešení.

Domovskou stránku tohoto systému s komplexnějším popisem nalezneme na adrese: http://www.vmware.com/

1.2.5 Xen

Hlavním zástupcem systémů využívajících princip paravirtualizace je Xen. Umožňuje sice i klasickou virtualizaci, ale pouze ve spojení s procesorem jenž ji podporuje v podobě speciálních instrukcí. Výhody a nevýhody tohoto systému tak plynou hlavně z principů paravirtualizace. Spolu s KVM se jedná o nejrozšířenější řešení na platformě Linux. Dnes se však zdá, že KVM získává nad Xen převahu. Celý systém je vyvíjen v rámci open source a jednotlivé komponenty jsou pod GNU licencí.

Domovskou stránku tohoto systému s komplexnějším popisem nalezneme na adrese: http://www.xen.org/

1.3 Proč nové řešení

Stávající řešení založené na virtualizačním systému Xen se postupem času začalo jevit jako nevyhovující. Vzhledem ke stále novým požadavkům Virtuální laboratoře počítačových sítí, jež dosavadní systém není schopen operativně reflektovat nebo jen velmi obtížně, bylo rozhodnuto o vývoji nového řešení.

Problematické vlastnosti současného řešení:

- Obtížný běh více druhů virtualizovaných strojů paralelně (všechny spouštěné instance musejí být jednoho typu).
- Silně omezené možnosti dynamického definování vlastností virtuálních strojů (vlastnosti instancí jsou předem dány konfigurací, během jejich startu se již nedají měnit).
- Omezené možnosti spouštění virtuálních strojů na požádání (dynamičnost spouštění instance na požádání je limitována existující konfigurací, virtuální stroje se nedají spouštět bez ní).
- Problematická aktualizace virtualizačního systému (aktualizace balíčků modifikovaných jader Xen nejsou tak časté jako v případě klasického jádra).
- Omezenost prostředí na platformu Linux a to především na straně hosta (je tak obtížná například instalace operačního systému platformy Microsoft Windows).

- Obtížná implementace GUI přístupu k virtuálním strojům (v současnosti se využívá pouze CLI přístup, do budoucna by byla vhodná i podpora GUI).
- Chybějící nástroje pro ukládání snapshotů virtuálních strojů.

V dalších kapitolách bude proto vybrán nový virtualizační systém. Do výběru bude ovšem taktéž zahrnuta i nejnovější verze momentálně používaného Xenu. Ta může obsahovat nové funkce oproti současně používané verzi a mohla by tak vyhovovat výše zmíněným požadavkům.

2 Analýza

Analýza je základním prvkem při návrhu celého nového řešení. Její chyby a nedostatky se zpravidla projeví v dalších fázích vývoje a můžou přinést výrazné komplikace. Vhodný návrh celého řešení během analýzy je tedy velmi důležitý pro budoucí průběh prací při jeho tvorbě.

Výsledkem analýzy by měl být návrh nového řešení a výběr vhodného virtualizačního systému, jenž bude použit pro jeho implemetaci. Vzhledem k povaze řešeného problému by nové řešení mělo být navrženo právě s ohledem na možnosti dostupných virtualizačních nástrojů. Během analýzy však nelze s určitostí zaručit, že tento nový návrh bude do detailu řešeitelný v rámci vybraného virtualizačního systému, a tak může dojít k drobným úpravám při implementaci vzhledem k jeho možnostem.

2.1 Požadavky na nové řešení

Vzhledem k potřebám Virtuální laboratoře počítačových sítí a co možná nejmenšímu nutnému zásahu do stávajícího řešení byly stanoveny následující požadavky.

2.1.1 Požadavky na obecné vlastnosti nového řešení

- Podpora snapshotů pro snadné spouštění více virtuálních strojů se stejnou základní konfigurací a stejným obrazem disku.
- Možnost snadného spouštění více variant virtuálních strojů pomocí více předem připravených obrazů disku.
- Možnost uchovávat konfigurace a opakovaně spouštět uživatelem konfigurované virtuální stroje.
- Přístup ke konzoli virtuálních strojů přes RAW TCP stream (telent na vysokém portu, telnet bez negociace).
- Možnost konfigurace limitů přidělené RAM paměti. Ideálně dynamicky při spouštění virtuálního stroje.
- Podpora více síťových rozhraní ve virtuálních strojích, na straně hostitele mapovaných na různé VLAN.
- Možnost dynamické specifikace počtu síťových rozhraní v okamžiku spouštění virtuálního stroje. Stejný obraz disku ze kterého operační systém virtuálního stroje bootuje s různým počtem sítových rozhraní, třeba i v každé jednotlivé instanci zvlášť.
- Administrativní funkce (spouštění, ukončování, restartování, výpis informací, ukládání, načítání a mazání).
- Snadné přidávání, odebírání a aktualizace obrazů disku, ze kterých virtuální stoje vychází.

- Bezpečnostní aspekty (oddělení instancí navzájem, oddělení síťového provozu apod.).
- Možnost ovládání skrze příkazovou řádku.

2.1.2 Požadavky na software serveru

- Bezplatný operační systém s dostatečnou podporou virtualizačních technologií. Nejlépe operační systém platformy Linux v podání distribuce Debian[7] GNU/Linux.
- Bezplatný virtualizační systém splňující obecné požadavky na nové řešení. Nejlépe s některou ze svobodných licencí.

2.1.3 Požadavky na software virtuálního stroje

- Operační systém platformy Linux. Bezplatná distribuce, nejlépe Debian GNU/Linux.
- Standardní instalace operačního systému, která zajistí možnost užívání bez dodatečného zaškolení.
- Softwarová výbava:
  - *apache2* – Apache Hypertext Transfer Protocol (HTTP) server
  - *bind9* – Berkeley Internet Name Domain (BIND) server
  - *bind9utils* – nástroje pro BIND
  - *dhcp3-client* – Dynamic Host Configuration Protocol (DHCP) client
  - *dhcp3-server* – Dynamic Host Configuration Protocol (DHCP) server
  - *dnsutils* – Berkeley Internet Name Domain (BIND) client
  - *ftpd* – File Transfer Protocol (FTP) server
  - *gzip* – GNU nástroje pro kompresi
  - *iproute* – nástroje pro ovládání sítě a přenosu po síti
  - *iptables* – administrační nástroje pro filtrování paketů a NAT
  - *iptraf* – nástroje pro monitoring IP LAN
  - *iputils-ping* – nástroje pro testování dosažitelnosti počítače na síti
  - *less* – stránkovací program
  - *lynx* – textový WWW prohlížeč
  - *mc* – dvoupanelový správce souborů
  - *net-tools* – sada síťových nástrojů NET-3
  - *nmap* – mapovač sítě
  - *openssh-client* – Secure Shell (SSH) client
  - *openssh-server* – Secure Shell (SSH) server
  - *openssl* – Secure Socket Layer (SSL) a související kryptografické nástroje

- *openvpn* – Virtual Private Network (VPN) démon
- *tcpdump* – nástroje pro sledování sítě a získání dat
- *tftpd* – Trivial File Transfer Protocol (TFTP) server
- *traceroute* – nástroje pro sledování trasy paketů po síti TCP/IP
- *vim* – Vi IMproved - rozšířený editor vi
- *vpnc* – Cisco kompatibilní VPN client

2.2 Návrh nového řešení

Při návrhu nového řešení je třeba vycházet převážně z toho stávajícího. Změny, které by se dotkly okolních modulů Virtuální laboratoře počítačových sítí nejsou příliš žádoucí, vzhledem k jejich vzájemnému provázání. V ideálním případě by tak mělo dojít pouze k nahrazení stávající řešené části, nezávisle na zbytku systému.

Tento požadavek určuje zásadním způsobem koncept celého navrhovaného řešení. To musí do jisté míry respektovat to stávající, hlavně co se ovládací části týká.

2.2.1 Serverová část

Celý systém poběží na samostatném serveru, který krom správy instancí virtuálních strojů nebude zatěžován žádnými dalšími úkoly. Pro využití maximálního výkonu současných virtualizačních systémů je nutné, aby jeho procesor obsahoval speciální sadu instrukcí pro podporu virtualizace. Žádné jiné speciální požadavky na hardware serveru kladeny nejsou. Výkon serveru by měl být ovšem přímo úměrný počtu zároveň spouštěných virtuálních strojů. Současné požadavky počítají s maximálně 30 současně běžícími virtuálními stroji.

Z požadavků na software serveru vyplývá nutnost volby jednoho z bezplatných operačních systémů s dostatečnou podporou virtualizace, nejlépe platformy Linux v podání distribuce Debian GNU/Linux. Vzhledem ke splnění požadavků zadání a spolu s faktem, že jde o jeden z nejpoužívanějších serverových systémů, což je dostatečnou zárukou kvality výběru, je vhodné tuto distribuci použít. Dostatečná kompatibilita operačních systémů této platformy a podpora virtualizace většiny rozšířenějších distribucí zaručuje případně snadný přechod k jiné distribuci.

2.2.2 Aplikační část

Centrálním prvkem nového řešení bude systém virtualizace, v ideálním případě instalovatelný a aktualizovatelný formou balíčkovacího nástroje operačního systému Debian GNU/Linux. Případná dodatečná konfigurace virtualizačního systému bude patrně závislá na potřebách plynoucích z implementace.

Z provázanosti na ostatní části Virtuální laboratoře počítačových sítí vyplývá nutnost nalezení a instalace nástrojů, jež umožní:

- Síťový provoz jednotlivých virtuálních strojů v samostatných VLAN.

- Přístup na konzoli jednotlivých virtuálních strojů skrze lokální porty serveru.

