Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Návrh a integrace privátního cloudu do prostředí ZČU

Plzeň 2014 Václav Martinovský

Originál zadání práce

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 14. května 2014

Václav Martinovský

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce, Ing. Jiřímu Siterovi, za poskytnuté odborné rady, vlídný přístup a věnovaný čas. Dále děkuji zaměstnancům Centra informatizace a výpočetní techniky, Ing. Michalovi Švambergovi a Ing. Petru Grolmusovi, za pomoc při realizaci vybraných technických aspektů, Ing. Luboši Matějkovi z Katedry informatiky a výpočetní techniky za užitečné komentáře a v neposlední řadě také Bc. Miriam Sovové za podporu a připomínky vedoucí k vylepšení této práce.

Abstract

Design and integration of private cloud into the UWB environment

The aim of this work is to prepare a design for the implementation of a private cloud into the environment of University of West Bohemia. In order to achieve this objective is in the theoretical part an overview of current virtualization and cloud computing technologies. Follows an overview of software platforms that meet the requirements and it is selected one, that is used for implementation.

The practical part deals with the design and description of the architecture of chosen solution. On the basis of this concept is prepared and launched a test run to confirm the feasibility of the concept.

Keywords: virtualization, cloud computing, IaaS, OpenStack, KVM

Abstrakt

Návrh a integrace privátního cloudu do prostředí ZČU

Cílem této práce je připravit návrh implementace privátního cloudu do prostředí Západočeské univerzity v Plzni. Za účelem dosažení tohoto cíle je v teoretické části práce provedeno seznámení s technologií virtualizace a s oborem cloud computingu. Následně je zpracován přehled softwarových platforem, které vyhovují požadavkům a z nich je vybrána jedna, která je použita k realizaci.

Praktická část se zabývá návrhem a popisem architektury zvoleného řešení. Na základě tohoto návrhu je pak připraven a spuštěn pilotní provoz, který má za cíl potvrdit realizovatelnost konceptu.

Klíčová slova: virtualizace, cloud computing, IaaS, OpenStack, KVM

Obsah

1 Úvod

Cloud je pravděpodobně jedno z nejvíce skloňovaných slov dnešního světa informačních technologií. At' už chceme nebo nechceme, obklopuje naše okolí a stále více organizací, firem i jednotlivců využívá v nějaké podobě cloudových služeb.

Tato práce si klade za cíl připravit návrh infrastruktury privátního cloudu pro potřeby Západočeské univerzity v Plzni, která chce umožnit studentům i zaměstnancům spouštět virtuální stroje pro výukové a výzkumné účely.

Prostředkem k dosažení tohoto cíle je nejprve seznámení s historií a aktuálním stavem odvětví cloud computingu. Neméně důležitou částí je přehled virtualizačních technologií, na kterých je odvětví postaveno a bez kterého si lze těžko představit tak obrovský rozmach.

Samostatná kapitola je věnována porovnání softwaru, kterým lze realizovat privátní cloud. Z této kapitoly vzejde jeden produkt, který pak bude použit v návrhu i praktické realizaci. Nedílnou součástí je i pilotní provoz (Proof of Concept), kterým je prakticky otestována a předvedena realizovatelnost zvoleného řešení.

1.1 Specifikace požadavků

Tato diplomová práce je zpracovávána ve spolupráci s Centrem informatizace a výpočetní techniky (dále jen „CIV“ nebo „zadavatel“). Součástí je soupis požadavků od zadavatele, které by mělo výsledné řešení splnit.

Primárním cílem řešení je vytvoření nové služby, kterou lze popsat jako samoobslužné zřizování virtuálních serverů pro potřeby výuky a výzkumu. Jako typické použití očekáváme virtuální laboratoř, kde si uživatel (student) spustí krátkodobě (hodiny) několik virtuálních počítačů propojených sítí.