Každý spuštěný virtuální stroj tak bude mít svoji konzoli přístupnou skrze daný lokální port serveru a jeho síťový provoz bude probíhat na samostatné VLAN. Jednotlivé spuštěné virtuální stroje budou definovány pomocí následujících parametrů:

- Identifikátor stroje – jeho název či číslo, např. virtualni_stroj_1 nebo pouze 1
- Paměť stroje – kolik paměti RAM bude stroji poskytnuto, např. 64 MB
- Diskový obraz stroje – z kterého image se má stroj spustit, např. debian.img
- Počet síťových rozhraní stroje – kolik síťových rozhraní má stroj obsahovat, např. 3
  - První síťové rozhraní – pracující na samostatné VLAN 1
  - Druhé síťové rozhraní – pracující na samostatné VLAN 2
  - Třetí síťové rozhraní – pracující na samostatné VLAN 3
- Port konzole stroje – na kterém portu hostitelského serveru je přístupná konzole stroje, např. 12345

Takto definované virtuální stroje budou s okolními moduly Virtuální laboratoře počítačových sítí spojeny pomocí trunk linky jdoucí ze serveru. Jednotliví uživatelé pak budou pracovat s těmito virtuálními stroji pomocí telnetu, jenž bude připojen na jejich konzole skrze lokální port serveru. Například takto:

```
$ telnet 127.0.0.1 12345
```

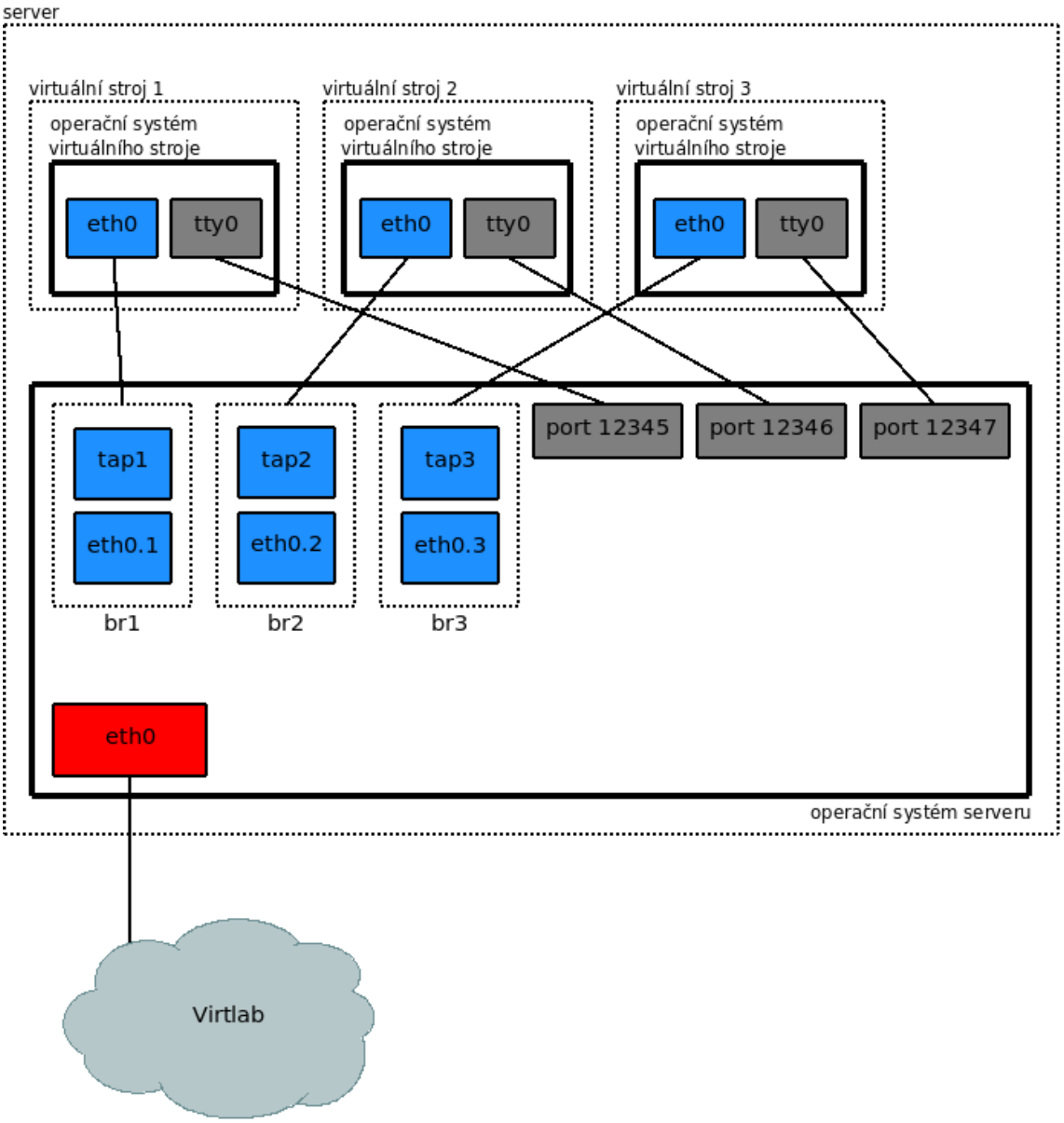
Tento příkaz by měl uživatele tedy připojit na konzoli virtuálního stroje, jež se skrývá za portem 12345. Celkové schéma nového řešení je vidět na obrázku 1.

2.2.3 Způsob ovládání

Vzhledem k provázanosti jednotlivých částí Virtuální laboratoře počítačových sítí je dobré dodržet dosavadní systém ovládání. Nové řešení bude tudíž taktéž využívat ovládání virtualizačního a operačního systému formou volání skriptů s potřebnými příkazy a parametry.

Vstupem těchto skriptů budou příkazy a případné parametry pro ovládní virtuálních strojů. Jejich výstupem pak posloupnost příkazů pro operační systém a systém virtualizace, jež zajistí požadovanou operaci.

Pro ovládání virtuálních strojů je nutné vytvořit nástroje pro jejich spuštění, ukončení, restart, výpis informací o běžících strojích, ukládání, načítání a mazání.



Obrázek 1: Celkové schéma nového řešení

2.3 Nalezení vhodného systému

Při výběru vhodného virtualizačního systému je nutné vycházet z pořadavků na nové řešení. Existujících systémů je na současném trhu mnoho a tudíž je nutné výběr zúžit jen na několik z nich. Dobrým kritériem je výběr jen těch nejznámějších a nejrozšířenějších zmíněných v kapitole 1.2.

Toto kritérium výběru zůžilo vhodné kandidáty pro budoucí virtualizační systém na: KVM, Windows Virtual PC, VirtualBox, VMware a Xen.

V následujících kapitolách budou jednotlivé systémy rozebrány a zhodnoceny vhledem k vhodnosti použití. Hodnotícími kritérii jsou požadavky na nové řešení vyplývající z potřeb Virtuální laboratoře počítačových sítí. Princip bude takový, že popsány budou ty vlastnosti díky nímž jednotlivé virtualizační systém nevyhovují požadavkům. Dále budou uvedeny případné další aspekty rozhodující o vhodnosti jejich použití. Ty požadované vlastnosti, jež systémy splňují, uváděny nebudou.

Většina požadovaných vlastností je dnes však již standardem moderních virtualizačních systémů. O případném vítězi tak pravděpodobně rozhodnou vlastnosti jako kvalitní dokumentace, dlouhodobá podpora ze strany výrobců hardwaru a softwaru, aktivní vývoj do budoucna apod.

2.3.1 KVM

Tento open source systém splňuje všechny požadavky na nové řešení pro Virtuální laboratoř počítačových sítí. Ani po důkladnějším prostudování možností tohoto systému nebyl nalezen závažnější nedostatek, jenž by znemožňoval jeho použití. Nejenže splňuje zadané požadavky z hlediska funkčnosti, ale oproti některým dalším je i vhodnějším kandidátem vzhledem ke svobodné licenci v rámci open source. Velkou výhodou tohoto systému je také bohatá nabídka možností práce se sítí.

2.3.2 Windows Virtual PC

Systém jednoho z největší výrobců softwaru na světě Microsoftu nesplňuje požadavky v několika důležitých bodech. Nejdůležitějším faktem je, že mezi podporovanými operačními systémy hostitele, ale i hosta se nenalézá platforma linux. Tato skutečnost definitivně vyřazuje Windows Virtual PC z výběru vhodných systémů. Celý nástroj je koncipován a úzce svázán právě s platformou Microsoft Windows.

2.3.3 VirtualBox

Prvním lehkým omezením systému VirtualBox od Oracle je použití maximálně osmi síťových rozhraní. Toto číslo se však zdá z hlediska dosavadních zkušeností s provozem současného řešení jako dostatečné. V porovnání s ostatními systémy se jedná o méně rozšířený nástroj. Jeho slibná budoucnost je však momentálně doprovázena překotným vývojem, jenž přináší časté vydávání nových verzí, což pro hledané řešení není výhodou.

Jedná se také o nejpoužívanější virtualizační nástroj na desktopech platformy Linux. Na poli profesionálních řešení však nejde o příliš častý systém. Vzhledem k těmto faktům a lepší vhodnosti použití u ostatních řešení byl tento systém zamítnut.

2.3.4 VMware

Produkty této společnosti patří ke špičce v oboru virtualizačních nástrojů. Část z nich je poskytována zdarma, část pouze v rámci placené licence. Z požadavků vyplývá, že se musí jednat o bezplatné řešení. V úvahu tudíž připadá hlavně bezplatný VMware Server. Tento produkt je zdarma poskytován převážně s účelem získání zákazníka pro koupi placených, lépe vybavených produktů. Oproti ostatním open source řešením však VMware Server nenabízí žádnou stěžejní či vyjímečnou funkci navíc, tudíž je dána přednost preferovanému svobonému softwaru.

2.3.5 Xen

Na tomto open source systému je postaveno současné řešení, jež z výše popsaných důvodů (viz kapitola 1.3) již nadále nesplňuje současné požadavky. Ani po prostudování změn v novějších verzích Xenu oproti té současně používané však nenastává pádný důvod, jenž by rozhodl o setrvání u tohoto virtualizačního nástroje. Nejedná se o zásadně nevyhovující systém, ovšem v současné době se nabízí použití lépe vyhovujícího KVM. Důvody tohoto rozhodnutí o změně virtualizačního systému jsou shrnuty v následující kapitole.

2.4 Zdůvodnění výběru KVM

Pomineme-li Windows Virtual PC, jsou ostatní systémy vyhovující z hlediska funkčních požadavků na nové řešení. Ostatně, jak již bylo zmíněno, většina požadovaných vlastností je dnes již standardem moderních virtualizačních systémů. Rozhodujícími se tak stala nepřímá kritéria jako je licence, součaná podpora ze strany výrobců hardwaru a softwaru apod.

Jako nejvhodnější byl zvolen systém KVM. Mezi jeho nesporné výhody patří zejména začlenění do jádra Linuxu. To zaručuje dostatečnou podporu této platformy, pravidelnou aktualizaci s každým vydáním nového jádra a mnoho dalších z toho plynoucích výhod.

Dalším rozdílem, který byl pro potřeby nového řešení shledán jako výhoda je princip virtualizace. Hardwarově asistovaná virtualizace v podání KVM se jeví vhodnější oproti paravirtualizaci Xen. Ta umožňuje lépe simulovat reálnou počítačovou síť, nabízí i možnost použití širšího spektra platforem hosta atd.

Možnou výhodou do budoucna je také příklon významných softwarových společností jako například Red Hat či IBM, jež dávají přednost virtulazačnímu systému KVM před doposud preferovaným Xenem. I tento fakt vedl k rozhodnutí o vítězství KVM. Tyto velké korporace povětšinou zaručují kvalitní a dlouhodobý vývoj, dostatečnou podporu atd.

Nelze ovšem říci, že by KVM bylo jedinou možnou volbou. I ostatní systémy s vyjímkou Windows Virtual PC by umožnily splnit současné zadání, přesto však KVM se jeví býti po všech stránkách nejvhodnějším nástrojem.

2.5 Detailnější představení KVM

Jak již bylo řečeno, tak systém KVM je součástí linuxového jádra od verze 2.6.20. Na vývoji KVM se v současnosti nejvíce podílí firma Red Hat s 52 % a IBM s 8 % na druhém místě. Jeho základem je systém Qemu, který je dále rozšířen o hardwarově asistovanou virtualizaci pro dosažení lepších výsledků. Jednotlivé příkazy se proto liší pouze v tom, zda je voláno Qemu či KVM. Dle toho se poté virtuální stroj spustí pomocí plné či hardwarově asistované virtualizace.