- Uzly budou plně v jeho správě a typicky nastartované z připraveného obrazu (například klient-server architektura pro laboratorní cvičení konfigurace či debugování síťového protokolu).
- Řešení musí splňovat provozní požadavky výpočetního prostředí Orion

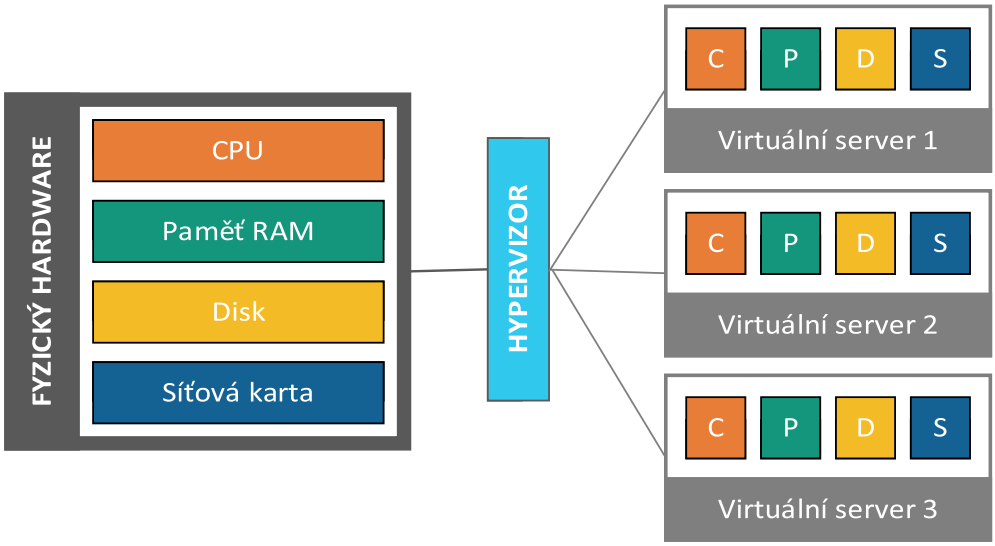
(monitoring, bezpečnost, zálohování) a být přiměřeně napojené do existující infrastruktury (zejména AAI[1], diskový systém, síť). Konkrétně je třeba napojení na bázi uživatelů a přístup všech uživatelů k nové službě.

- Realizované řešení musí umožňovat stanovení limitů na využití zdrojů a měření využívání těchto zdrojů.
- Součástí zadání není příprava obrazů virtuálních serverů, ale zvolené řešení musí podporovat práci s obrazy (import z jiných prostředí, podpora kontextualizace, podpora různých operačních systémů).
- Preferujeme otevřené řešení bez licenčních omezení a poplatků, modulární a založené na standardech. Důležitým parametrem je malá provozní náročnost a možnost rozšiřování (kapacita, vlastnosti).
- Výhodou je použití v současnosti užívaných řešení a technologií.

[1] Authentication and Authorisation Infrastructure

2 Virtualizace

Virtualizace představuje mezivrstvu mezi hostitelským systémem a hostovaným systémem. Fyzické zdroje, mezi které patří zejména paměť, procesor, disk, síťová karta jsou virtualizovány a dále poskytnuty hostovanému systému (obr. 2.1), na kterém běží samostatný operační systém (dále jen OS). Ten takto přidělené zdroje vnímá, jako by byly fyzické a vyhrazené. O správu se stará tzv. virtualizační manager, často také označovaný jako hypervizor.

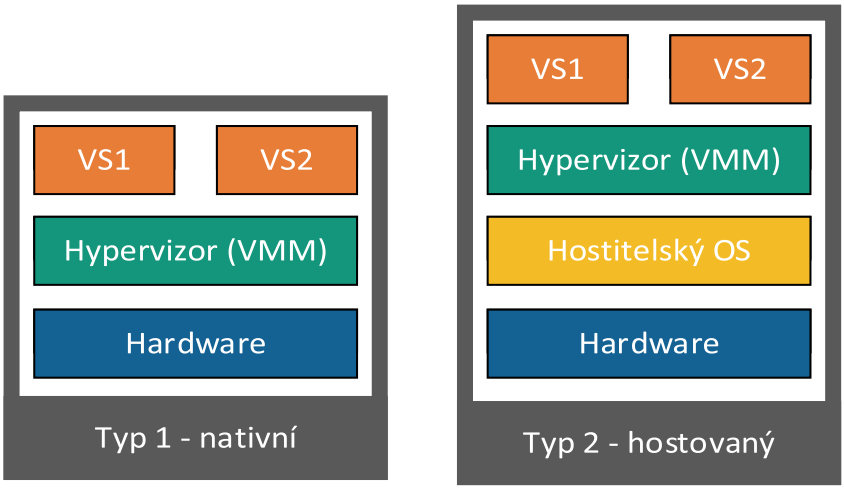


Obrázek 2.1: Znázornění virtualizace hardware

Zdroj: vlastní zpracování

V roce 1974 publikovali Popel a Goldberg článek, ve kterém rozdělili hypervizory na 2 základní typy, což ilustruje obr. 2.2: [42]

- **Typ 1** (nativní, bare-metal) – hypervizor je spuštěn přímo na hardware, kde tvoří mezivrstvu mezi hostovanými operačními systémy a fyzickými prostředky.
- **Typ 2** (hostovaný, hosted) – na hardware je spuštěn klasický operační systém a hypervizor je nainstalován jako součást tohoto OS. Oproti prvnímu typu je zde tedy navíc jedna vrstva.