Následující postupy a příkazy[4] jsou určeny pro současné verze operačního systému Debian GNU/Linux a virtualizačního systému KVM. V ostatních systémech této platformy budou příkazy pravděpodobně stejné nebo velmi podobné v závislosti na odlišnostech jednotlivých systémů.

V následujícím textu bude dodržena standardní konvence pro rozlišení práv nutných k provedení příkazu. Tedy `$` pro standardního uživatele a `#` pro uživatele root.

2.5.1 Zjištění podpory virtualizace v procesoru

O tom, zda můžeme použít KVM či se musíme spokojit pouze s Qemu se přesvědčíme příkazem:

```
$ egrep '^flags.*(vmx|svm)' /proc/cpuinfo
```

Pokud nám příkaz něco vypíše, pak procesor daného stroje obsahuje podporu virtualizace a můžeme použít KVM. V opačném případě se musíme spokojit pouze s Qemu.

2.5.2 Instalace

Instalace na většině operačních systémů platformy Linux probíhá skrze balíčkovací systém. Nainstalujeme jej proto pomocí příkazu:

```
# aptitude install kvm
```

Obdobně tak můžeme nainstalovat i případné další potřebné balíčky plynoucí z implementace. Pokud se tak nestalo automaticky je ještě potřeba zavést jaderné moduly *kvm.ko* a *kvm-intel.ko* nebo *kvm-amd.ko* v závislosti na použitém typu procesoru. Moduly lze zavést příkazem:

```
# modprobe kvm
```

2.5.3 Vytvoření obrazu disku

Před spuštěním virtuálního stroje je nejprve potřeba vytvořit obraz jeho disku. Ten vytvoříme příkazem:

```
$ kvm-img create -f qcow2 debian.img 7G
```

Tento příkaz vytvoří obraz disku s názvem debian.img o maximální velikosti 7 GB. Parametr -f nám určí formát tohoto obrazu. Nejčastěji používaný qcow2 mění dynamicky svojí velikost dle aktuálně zabraného místa.

Podrobný popis jednotlivých parametrů a jejich hodnot nalezneme v manuálových stránkách.

2.5.4 Spuštění virtuálního stroje

Na připravený obraz disku z předešlé kapitoly můžeme nyní nainstalovat operační systém hosta a poté jej začít používat. Pro počáteční instalaci je nejprve potřeba jej spustit například s virtuální mechanikou CD-ROM, ze které nabootujme instalační obraz požadovaného operačního systému.

Pomocí vzorového příkladu si obecně popíšeme, jak se virtuální stroj s námi požadovanými parametry spouští. Úmyslně je zde uvedena složitější konfigurace pro lepší představu nabízených možností. Akci provedeme příkazem:

```
   $ kvm \
-name virtualnistroj1 \
-smp 2 \
-snapshot \
-hda debian.img \
-cdrom debian.iso \
-m 128 \
-net nic,vlan=1,macaddr=DE:AD:BE:EF:12:27 \
-net tap,vlan=1,ifname=tap1,script=no \
-serial tcp::12345,server,nowait \
-monitor tcp::12346,server,nowait \
-boot c \
-usb \
-no-acpi \
-localtime \
-nographic \
-daemonize
```

Nyní si popíšeme jednotlivé parametry.

- `-name virtualni_stroj_1` – nastaví jméno virtuálního stroje

- `-smp 2` – nastaví počet využívaných procesorů hostitele
- `-snapshot` – spustí virtuální stroj jako snapshot
- `-hda debian.img` – nastaví obraz pevného disku
- `-cdrom debian.iso` – nastaví obraz disku CD-ROM
- `-m 128` – udává velikost paměti RAM v MB
- `-net nic,vlan=1,macaddr=DE:AD:BE:EF:12:27` – vytvoří síťový adaptér, nastaví jeho MAC adresu a číslo VLAN
- `-net tap,vlan=1,ifname=tap1,script=no` – udává způsob připojení síťového adaptéru, číslo TAP rozhraní a číslo VLAN
- `-serial tcp::12345,server,nowait` – udává číslo portu na kterém bude bez čekání naslouchat sériový port
- `-monitor tcp::12346,server,nowait` – udává číslo portu na kterém bude bez čekání naslouchat Monitor, jenž je popsán v následující kapitole
- `-boot c` – určí zařízení ze kterého stroj nabootuje
- `-usb` – aktivuje USB ovladač
- `-no-acpi` – deaktivuje ACPI
- `-localtime` – nastavení hodiny hosta dle hostitele
- `-nographic` – vypne veškerý grafický výstup
- `-daemonize` – spustí virtuální stroj jako démona

I přesto, že tento příklad zahrnuje pouze zlomek nabízených parametrů je názorně vidět, že možnosti pro nastavení virtuálního stroje dle vlastních potřeb jsou velké. Podrobný popis jednotlivých parametrů a jejich hodnot nalezneme v manuálových stránkách.

2.5.5 Ovládání skrze Monitor

Každý virtualizační systém zpravidla obsahuje nástroj pro ovládání virtuálních strojů. V případě KVM se nazývá Monitor. Existuje jak v grafické podobě, tak i variantě pro příkazový řádek. Pomocí něj lze realizovat operace jako získání různých informací o běžícím stroji, připojení USB zařízení, vytvoření otisku obrazovky, vypnutí stroje, restart stroje, uložení stroje atd. Podrobný popis opět najdeme v manuálových stránkách.

Práce v příkazovém řádku spočívá v napojení na speciální konzoli virtuálního stroje, která interpretuje jednotlivé příkazy Monitoru.

2.5.6 kQemu

V případě absence podpory virtualizace ze strany procesoru se musíme spokojit pouze s plnou virtualizací pomocí Qemu. Tento systém však sám o sobě nedosahuje příliš velkého výkonu.

Tento stav lze vylepšit použitím kQemu. Jedná se o akcelerační modul jádra sloužící pro urychlení systému Qemu. Zavedení tohoto jaderného modulu je však vzhledem k výchozí rychlosti Qemu takřka nutností v případě, že jej chceme reálně používat.

Od verze Qemu 0.12.0 již však není kQemu nadále podporováno a je doporučeno použít akceleraci skrze KVM.

Instalace kQemu a zavedení jaderného modulu probíhá následovně. Nejprve je nutné nainstaloval balíček *module-assistant* stejným způsobem, jako již bylo uvedeno výše. Poté již dokompilujeme *kqemu-modules-** balíček skrze Module Assistant následovně:

```
# m-a a-i kqemu
```

Na konec je nutné přidat kQemu do konfiguračního souboru */etc/modules* pro automatický start a zavést jeho jaderný modul. To prvedeme následovně:

```
# echo kqemu >> /etc/modules
# modprobe kqemu
```

Pro plné využití je je ještě třeba při volání Qemu použít parametr:

```
-kernel-kqemu
```

3 Návrh implementace

Po vypracování obecné analýzy je dalším krokem vývoje nového řešení návrh implementace. Tato část vývoje řeší problém, jak naimplementovat nové řešení dle univerzální analýzy na konkrétní zvolené prostředí.

Bude zde rozebrán celý koncept nového řešení a následně pak jeho dílčí části. Na závěr této kapitoly se také podíváme na možnosti rozšíření do budoucna.

3.1 Návrh implementačního prostředí

Vzhledem k výběru operačního systému Debian GNU/Linux a požadavkům na ovládání pomocí skriptů jsem volil mezi těmi programovacími či skriptovacími jazyky, jež budou tuto podmínku splňovat. Dalším aspektem, který bylo třeba brát v úvahu je možné budoucí rozšíření tohoto nového řešení dalšími lidmi pracujícími na projektu Virtuální laboratoře počítačových sítí. Pokud by ovládání probíhalo skrze nepříliš rozšířený jazyk, mohlo by v budoucnu dojít k obtížné situaci při jeho rozvoji. Po zvážení těchto nároků jsem zvolil obecně známý skriptovací jazyk bash, který vyhovuje zmíněným požadavkům.

Dále byla řešena otázka, zdali virtualizační systém ovládat pomocí sady skriptů přímo, skrze jeho vlastní nástroje a rozhraní API nebo využít některý z univerzálních ovládacích nástrojů. Tím nejznámějším z nich je knihovna libvirt[8], jež poskytuje jednotné ovládací rozhraní pro různé virtualizační technologie a její konzolové rozhraní virsh. Po důkladné úvaze, prostudování možností i přívětivosti nástrojů a rozhraní API virtualizačního systému KVM, podložené aktuálními a možnými budoucími požadavky jsem zvolil přímé ovládání, nikoliv skrze nástroj třetích stran co by prostředníka.

3.2 Celková struktura

Při návrhu nového řešení je třeba stanovit několik implementačních pravidel, jejichž důsledným dodržením dosáhneme odpovídající kvality řešení a jeho snadné rozšiřitelnosti do budoucna v rámci dalšího rozvoje Virtuální laboratoře počítačových sítí.

Je potřeba navrhnout a dodržovat jednotnou souborovou strukturu, dostatečné členění jednotlivých skriptů na menší celky, centrální místo pro nastavení globálních parametrů virtualizace, srozumitelnost a čistotu kódu jednotlivých skriptů atd.

Z požadavků zadání vyplynulo, že po novém řešení se z hlediska implementace se požaduje vytvořit následující sadu nástrojů pro ovládání virtualizačního systému:

- Spouštění – nástroj spouštějící jednotlivé virtuální stroje s požadovanými parametry
- Ukončování – nástroje ukončující dále již nepotřebné virtuální stroje
- Restartování – nástroje restartující zablokované či jinak problémové virtuální stroje
- Výpis informací – nástroje vypisující informace o právě běžících strojích

- Ukládání, načítání a mazání – nástroje ukládající, načítající a mazající obrazy virtuálních strojů pro pozdější opětovné použití

Tyto skripty budou volány za pomocí dodatečných parametrů upravujících jejich chování dle aktuálních požadavků.

3.2.1 Souborová struktura

Základním kamenem je vhodné navržení souborové, tedy i adresářové struktury. Dále je potřeba z hlediska přehlednosti a snadné upravitelnosti rozdělit skripty do jednotlivých souborů.

Po důkladné úvaze byla adresářová struktura navržena takto:

- *instalacni_disky* – adresář obsahující obrazy instalačních disků
- *skripty* – adresář obsahující jednotlivé skripty
  - *kvm.sh*
  - *kvm_delete.sh*
  - *kvm_funkce.sh*
  - *kvm_hardreset.sh*
  - *kvm_hardresetall.sh*
  - *kvm_hardstop.sh*
  - *kvm_hardstopall.sh*
  - *kvm_kontrola_parametru_cas.sh*
  - *kvm_kontrola_parametru_disk.sh*
  - *kvm_kontrola_parametru_pamet.sh*
  - *kvm_kontrola_parametru_rozhrani.sh*
  - *kvm_kontrola_parametru_stroj.sh*
  - *kvm_load.sh*
  - *kvm_parametry.sh*
  - *kvm_reset.sh*
  - *kvm_resetall.sh*
  - *kvm_save.sh*
  - *kvm_show.sh*
  - *kvm_showall.sh*
  - *kvm_start.sh*
  - *kvm_stop.sh*
  - *kvm_stopall.sh*

  - *spusteni_bez_snaspshotu.sh*

- *ulozene_snapshoty* – adresář obsahující uložené snapshoty
- *virtualni_disky* – adresář obsahující obrazy disků

Detailní popis jednotlivých skriptů a jejich parametrů najdeme v následujících kapitolách.