Obrázek 2.2: Ilustrace hypervizorů typu 1 a 2

Zdroj: vlastní zpracování

Příkladem prvního typu je např. VMware ESX, Microsoft Hyper-V či Citrix XenServer. Do druhého typu řadíme VMware Workstation a VirtualBox.

Ne vždy je ale klasifikace typu jasná. Například KVM i XEN potřebují pro svůj běh operační systém (což by naznačovalo, že jsou typu 2), ale jsou považovány za hypervizory typu 1, protože jejich kód pracuje s hardwarem na stejné úrovni jako jádro (kernel).

2.1 Rozšíření, přínosy

Ačkoliv koncept virtualizace sahá do 60. let 20. století k IBM a jeho operačnímu systému CP-40, opravdový rozmach a širšího produkčního nasazení se virtualizace dočkala až v novém tisíciletí. Letošní rozsáhlá studie firmy Kapsch [47] s názvem *Trendy a výzvy ICT ve střední a východní Evropě* ukázala, že nějakou formu virtualizace využívá 58 % českých firem, přičemž z této části se jedná z 90 % o virtualizaci serverů a datových center.

Důvody rozšíření jsou zejména:

- **Konsolidace**, při které dochází ke slučování více fyzických serverů do virtuálního prostředí. Cílem je zejména úspora, protože zkušenosti z praxe ukazují, že v nevirtualizovaném prostředí jsou servery zatíženy pouze na zlomek své kapacity. Příkladem je Amazon, který naměřil pouze 10% průměrnou zátěž. [25] Konsolidace vede k lepšímu využití zdrojů a tedy úsporám jak při nákupu hardware, tak při provozu (servis, elektřina, chlazení, ...).

- **Zvýšení dostupnosti** pomocí High Availability řešení, která vytváří virtuální clustery s vyvažováním zátěže. Většina virtualizačních technologií umožňuje provádět živou migraci[1], díky čemuž lze bezproblémově stěhovat servery po dostupné fyzické architektuře (např. kvůli plánované údržbě hardware).

- **Izolace**, díky které lze provozovat na jednom fyzickém serveru i vzájemně nekompatibilní služby nebo služby vyžadující specifické prostředí, které by jinak ovlivnilo ostatní aplikace. Dále přispívá k vyšší bezpečnosti dat díky důslednému oddělení jednotlivých prostředí.

- **Testování** nových verzí aplikací nebo služeb lze realizovat velmi snadno díky možnosti naklonovat virtuální server a na něm „nanečisto“ vyzkoušet, jak se bude chovat po aktualizaci. Dalším scénářem může být situace, kdy potřebujeme otestovat aplikaci na velké škále různých operačních systémů, k čemuž bychom jinak potřebovali velké množství fyzických serverů.

- **Zálohování** lze velmi snadno provádět pomocí tzv. snapshotů (binárních obrazů souborového systému), díky kterým můžeme velmi rychle obnovit stav serveru k určitému okamžiku.

2.2 Přístupy k virtualizaci

Tato podkapitola představí základní virtualizační techniky od emulace až po virtualizaci na úrovni operačního systému. U každého přístupu bude proveden krátký popis a zmíněn příklad konkrétní technologie, který reprezentuje nasazení dané techniky v praxi.

[1] Živá migrace virtuálních serverů je základní vlastnost virtualizace, která umožňuje přesun spuštěného virtuálního serveru z jednoho fyzického serveru na druhý bez přerušení. [9]

2.2.1 Emulace

Emulace umožňuje provozovat virtuální systém jiné architektury, než je hostitelský systém (lze tak provozovat např. Windows na PowerPC[2]). Je však nutné veškeré operace interpretovat, což má velmi negativní vliv na výkon. I vzhledem k tomuto faktu je využívána spíše okrajově pro velmi specifické případy. Příkladem je emulátor QEMU[3].