3.2.2 Centrální skript

Centrálním spustitelným skriptem je *kvm.sh*. Pomocí volání tohoto skriptu a jeho parametrů dojde k provední veškerých nabízených a požadovaných operací.

Na záčátku tohoto skriptu dojde k načtení zadaných parametrů a jejich nastavení v rámci globálních proměných. Načítají se parametry: operátor, číslo stroje nebo čas, paměť stoje, disk stroje a počet síťových rozhraní stroje. Pokud některý z parametrů nebyl zadán, nastaví se na výchozí nulovou hodnotu. Počet a druh volaných parametrů tak závisí na požadované operaci.

Dále pak dojde k prolinkování mezi skripty *kvm_funkce.sh* a *kvm_parametry.sh*. První z nich umožní ostatním částem skriptu volat globální funkce, druhý pak nastaví globální promenné.

Na závěr skriptu pak dojde k provedení přepínače case, který dle prvního parametru zavolá požadovanou část skriptu. Volání jednotlivých funkcí pak tedy vypadá například takto:

```
# ./kvm.sh stopall 03:15:00
```

První parametr pak může nabývat hodnot: *start, stop, hardstop, stopall, hardstopall, reset, hardreset, resetall, hardresetall, show, showall, save, load a delete*. V případě neznámého parametru dojde k vypsání chybové hlášky s nápovědou správného zadání.

Vzhledem k tomu, že v jednotlivých skriptech bude docházet k volání příkazů vyžadujícímu práva uživatele root je nutné jej s těmito právy pokaždé spouštět.

3.2.3 Globální funkce

Globální funkce se nalézají v souboru *kvm_funkce.sh*. Jedná se o pomocné funkce volané z jednotlivých částí skriptů.

Tvorba náhodné MAC adresy virtuálního stroje
Argumenty: žádné

```
function mac() {

  echo $(echo -n DE:AD:BE:EF ; for i in `seq 1 2`  ;  do echo -n `echo ":$RANDOM$RANDOM" |
      cut -n -c -3` ; done)
```

```
}
```

Výpis 1: Globální funkce mac

Tato funkce vytvoří pří každém svém volání unikátní MAC adresu s jednotným prefixem DE:AD:BE:EF. Takto vytvořené MAC adresy jsou následně přidělovány jednotlivým spouštěným virtuálním strojům.

Tvorba jednorozměrného pole pro čísla TAP, VLAN, bridge
Argumenty: číslo virtuálního stoje, maximální počet rozhraní virtuálního stoje

```
function pole_cisel_stroje () {

    for (( i=1;i<=$2;i++)); do

        cislo=$(($1 * $2 - ($2 - $i)))
        pole[( $i-1)]=$cislo

    done

    pole_cisel_stroje =(${pole[@]})

}
```

Výpis 2: Globální funkce pole_cisel_stroje

Jednotlivé spouštěné virtuální stroje mají dynamicky se měnící počet síťových adaptérů. Pro vytvoření TAP rozhraní, VLAN rohraní a bridge na straně hostitele, s nimiž mohou jednotlivé síťové adaptéry virtuálního stroje pracovat, je tudíž zapotřebí vytvořit sadu čísel pro jejich jednotné označení. Tato sada čísel musí být samozřejmě pro každý stroj jiná.

Tato funkce vytvoří pole čísel, které se dále mohou použít při vytvoření oněch TAP rozhraní, VLAN rohraní a bridge. Při každém volání funkce jsou předány dva argumenty: číslo virtuálního stroje a maximální počet rozhraní virtuálního stoje, který je pro všechny stroje nastaven stejně v globálním nastavení parametrů. Následně dojde k vytvoření jednorozměrného pole o velikosti maximálního počtu rozhraní s čísly, jež jsou rezervovány pro udávaný stroj v prvním argumentu.

Pokud je například zavolána funkce s argumenty 2 a 3. Tak dojde k vytvoření pole s čísly 4, 5 a 6. Číslo 3 tak udává maximální počet rozhraní, tudíž tři čísla. Číslo 2 udává číslo stroje, pro který je toto pole vytvořeno. Čísla 1, 2 a 3 náleží stroji prvnímu, takže čísla 4, 5 a 6 pak tomuto stroji druhému. Tento stroj pak může být připojen na TAP rozhraní 4, 5 a 6, jež budou pracovat ve VLAN 4, 5, a 6 a TAP s VLAN budou společně svázány bridge 4, 5, a 6.

V případě zavolání funkce s argumenty 2 a 4 pak dojde k vytvoření pole s čísly 5, 6, 7 a 8. Tato čísla jsou poté opět rezervována druhému stroji. Obdobě pak pro stroje další.

Tvorba TAP rozhraní
Argumenty: číslo TAP

```
function tap_start () {

    tunctl -t tap$1 1>/dev/null 2>/dev/null
    ip link set dev tap$1 up 1>/dev/null 2>/dev/null

    logger -i -t $log_nazev -p $log_umisteni "Tvorba tap rozhraní: tap$1"

}
```

Výpis 3: Globální funkce tap_start

Tato funkce vytvoří a aktivuje TAP rozhraní s číslem udávaným v argumentu. Následně pak tuto událost zapíše do log souboru.

Pokud je například zavolána funkce s argumentem 3 vytvoří se a aktivuje TAP rozhraní tap3.

Zrušení TAP rozhraní
Argumenty: číslo TAP

```
function tap_stop() {

    tunctl -d tap$1 1>/dev/null 2>/dev/null

    logger -i -t $log_nazev -p $log_umisteni "Zrušení tap rozhraní: tap$1"

}
```

Výpis 4: Globální funkce tap_stop

Tato funkce zruší TAP rozhraní s číslem udávaným v argumentu. Následně pak tuto událost zapíše do log souboru.

Tvorba VLAN rozhraní
Argumenty: číslo VLAN

```
function vlan_start () {

    vlan=`expr $vlan_offset + $1`

    vconfig add eth0 $vlan 1>/dev/null 2>/dev/null

    logger -i -t $log_nazev -p $log_umisteni "Tvorba vlan rozhraní: eth0.$vlan"

}
```

Výpis 5: Globální funkce vlan_start

Tato funkce vytvoří VLAN rozhraní pro eth0 s číslem udávaným v argumentu sečteným s VLAN offsetem z globálního nastavení parametrů. Následně pak tuto událost zapíše do log souboru.

Pokud je například zavolána funkce s argumentem 3 a VLAN offset je nastaven například na 1000 vytvoří se VLAN rozhraní eth0.1003.

Zrušení VLAN rozhraní
Argumenty: číslo VLAN

```
function vlan_stop() {

    vlan=‘expr $vlan_offset + $1‘

    vconfig rem eth0.$vlan 1>/dev/null 2>/dev/null

    logger -i -t $log_nazev -p $log_umisteni ”Zrušení vlan rozhraní: eth0.$vlan”

}
```

Výpis 6: Globání funkce vlan_stop

Tato funkce zruší VLAN rozhraní pro eth0 s číslem udávaným v argumentu sečteným s VLAN offsetem z globálního nastavení parametrů. Následně pak tuto událost zapíše do log souboru.

Tvorba bridge rozhraní
Argumenty: číslo bridge

```
function bridge_start () {

    vlan=‘expr $vlan_offset + $1‘

    brctl addbr br$1 1>/dev/null 2>/dev/null
    brctl addif br$1 tap$1 eth0.$vlan 1>/dev/null 2>/dev/null

    logger -i -t $log_nazev -p $log_umisteni ”Tvorba bridge: br$1”

}
```

Výpis 7: Globání funkce bridge_start

Tato funkce vytvoří bridge s číslem udávaným v argumentu. Do tohoto bridge následně přiřadí odpovídající TAP a VLAN rozhraní. Následně pak tuto událost zapíše do log souboru.

Pokud je například zavolána funkce s argumentem 3 a VLAN offset je nastaven například na 1000, vytvoří se bridge br3, do nejž jsou přiřazeny rozhraní tap3 a eth0.1003.

Zrušení bridge rozhraní
Argumenty: číslo bridge

```
function bridge_stop() {

    brctl delbr br$1 1>/dev/null 2>/dev/null

    logger -i -t $log_nazev -p $log_umisteni "Zrušení bridge: br$1"

}
```

Výpis 8: Globání funkce bridge_stop

Tato funkce zruší bridge s číslem udávaným v argumentu. Následně pak tuto událost zapíše do log souboru.

3.2.4 Centrální nastavení parametrů

Globální parametry se nalézají v souboru *kvm_parametry.sh*. Jedná se o místo, které slouží k centrálnímu nastavení jednotlivých parametrů virtualizace.

Nastavit lze tyto následující parametry:

- *stroj_min* – minimální číslo stroje (výchozí hodnota: 1)
- *stroj_max* – maximální číslo stroje, neboli maximální počet strojů (výchozí hodnota: 30)
- *pamet_min* – minimální velikost paměti RAM v MB (výchozí hodnota: 32)
- *pamet_max* – maximální velikost paměti RAM v MB (výchozí hodnota: 128)
- *rozhrani_min* – minimální počet síťových rozhraní (výchozí hodnota: 1)
- *rozhrani_max* – maximální počet síťových rozhraní (výchozí hodnota: 3)
- *disk_cesta* – relativní cesta k adresáři s obrazy disků (výchozí hodnota: ../virtualni_disky/)
- *jmeno_prefix* – prefix jména stroje (výchozí hodnota: virtualni_stroj_)
- *port_terminal_offset* – offset pro výpočet lokálního portu s terminálem stroje (výchozí hodnota: 50000)
- *port_monitor_offset* – offset pro výpočet lokálního portu s monitorem stroje (výchozí hodnota: 55000)
- *vlan_offset* – offset pro výpočet čísel VLAN (výchozí hodnota: 4000)
- *log_nazev* – název pro záznam do log souboru (kvm_virtualizace)
- *log_umisteni* – určení log souboru (výchozí hodnota: daemon.info)

3.3 Spouštění

Spouštění jednotlivých virtuálního stojů zajišťuje nástroj Start. Jeho podrobnější popis nalezneme v následujících kapitolách.

3.3.1 Příkaz nástroje Start

Spuštění nástroje Start probíhá skrze příkaz:

```
# ./kvm.sh start 1 64 debian.img 3
```

3.3.2 Parametry příkazu nástroje Start

Všechny níže popsané parametry jsou při volání nástroje Start povinné. Jejich možné hodnoty jsou omezeny ve skriptu *kvm_parametry.sh*, kde lze nastavit jejich minimální či maximální hodnoty apod.

- První parametr – volaný nástroj, tedy Start
- Druhý parametr – číslo spouštěného stroje
- Třetí parametr – velikost paměti RAM stroje v MB
- Čtvrtý parametr – obraz disku stroje
- Pátý parametr – počet síťových rozhraní stroje

3.3.3 Popis činnosti skriptu nástroje Start

Nástroj Start se nalézá ve skriptu *kvm_start.sh*.

Na záčátku tohoto skriptu dojde ke kontrole správnosti zadaných parametrů zavoláním pomocných skriptů *kvm_kontrola_parametru_stroj.sh*, *kvm_kontrola_parametru_pamet.sh*, *kvm_kontrola_parametru_disk.sh* a *kvm_kontrola_parametru_rozhrani.sh*. Pokud dojde k zadání neplatných parametrů, je činnost skriptu ukončena a je vypsána chybová hláška s nápovědou správných hodnot.

Dále dojde v cyklu for k vytvoření potřebného počtu TAP rozhraní. To probíhá nejprve zavoláním globální funkce *pole_cisel_stroje*, jež nám připraví výše popsané pole čísel pro spouštěný stroj. Tato jednotlivá čísla jsou pak následně postupně předávána globální funkci *tap_start*, jež vytvoří jednotlivá TAP rozhraní. Počet takto vytvářených rozhraní je definován počtem průběhů tohoto cyklu for v závislosti na zadaném parametru pro spouštění nového stroje.

Pokud je vyžadováno například pouze jediné síťové rozhraní, tak tento cyklus proběhne pouze jednou a vytvoří jediné TAP rozhraní jehož číslo odpovídá první buňce z pole čísel pro spouštěný stroj.

Stejným způsobem je dále vytvořen potřebný počet VLAN rozhraní a bridge.

Následně dojde opět v cyklu for k přípravě parametrů, jenž budou použity při finálním volání příkazu pro virtualizační systém KVM. Dle počtu požadovaných síťových rozhraní stroje a za pomocí globálních funkcí *pole_cisel_stroje* a *mac* dojde k vytvoření té části parametrů, jež definuje síťová rozhraní virtuálního stroje.

Poté dle výše uvedeného příkazu pro spuštění nástroje Start a vložení předpřipravené síťové části parametrů dojde k sestavení následujícího příkazu pro KVM:

```
   $ kvm \
-name virtualni_stroj_1 \
-snapshot \
-hda ../virtualni_disky/debian.img \
-m 64 \
-net nic,vlan=4001,macaddr=DE:AD:BE:EF:13:14 \
-net tap,vlan=4001,ifname=tap1,script=no \
-net nic,vlan=4002,macaddr=DE:AD:BE:EF:17:23 \
-net tap,vlan=4002,ifname=tap2,script=no \
-net nic,vlan=4003,macaddr=DE:AD:BE:EF:93:13 \
-net tap,vlan=4003,ifname=tap3,script=no \
-serial tcp::50001,server,nowait \
-monitor tcp::55001,server,nowait \
-boot c \
-localtime \
-nographic \
-daemonize
```

Kadý virtuální stroj spouštěný pomocí nástroje Start je tedy konfigurován následovně:

- je nastaveno jeho jméno dle čísla stroje,
- je spuštěn jako snapshot,
- je nastaven obraz disku,
- je nastavena velikost paměti RAM,
- jsou nastaveny jednotlivé síťové adaptéry,
- je přesměrován výstup sériového portu s konzolí na lokální port hostitele,
- je přesměrován Monitor na lokální port hostitele,
- je nastaveno zařízení pro boot,
- jsou nastaveny hodiny na čas hostitele,
- je vypnut veškerý grafický výstup
- a je spuštěn jako démon.

Na závěr dojde k vypsání informační hlášky o provedeném příkazu a následně je pak tato událost zapsána do log souboru.

3.4 Ukončování

Jednotlivé či hromadné ukončování virtuálních stojů zajišťují nástroje Stop, HardStop, StopAll a HardStopAll. Nástroj Stop ukončuje běh virtuálního stroje standardním způsobem skrze Monitor virtualizačního systému. Pro případ nefunkčnosti tohoto korektního způsobu je tu ještě nástroj HardStop, který virtuální stroj ukončí nestandardně skrze operační systém hostitele. Další dva nástroje StopAll a HardStopAll jsou jejich obdobou pro hromadné ukončování všech strojů naráz. Jejich podrobnější popis nalezneme v následujících kapitolách.

Nástroje StopAll a HardStopAll mají ještě možnost zadání časového limitu. V tomto případě dojde k ukončení pouze těch strojů, jež běží déle, než je zadaný časový limit. Tato možnost slouží pro ukončování strojů, jež by v systému mohly běžet z jakéhokoliv důvodu nepřiměřeně dlouho.

3.4.1 Příkaz nástroje Stop

Spuštění nástroje Stop probíhá skrze příkaz:

```
# ./kvm.sh stop 1
```

3.4.2 Parametry příkazu nástroje Stop

Všechny níže popsané parametry jsou při volání nástroje Stop povinné. Možné hodnoty druhého parametru jsou omezeny ve skriptu *kvm_parametry.sh*, kde lze nastavit minimální či maximální hodnotu.

- První parametr – volaný nástroj, tedy Stop
- Druhý parametr – číslo ukončovaného stroje

3.4.3 Popis činnosti skriptu nástroje Stop

Nástroj Stop se nalézá ve skriptu *kvm_stop.sh*.

Na záčátku tohoto skriptu dojde ke kontrole správnosti zadaného parametru zavoláním pomocného skriptu *kvm_kontrola_parametru_stroj.sh*. Pokud dojde k zadání neplatného parametru, je činnost skriptu ukončena a je vypsána chybová hláška s nápovědou správné hodnoty.

Dále dojde pomocí speciálního příkazu *quit* pro Monitor k ukončení běhu virtuálního stroje. Tento příkaz je zaslán na lokální port hostitele, na němž naslouchá Monitor daného stroje.

```
echo quit | netcat 127.0.0.1 $port_monitor 1>/dev/null 2>/dev/null
```

Výpis 9: Ukončení skrze nástroj Stop

Zde se nabízel ještě speciální příkaz *system_powerdown*. Rozdíl obou dvou příkazů je v tom, že *quit* ukončuje stroj okamžitě, dalo by se říci nekorektně a *system_powerdown* simuluje stitsknutí tlačítka napájení, kdy virtuální stroj obdrží ACPI žádost o vypnutí. Vzhledem k tomu, že spouštěné virtuální stroje pracují jako snapshoty, kdy je zbytečné čekat na korektní ukončení, byl vybrán příkaz *quit*.

Poté dojde pomocí globálních funkcí *bridge_stop*, *vlan_stop* a *tap_stop* ke zrušení bridge, VLAN rozhraní a TAP rozhraní, jež daný stroj využíval.

Na závěr dojde k vypsání informační hlášky o provedeném příkazu a následně je pak tato událost zapsána do log souboru.

3.4.4 Příkaz nástroje HardStop

Spuštění nástroje HardStop probíhá skrze příkaz:

```
# ./kvm.sh hardstop 1
```

3.4.5 Parametry příkazu nástroje HardStop

Všechny níže popsané parametry jsou při volání nástroje HardStop povinné. Možné hodnoty druhého parametru jsou omezeny ve skriptu *kvm_parametry.sh*, kde lze nastavit minimální či maximální hodnotu.

- První parametr – volaný nástroj, tedy HardStop
- Druhý parametr – číslo ukončovaného stroje

3.4.6 Popis činnosti skriptu nástroje HardStop

Nástroj HardStop se nalézá ve skriptu *kvm_hardstop.sh*.

Činnost tohoto skriptu se shoduje se skriptem nástroje Stop vyjma části řešící samotné ukončování virtuálního stroje.

Zde na rozdíl od nástroje Stop využívající korektní ukončení pomocí Monitoru dojde k ukončení skrze operační systém hostitele za použití příkazu *kill*. Nejprve je získán PID ukončovaného virtuálního stroje a ten je poté pomocí příkazu *kill* ukončen.

```
pid=‘ps −AF | grep $jmeno_prefix$stroj | grep −v ’pts’ | awk ’{ print $2 }’‘
 kill −9 $pid 1>/dev/null 2>/dev/null
```

Výpis 10: Ukončení skrze nástroj HardStop

3.4.7 Příkaz nástroje StopAll

Spuštění nástroje StopAll probíhá skrze příkazy:

```
# ./kvm.sh stopall
```

nebo případně

```
# ./kvm.sh stopall 01:30:00
```

3.4.8 Parametry příkazu nástroje StopAll

První parametr je při volání nástroje StopAll povinný, druhý je volitelný.

- První parametr – volaný nástroj, tedy StopAll
- Druhý parametr – časový limit ve formátu HH:MM:SS

3.4.9 Popis činnosti skriptu nástroje StopAll

Nástroj StopAll se nalézá ve skriptu *kvm_stopall.sh*.

Nejprve dojde k rozhodnutí, která část skriptu se má provést, dle toho zda byl či nebyl zadán druhý parametr času.

Pokud čas zadán byl, dojde na záčátku ke kontrole správnosti zadaného parametru zavoláním pomocného skriptu *kvm_kontrola_parametru_cas.sh*. Pokud dojde k zadání neplatného parametru, je činnost skriptu ukončena a je vypsána chybová hláška s nápovědou správné hodnoty.

Následně je v cyklu for u všech strojů prověřena podmínka, zda doba jejich běhu nepřekračuje zadaný limit. Pokud ano, je na daný stroj zavolán nástroj Stop pro jeho ukončení.

Na závěr dojde k vypsání informační hlášky o provedeném příkazu a následně je pak tato událost zapsána do log souboru.

Pokud čas zadán nebyl, dojde k obdobnému ukončení všech strojů bez dodatečné podmínky. Zrušení všech jimi využívaných bridge, VLAN rozhraní a TAP rozhraní atd.

Veškeré ukončování činnosti strojů pomocí tohoto skriptu probíhá korektně skrze Monitor.

3.4.10 Příkaz nástroje HardStopAll

Spuštění nástroje HardStopAll probíhá skrze příkazy:

```
# ./kvm.sh hardstopall
```

nebo případně

```
# ./kvm.sh hardstopall 01:30:00
```

3.4.11 Parametry příkazu nástroje HardStopAll

První parametr je při volání nástroje StopAll povinný a druhý je volitelný.

- První parametr – volaný nástroj, tedy HardStopAll
- Druhý parametr – časový limit ve formátu HH:MM:SS

3.4.12 Popis činnosti skriptu nástroje HardStopAll

Nástroj HardStopAll se nalézá ve skriptu *kvm_hardstopall.sh*.

Činnost tohoto skriptu se shoduje se skriptem nástroje StopAll vyjma částí řešící samotné ukončování virtuálních strojů. Rozdíl je zde opět stejný jako v případě Stop a HardStop. Tedy k ukončení dojde nikoliv skrze Monitor, ale příkaz *kill* v operačním systému hostitele.

3.5 Restartování

Jednotlivé či hromadné restartování virtuálních stojů zajišťují nástroje Reset, HardReset, ResetAll a HardResetAll. Nástroj Reset restartuje běh virtuálního stroje standardním způsobem skrze Monitor virtualizačního systému. Pro případ nefunkčnosti tohoto korektního způsobu je tu ještě nástroj HardReset, který virtuální stroj restartuje nestandardně skrze operační systém hostitele. Další dva nástroje ResetAll a HardResetAll jsou jejich obdobou pro hromadné restartování všech strojů naráz. Jejich podrobnější popis nalezneme v následujících kapitolách.

3.5.1 Příkaz nástroje Reset

Spuštění nástroje Reset probíhá skrze příkaz:

```
# ./kvm.sh reset 1
```

3.5.2 Parametry příkazu nástroje Reset

Všechny níže popsané parametry jsou při volání nástroje Reset povinné. Možné hodnoty druhého parametru jsou omezeny ve skriptu *kvm_parametry.sh*, kde lze nastavit minimální či maximální hodnotu.

- První parametr – volaný nástroj, tedy Reset
- Druhý parametr – číslo resetovaného stroje

3.5.3 Popis činnosti skriptu nástroje Reset

Nástroj Reset se nalézá ve skriptu *kvm_reset.sh*.

Na záčátku tohoto skriptu dojde ke kontrole správnosti zadaného parametru zavoláním pomocného skriptu *kvm_kontrola_parametru_stroj.sh*. Pokud dojde k zadání neplatného parametru, je činnost skriptu ukončena a je vypsána chybová hláška s nápovědou správné hodnoty.

Dále dojde pomocí příkazu *ps* k uložení spouštěcího příkazu daného stroje pro KVM do pomocné proměnné. Tento stroj je poté zastaven pomocí nástroje Stop a následne je opět nanovo spuštěn ve stejné konfiguraci pomocí příkazu, jenž byl uložen v pomocné proměnné.

```
prikaz=‘ps −AF | grep qemu | grep $jmeno_prefix$stroj | grep −v ’pts’ | egrep −o ’qemu.∗’‘
$0 stop $stroj
$prikaz
```

Výpis 11: Restartování skrze nástroj Reset

Zde se nabízela ještě možnost použití speciálního příkazu *system_reset* pro Monitor. V tomto případě ovšem nedojde ke smazání starého a nahrání nového snapshotu, ale pouze k restartování virtuálního stroje se snapshotem původním. Tato vlastnost rozhodla o použití výše popsaného řešení.

Na závěr dojde k vypsání informační hlášky o provedeném příkazu a následně je pak tato událost zapsána do log souboru.

3.5.4 Příkaz nástroje HardReset

Spuštění nástroje HardReset probíhá skrze příkaz:

```
# ./kvm.sh hardreset 1
```

3.5.5 Parametry příkazu nástroje HardReset

Všechny níže popsané parametry jsou při volání nástroje HardReset povinné. Možné hodnoty druhého parametru jsou omezeny ve skriptu *kvm_parametry.sh*, kde lze nastavit minimální či maximální hodnotu.

- První parametr – volaný nástroj, tedy HardReset
- Druhý parametr – číslo resetovaného stroje

3.5.6 Popis činnosti skriptu nástroje HardReset

Nástroj HardReset se nalézá ve skriptu *kvm_hardreset.sh*.

Činnost tohoto skriptu se shoduje se skriptem nástroje Reset vyjma části řešící samotné restartování virtuálního stroje.

Zde na rozdíl od nástroje Reset využívající korektní ukončení pomocí nástroje Stop dojde k ukončení skrze nástroj HardStop. Tudíž nedojde k ukončení původního snapshotu pomocí Monitoru, ale skrze příkaz operačního systému *kill*.

```
prikaz=‘ps −AF | grep qemu | grep $jmeno_prefix$stroj | grep −v ’pts’ | egrep −o ’qemu.∗’‘
$0 hardstop $stroj
$prikaz
```

Výpis 12: Restartování skrze nástroj HardReset

3.5.7 Příkaz nástroje ResetAll

Spuštění nástroje ResetAll probíhá skrze příkaz:

```
# ./kvm.sh resetall
```

3.5.8 Parametry příkazu nástroje ResetAll

První a zároveň jediný parametr je při volání nástroje ResetAll povinný.

- První parametr – volaný nástroj, tedy ResetAll

3.5.9 Popis činnosti skriptu nástroje ResetAll

Nástroj ResetAll se nalézá ve skriptu *kvm_resetall.sh*.

Činnost tohoto skriptu se shoduje se skriptem nástroje Reset. Tento skript ovšem nerestartuje pouze jeden virtuální stroj, ale v cyklu for dojde k restartování vších strojů.

3.5.10 Příkaz nástroje HardResetAll

Spuštění nástroje HardResetAll probíhá skrze příkaz:

```
# ./kvm.sh hardresetall
```

3.5.11 Parametry příkazu nástroje HardResetAll

První a zároveň jediný parametr je při volání nástroje HardResetAll povinný.

- První parametr – volaný nástroj, tedy HardResetAll

3.5.12 Popis činnosti skriptu nástroje HardResetAll

Nástroj HardResetAll se nalézá ve skriptu *kvm_hardresetall.sh*.

Činnost tohoto skriptu se shoduje se skriptem nástroje HardReset. Tento skript ovšem nerestartuje pouze jeden virtuální stroj, ale v cyklu for dojde k restartování všech strojů.

3.6 Výpis informací

Výpis informací o jednotlivých či všech virtuálních stojích zajišťují nástroje Show a ShowAll. Nástroj Show vypisuje informace o jednotlivých virtuálních strojích a nástroj ShowAll je pak jeho obdobou pro výpis informací o všech strojích naráz. Jejich podrobnější popis nalezneme v následujících kapitolách.

3.6.1 Příkaz nástroje Show

Spuštění nástroje Show probíhá skrze příkaz:

```
# ./kvm.sh show 1
```

3.6.2 Parametry příkazu nástroje Show

Všechny níže popsané parametry jsou při volání nástroje Show povinné. Možné hodnoty druhého parametru jsou omezeny ve skriptu *kvm_parametry.sh*, kde lze nastavit minimální či maximální hodnotu.

- První parametr – volaný nástroj, tedy Show
- Druhý parametr – číslo stroje pro výpis informací

3.6.3 Popis činnosti skriptu nástroje Show

Nástroj Show se nalézá ve skriptu *kvm_show.sh*.

Na záčátku tohoto skriptu dojde ke kontrole správnosti zadaného parametru zavoláním pomocného skriptu *kvm_kontrola_parametru_stroj.sh*. Pokud dojde k zadání neplatného parametru, je činnost skriptu ukončena a je vypsána chybová hláška s nápovědou správné hodnoty.

Dále dojde pomocí příkazu *ps* k výpisu informací o daném virtuálním stroji.

```
ps -AF | grep qemu | grep $jmeno_prefix$stroj | grep -v 'pts' | grep --color=always
    $jmeno_prefix[0-9]*
```

Výpis 13: Výpis informací skrze nástroj Show

Zde se nabízela ještě možnost použití několika speciálních příkazů Monitoru. Tato možnost byla ovšem zamítnuta vzhledem ke své složitosti, jelikož by obdobný výpis vyžadoval použití více příkazů Monitoru a použití *ps* je zcela dostatečné.

Na závěr dojde k vypsání informační hlášky o provedeném příkazu a následně je pak tato událost zapsána do log souboru.

3.6.4 Příkaz nástroje ShowAll

Spuštění nástroje ShowAll probíhá skrze příkaz:

```
# ./kvm.sh showall
```

3.6.5 Parametry příkazu nástroje ShowAll

První a zároveň jediný parametr je při volání nástroje ShowAll povinný.

- První parametr – volaný nástroj, tedy ShowAll

3.6.6 Popis činnosti skriptu nástroje ShowAll

Nástroj ShowAll se nalézá ve skriptu *kvm_showall.sh*.

Činnost tohoto skriptu se shoduje se skriptem nástroje Show. Tento skript ovšem nevypisuje informace pouze o jednom virtuálním stroj, ale v cyklu for dojde k výpisu informací o všech strojích.

3.7 Ukládání, načítání a mazání

Ukládání, načítání a mazání jednotlivých virtuálních strojů zajišťují nástroje Save, Load a Delete. Nástroj Save ukládá snapshot virtuálního stroje na server. Načítání takto uložených snapshotů umožňuje nástroj Load. Mazání takto uložených snapshotů pak nástroj Delete. Jejich podrobnější popis nalezneme v následujících kapitolách.

3.7.1 Příkaz nástroje Save

Spuštění nástroje Save probíhá skrze příkaz:

```
# ./kvm.sh save 1 jus011
```

3.7.2 Parametry příkazu nástroje Save

Všechny níže popsané parametry jsou při volání nástroje Save povinné. Možné hodnoty druhého parametru jsou omezeny ve skriptu *kvm_parametry.sh*, kde lze nastavit minimální či maximální hodnotu.

- První parametr – volaný nástroj, tedy Save
- Druhý parametr – číslo stroje pro uložení snapshotu
- Třetí parametr – název ukládaného snapshotu

3.7.3 Popis činnosti skriptu nástroje Save

Nástroj Save se nalézá ve skriptu *kvm_save.sh*.

Na záčátku tohoto skriptu dojde ke kontrole správnosti zadaného parametru zavoláním pomocného skriptu *kvm_kontrola_parametru_stroj.sh*. Pokud dojde k zadání neplatného parametru, je činnost skriptu ukončena a je vypsána chybová hláška s nápovědou správné hodnoty.

Dále dojde pomocí speciálního příkazu *savevm* pro Monitor k uložení snapshotu virtuálního stroje. Tento příkaz je zaslán na lokální port hostitele, na němž naslouchá Monitor daného stroje.

```
echo savevm $snapshot_nazev | netcat 127.0.0.1 $port_monitor 1>/dev/null 2>/dev/null
```

Výpis 14: Uložení skrze nástroj Save

Na závěr dojde k vypsání informační hlášky o provedeném příkazu a následně je pak tato událost zapsána do log souboru.

3.7.4 Příkaz nástroje Load

Spuštění nástroje Load probíhá skrze příkaz:

```
# ./kvm.sh load 1 jus011
```

3.7.5 Parametry příkazu nástroje Load

Všechny níže popsané parametry jsou při volání nástroje Load povinné. Možné hodnoty druhého parametru jsou omezeny ve skriptu *kvm_parametry.sh*, kde lze nastavit minimální či maximální hodnotu.

- První parametr – volaný nástroj, tedy Load
- Druhý parametr – číslo stroje pro načtení snapshotu
- Třetí parametr – název načítaného snapshotu

3.7.6 Popis činnosti skriptu nástroje Load

Nástroj Load se nalézá ve skriptu *kvm_load.sh*.

Na záčátku tohoto skriptu dojde ke kontrole správnosti zadaného parametru zavoláním pomocného skriptu *kvm_kontrola_parametru_stroj.sh*. Pokud dojde k zadání neplatného parametru, je činnost skriptu ukončena a je vypsána chybová hláška s nápovědou správné hodnoty.

Dále dojde pomocí speciálního příkazu *loadvm* pro Monitor k načtení nového snapshotu virtuálního stroje. Tento příkaz je zaslán na lokální port hostitele, na němž naslouchá Monitor daného stroje.

```
echo loadvm $snapshot_nazev | netcat 127.0.0.1 $port_monitor 1>/dev/null 2>/dev/null
```

Výpis 15: Načtení skrze nástroj Load

Na závěr dojde k vypsání informační hlášky o provedeném příkazu a následně je pak tato událost zapsána do log souboru.

3.7.7 Příkaz nástroje Delete

Spuštění nástroje Delete probíhá skrze příkaz:

```
# ./kvm.sh delete jus011
```

3.7.8 Parametry příkazu nástroje Delete

Všechny níže popsané parametry jsou při volání nástroje Delete povinné.

- První parametr – volaný nástroj, tedy Delete
- Druhý parametr – název mazaného snapshotu

3.7.9 Popis činnosti skriptu nástroje Delete

Nástroj Delete se nalézá ve skriptu *kvm_delete.sh*.

Pomocí příkazu operačního systému hostitele *rm* dojde ke smazání snapshotu virtuálního stroje.

```
rm $snapshot_nazev 1>/dev/null 2>/dev/null
```

Výpis 16: Mazání skrze nástroj Delete

Na závěr dojde k vypsání informační hlášky o provedeném příkazu a následně je pak tato událost zapsána do log souboru.

3.8 Obraz disku

Vzhledem k požadavkům zadaní byl vytvořen obraz disku *debian.img* nalézající se v adresáři *virtualni_disky*.

Jeho maximální velikost byla stanovena na 2 GB. Tato velikost by měla být dostatečná co se kapacity disku týká a zároveň slouží jako bezpečnostní pojistka pro úmyslné neúměrné zvětšení velikosti snapshotu. Při případném přetečení tohoto limitu dojde k včasnému pádu virtuálního stroje, nikoliv neúměrnému zatížení či dokonce pádu serveru. Za formát obrazu disku byl zvolen qcow2, který tak dynamicky mění velikost dle aktuálně zabraného místa.

Na tento obraz disku byl nainstalován operační systém Debian GNU/Linux 5.0.4 Lenny a ostatní požadovaný software z jeho repozitářů.

Vzhledem k požadavkům zadání nebyl systém nijak speciálně upravován či konfigurován. Vyjímku tak tvoří pouze přesměrování konzole na sériový port. Důvod tohoto nastavení je popsán v kapitole 4.1.2.

V operačním systému se nalézají dva uživatelské účty. Účet uživatele root s heslem cisco a standardní uživatel cnap s heslem cisco.

3.9 Spuštění bez snapshotu

Dalším doplňkovým nástrojem, který bylo vhodné implementovat, je skript pro spuštění obrazu disku bez režimu snapshotu. Tento nástroj tedy slouží k úpravě originálního obrazu disku a nalézá se ve skriptu *spusteni_bez_snaspshotu.sh*.

Spuštění tohoto nástroje probíhá pomocí příkazů:

```
# ./spusteni_bez_snaspshotu.sh start debian.img
```

nebo případně

```
# ./spusteni_bez_snaspshotu.sh stop
```

První parametr udává, zdali jde o spuštění nebo úklid (například zrušení TAP rozhraní) po spuštění virtuálního stroje. Případný druhý parametr poté určí, který obraz disku chceme ve stroji použít.

3.10 Možnosti rozšíření do budoucna

Výše navržené nové řešení ve své současné podobě představuje kompletní nástroj pro ovládání virtuálních strojů v rámci Virtuální laboratoře počítačových sítí a jejích současných potřeb. Nabízí se však i možnosti, jak toto řešení v budoucnu rozšířit a doplnit.

Momentálně je k dispozici pouze jediný obraz disku. Nabízí se zde samozřejmě možnost vytvořit další obrazy disků. Ty mohou obsahovat odlišné operační systémy či pouze potřebné úpravy či konfigurace obrazu stávajícícho.

Díky strukturovanému a přehlednému návrhu skriptů se také nabízí možnost snadného přidání nových či úprava těch stávajících. Přidání nového nástroje tak spočívá pouze v rozšíření centrálního skriptu o další přepínač case a jeho implementaci. Nabízí se například tvorba nástroje pro vytváření obrazů disků dle zadaných parametrů s následnou instalací operačního systému.

4 Implementace

V následující kapitole bude rozebrána implementace nového řešení. Popsány budou řešené problémy, finální nasazení na server a některé další věci týkající se implementace. Kompletní implementaci pak nalezneme na serveru.

4.1 Řešené problémy

V průběhu implementace nového řešení se postupně objevilo několik problémů, jež více či méně komplikovaly jeho vývoj. Následuje popis nejzávažnějších z nich.

4.1.1 Dokumentace

Během výběru nového virtualizačního systému byl samozřejmě brán zřetel i na existenci webové dokumentace s dostatečným množstvím ukázek a informací k řešeným situacím pomocí KVM a Qemu. Ta se zpočátku jevila jako dostatečná. Během samotné implementace doprovázené častými potřebami získání informací se ovšem ukázalo, že její kvalita není tak dobrá, jak se zpočátku zdálo.

Většinu ukázkových konfigurací a hlubších popisů možností jsem ve finále většinou nalezl skrze internetové vyhledávače všude jinde, jen né na domovských webových stránkách KVM a Qemu. Dobrým zdrojem byly také manuálové stránky, kde ovšem zpravidla nebývá uvedeno velké množství úkazkových příkladů.

4.1.2 Přesměrování portů

Z požadavků na nové řešení vyplynula nutnost připojení konzole jednotlivých virtuálních strojů na lokální porty hostitele. To znamená, aby se uživatel skrze lokální port serveru připojil na konzoli požadovaného virtuálního stroje, jak již bylo popsáno výše.

Z prostudování dokumentace a ukázkových příkladů na internetu vyplynulo, že pro přesměrování portů slouží následující parametr zadávaný při spuštění virtuálního stroje:

```
-redir tcp:2323::23
```

nebo v novějších verzích Qemu jeho ekvivalent

```
-net user,hostfwd=tcp:2323::23
```

Oba tyto parametry by měly propojit port 23 pro telenet virtuálního stroje na lokální port serveru 2323.

Tento postup je uváděn ve většině nalezených ukázkových příkladů i v dokumentaci, ale ani po důkladném nastudování těchto zdrojů se jej nepodařilo zprovoznit. Tentýž problém byl nalezen i v dotazech na několika fórech od dalších uživatelů, ovšem bez jediného funkčního řešení.

Vzhledem ke klíčovosti této funkce bylo potřeba vymyslet jiné náhradní řešení. Tím se na konec stalo přesměrování konzole virtuálního stroje na jeho sériový port. Tento port byl následně již bezproblémově přesměrován na lokální port hostitele.

Pro přesměrování konzole na sériový port bylo potřeba mírně upravit konfiguraci operačního systému virtuálního stroje. Nejprve byla v konfiguračním souboru */boot/grub/menu.lst* zavadeče GNU GRUB přesměrována konzole na sériový port pomocí parametru jádra *console=ttyS0,9600n8*. Upravený řádek poté vypadá následovně:

```
kernel /boot/vmlinuz-2.6.26-2-686 root=/dev/sda1 console=ttyS0,
9600n8 ro quiet
```

Dále bylo nutné přidat getty pro login na sériovém portu do konfiguračního souboru */etc/inittab*. Přidaný řáděk vypadá následovně:

```
co:2345:respawn:/sbin/getty -8 9600 ttyS0 linux
```

Na závěr je třeba zkontrolovat, případně přidat, zda je v konfiguračním souboru */etc/secutetty* povoleno přihlášení uživatele root skrze sériový port *ttyS0*.

Po této drobné úpravě operačního systému virtuálního stroje již bylo pouze potřeba zajistit přesměrování jeho sériového portu na lokální port serveru. To provedeme přidáním parametru do spouštěcího příkazu virtuálního stroje. Potřebný parametr vypadá následovně:

```
-serial tcp::50001,server,nowait
```

4.1.3 Režim síťové komunikace

Nutnost připojení virtuálních strojů do samostatných VLAN, komunikujících s okolními moduly Virtuální laboretoře počítačových sítí, plynoucí z požadavků zadání, zkomplikovala počáteční vývoj nového řešení.

Ze síťové části dokumentace virtualizačního systému Qemu se nabízela možnost připojení síťového adaptéru skrze VLAN pomocí několika režimů. Nejprve byl zvolen způsob pomocí parametru:

```
-net user
```

Tento parametr, jevící se zpočátku jako nejvhodnější, nastaví síťový adaptér do takzvaného *user-mode* režimu. Následně však bylo zjištěno, že tento mód se chová jako jednosměrný firewall a neumožňuje žádný příchozí provoz. Nepodporuje také jiné síťové protokoly nežli TCP a UDP. Nevhodným se také ukázal parametr:

```
-net socket
```

Komunikace síťového adaptéru zde probíhá pomocí TCP socketů. Ani zde se však nepodařilo dosáhnout kýženého výsledku. Oba parametry, jevící se zpočátku jako vhodné, sice fungují dle popisu v dokumentaci, ale ani u jednoho z nich se po mnoha neúspěšných pokusech nepodařilo zprovoznit připojení dle požadavků Virtuální laboratoře počítačových sítí.

Požadovaného výsledku bylo dosaženo až použitím parametru:

```
-net tap
```

V tomto režimu je virtuální stroj připojen k TAP rozhraní, které bylo propojeno pomocí bridge s VLAN rozhraním serveru. Toto řešení již umožňovalo požadovanou komunikaci mezi virtuálními počítači a okolím serveru dle zadání.

4.2 Nasazení na server

Konečná implementace byla nasazena na starší záložní server Virtuální laboratoře počítačových sítí srva10nb.vsb.cz.

Nejprve bylo ověřeno, zda tento starší server obsahuje procesor s podporou virtualizace. Tato technologie se však u procesoru tohoto záložního serveru nenalézá. Z tohoto důvodu byl nainstalován virtualizační systém Qemu a jaderný modul kQemu pro jeho akceleraci. Jak již bylo výše popsáno, tak ten se oproti KVM liší pouze v absenci podpory ze strany hardwaru. Jediný rozdíl je tak ve výkonu celého virtualizačního systému.

Úprava se tak dotkla pouze skriptu *kvm_start.sh*, kde byl přepsán spouštěcí příkaz virtuálních strojů z *kvm* na *qemu*. Jeho parametry zůstávají stejné. V případě nasazení na ostrý provozní server, jenž obsahuje podporu virtualizace v procesoru stačí opět pouze změnit tento příkaz zpět a nainstalovat KVM.

Dále byly do operačního systému serveru doinstalovány balíčky *uml-utilities*, *vlan* a *bridge-utils*, jež jsou nutné pro chod některých příkazů ovládacích skriptů a nejsou jeho standardní součástí.

Na závěr byla na server do adresáře */virtualizace* nahrána celá implementace nového řešení. To bylo následně kompletně testováno. Vzhledem k tomu, že průběžné testy probíhaly během celého vývoje jednotlivých nástrojů, nedošlo k negativnímu výsledku.

Před finálním nasazením na ostrý provozní server bude toto nové řešení nadále nějaký čas testováno na záložním serveru. Důvodem je jednak případná větší jistota bezproblémovosti provozu daná delším testováním a také fakt, že momentálně stále probíhá letní semestr a současné řešení je nyní aktivně využíváno. Případný přechod za ostrého provozu je tak příliš velkým rizikem, které by mohlo zapříčinit nefunkčnost části Virtuální laboratoře počítačových sítí.

4.3 Použité programové vybavení

Následující programové vybavení pomohlo vytvořit tuto diplomovu práci.

Vývoj nového řešení:

- Operační systém – Ubuntu GNU/Linux 9.10 Karmic Koala
- Virtualizační systém - KVM verze 84 z jádra Linuxu 2.6.31-20
- Skriptovací jazyk – bash 3.2.48
- Vývojové prostředí – gedit 2.28.0

Sestavení dokumentace:

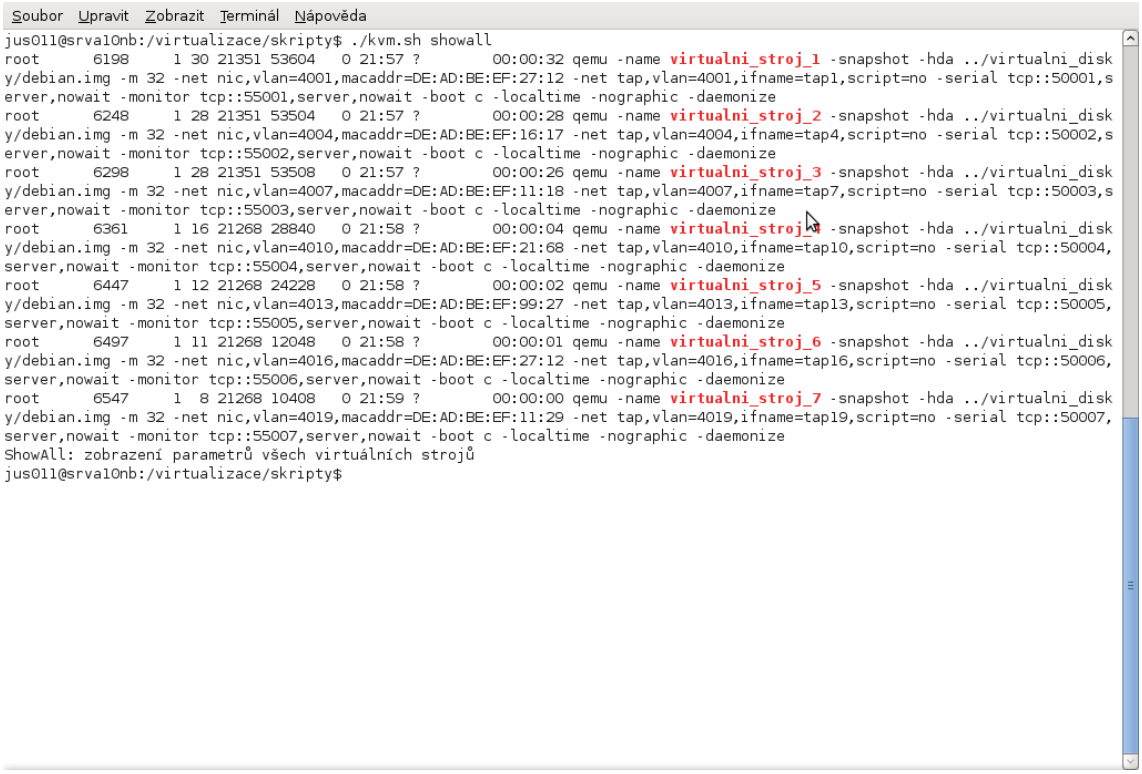
- Tvorba textu – gedit 2.28.0
- Tvorba diagramů – dia 0.97
- Tvorba obrázků – gimp 2.6.7
- Makro pro sazbu – diploma 2.2
- Sazba – LaTeX

Sestavení prezentace:

- Tvorba prezentace – OpenOffice.org 3.2.0 (Impress)

4.4 Ukázka běhu

Následující otisk obrazovky zobrazuje výpis nástroje ShowAll se sedmi běžícími virtuálními stroji. Otisk obrazovky je vidět na obrázku 2.



Obrázek 2: Otisk obrazovky

5 Závěr

Nutným předpokladem pro realizaci této práce bylo prostudovat základní teorii virtualizace. Zejména část vztahující se k virtualizaci platformy. Následovalo seznámení se současně existujícími virtualizačními systémy a nejrozšířenější z nich byly představeny z hlediska jejich současné pozice na trhu. Stávající řešení postavené na virtualizačním systému Xen již nadále nevyhovuje aktuálním požadavkům Virtuální laboratoře počítačových sítí. Problematické vlastnosti současného systému, jež vedou k vývoji nového řešení byly poté zhodnoceny.

Ze současných požadavků Virtuální laboratoře počítačových sítí byl následně vypracován návrh nového řešení serverové i aplikační části a způsob jejich ovládání. Pro tento návrh nového řešení byla poté provedena analýza nejvhodnějšího virtualizačního systému. Spoučasné požadavky ovšem jednoznačně neurčily nejvhodnějšího kandidáta. O vítězi tak musela rozhodnout až nepřímá kritéria výběru. Zvolen byl virtualizační systém KVM a tato volba byla následně podrobněji zdůvodněna. Vítězný virtualizační nástroj KVM byl poté stručně představen.

Po analýze nového řešení a výběru vhodného virtualizačního systému již následoval návrh implmentace. Nejprve došlo na výběr vhodnho implementačního prostředí a navržení celkové struktury ovládacích nástrojů. Zvoleno bylo vytvoření knihovny bash skriptů, jež budou dle uživatelských požadavků formou zadaných parametrů ovládat virtualizační a operační systém. Poté byl proveden detailní návrh skriptů jednotlivých nástrojů pro spuštění, ukončení, restart, výpis informací o běžících strojích, ukládání, načítání a mazání virtuálních strojů. Dalším krokem bylo navržení obrazu disku virtuálního stroje, dle požadavků zadání. Zhodnoceny byly také možnosti rozšíření tohoto navrhovaného řešení do budoucna jako přidání dalších odlišných či modifikovaných obrazů disku pro virtuální stroje či doplnění sady nástrojů pro jejich ovládání.

Takto vytvořený návrh nového řešení byl následně implementován ve zvoleném prostředí. Během vývoje se postupně objevilo několik problémů. Nejvýznaměnjším z nich bylo nefunkční přesměrování portů, jež bylo jedním z klíčových požadavků zadání. Hledání funkčního řešení tohoto problému zabralo poměrně dlouhou dobu. Práci také často komplikovala špatná použitelnost dokumentace virtualizačního systému a celková nepřehlednost jeho oficiálních webových stránek. Řešené problémy jsou podrobně shrnuty v samostatné kapitole. Celé nové řešení bylo nasazeno na server a úspěšně testováno.

Vývoj tohoto nového řešení pro Virtuální laboratoř počítačových sítí spočíval významnou částí také v hledání potřebných informací na internetu. Nejprve při počátečním získávání informací o virtualizačních systémech a dále pak při častém procházení dokumentací a řešení problémů a komplikací vyvstávajících během vývoje.

Konečná implementace splňuje jednotlivé požadavky zadání a její vhodný návrh umožňuje snadnou rozšiřitelnost či přizpůsobení konkrétním podmínkám do budoucna.

Závěrem bych chtěl říci, že celá tato práce mi rozšířila znalosti a dovednosti v oblasti virtualizace a operačních systému platformy Linux. Těší mě, že jsem se mohl zapojit do projektu Virtuální laboratoře počítačových sítí, a věřím, že mnou řešená část bude úspěšně sloužit jejímu dalšímu rozvoji.

6 Reference

[1] Danielle Ruest, Nelson Ruest, *Virtualizace, Podrobný průvodce*, Computer Press, a. s., Brno, 2010.

[2] Amit Singh, *An Introduction to Virtualization*, 2004, dostupné na url adrese: *http://www.kernelthread.com/publications/virtualization/*.

[3] WMvare, Inc., *Understanding Full Virtualization, Paravirtualization, and Hardware Assist*, WMvare, Inc., Palo Alto, USA, 2007.

[4] Robert Warnke, Thomas Ritzau, *qemu-kvm & libvirt*, Books on Demand GmbH, Norderstedt, Germany, 2010.

[5] KVM – oficiální webové stránky virtualizačního systému KVM, dostupné na url adrese: *http://www.linux-kvm.org/*.

[6] Qemu – oficiální webové stránky virtualizačního systému Qemu, dostupné na url adrese: *http://www.qemu.org/*.

[7] Debian GNU/Linux – oficiální webové stránky operačního systému Debian GNU/Linux, dostupné na url adrese: *http://www.debian.org/*.

[8] libvirt – oficiální webové stránky virtualizačního API libvirt, dostupné na url adrese: *http://www.libvirt.org/*.

A Tabulka virtuálních strojů

Tato tabulka odpovídá výchozímu nastavení. V případě úpravy parametrů virtualizace tak již nemusí odpovídat skutečnosti.

Číslo	Název	Port s terminálem	Port s Monitorem
1	virtualni_stroj_1	50001	55001
2	virtualni_stroj_2	50002	55002
3	virtualni_stroj_3	50003	55003
4	virtualni_stroj_4	50004	55004
5	virtualni_stroj_5	50005	55005
6	virtualni_stroj_6	50006	55006
7	virtualni_stroj_7	50007	55007
8	virtualni_stroj_8	50008	55008
9	virtualni_stroj_9	50009	55009
10	virtualni_stroj_10	50010	55010
11	virtualni_stroj_11	50011	55011
12	virtualni_stroj_12	50012	55012
13	virtualni_stroj_13	50013	55013
14	virtualni_stroj_14	50014	55014
15	virtualni_stroj_15	50015	55015
16	virtualni_stroj_16	50016	55016
17	virtualni_stroj_17	50017	55017
18	virtualni_stroj_18	50018	55018
19	virtualni_stroj_19	50019	55019
20	virtualni_stroj_20	50020	55020
21	virtualni_stroj_21	50021	55021
22	virtualni_stroj_22	50022	55022
23	virtualni_stroj_23	50023	55023
24	virtualni_stroj_24	50024	55024
25	virtualni_stroj_25	50025	55025
26	virtualni_stroj_26	50026	55026
27	virtualni_stroj_27	50027	55027
28	virtualni_stroj_28	50028	55028
29	virtualni_stroj_29	50029	55029
30	virtualni_stroj_30	50030	55030