2.2.2 Plná virtualizace

V případě plné virtualizace můžeme provozovat na hostiteli operační systémy bez nutnosti je jakkoliv upravovat. Pro běh je nutná hardwarová podpora procesoru, v případě Intelu se jedná o technologii Intel VT a u AMD jde o AMD-V. Ověření jestli procesor hardwarově podporuje virtualizaci provedeme (pod Linuxem) příkazem:

```
egrep '(vmx|svm)' /proc/cpuinfo
```

Pokud je virtualizace podporována, výstupem bude VMX (pro Intel) nebo SVM (pro AMD).

Do této skupiny řadíme většinu nejpoužívanějších systémů, zejména VMware, Hyper-V, KVM a také XEN (bližší popis v kap. 2.3).

2.2.3 Paravirtualizace

Při paravirtualizaci je hostovanému systému poskytnuto jen částečně virtualizované prostředí. Je nezbytně nutné, aby toto podporoval virtualizovaný OS. Výhodou je vyšší rychlost než u plné virtualizace (testovací benchmark[4] uvádí rozdíl cca. 3 %).

Typickým zástupcem je XEN[5], který umožňuje buď plnou virtualizaci nebo paravirtualizaci – proto je uveden v obou kategoriích.

[2]Detaily na: `http://www.microsoft.com/australia/office/mac/virtualpc7/`

[3]Více informací o emulátoru na: `http://www.qemu.org/`

[4]`http://shortrecipes.blogspot.cz/2009/03/xen-performance-of-full-virtualization.html`

[5]Více v samostatné kap. 2.3.2.

2.2.4 Virtualizace na úrovni OS

Technika je postavena na sdílení modifikovaného linuxového jádra mezi hostovaným a hostitelským systémem. Virtuální servery jsou označovány jako kontejnery a nedochází k virtualizaci hardware – není k dispozici vlastní síťová karta ani samostatná disková jednotka (souborový systém je sdílen s hostitelským systémem a ohraničen kvótami), procesy všech hostovaných systémů jsou vidět z hostitelského OS. Jednotlivé virtuální servery jsou ale od sebe odděleny a není možno se navzájem ovlivňovat.

Výhodou je velmi nízká režie a možnost serverům přidělit více zdrojů, než je fyzicky k dispozici. Nevýhodou je možnost takto provozovat pouze linuxové distribuce.

Mezi zástupce patří zejména OpenVZ[6] a Linux-VServer.

2.3 Virtualizační systémy

Tato kapitola se zabývá přehledem nejvýznamnějších virtualizačních systémů dneška – KVM, XEN, VMware a Hyper-V. Tyto systémy budou podrobněji představeny a v závěrečné podkapitole bude zvolen jeden systém, který bude použit pro praktickou část.

2.3.1 KVM

KVM (Kernel-based Virtual Machine) je open-source[7] virtualizační řešení pro platformu Linux na architektuře x86, které poskytuje plnou virtualizaci (pomocí modifikovaného emulátoru QEMU) a pracuje s rozšířeními Intel VT a AMD-V. Ke svému běhu tedy potřebuje kromě Linuxu také procesor s danou instrukční sadou. [30]

Přímo linuxové jádro zajišťuje službu hypervizora, který poskytuje virtualizovaný hardware spuštěným virtuálním serverům. Můžeme tak mluvit o hypervizoru typu 1 (bare-metal).

[6] Detaily na: `http://openvz.org/`

[7] Počítačový software s otevřeným zdrojovým kódem.

Není třeba modifikovat hostovaný operační systém a díky tomu lze spouštět různé operační systémy (Linux, Windows, ...), je podporována živá migrace virtuálních serverů. Původně byl vyvíjen firmou Qumranet, kterou následně koupil Red Hat[8] a na KVM postavil svůj produkt Red Hat Enterprise Virtualization[9]. Vznikl v roce 2006 a je standardní součástí linuxového jádra od roku 2007 (verze 2.6.20).

V současné době se jedná o vyspělou technologii a je často nasazovaným open-source hypervizorem. Díky svému pozdnímu uvedení, kdy již byly dostupné instrukční sady procesorů pro podporu virtualizace, je KVM koncepčně velmi prostý. Zatímco hypervizor VMware obsahuje přes 6 milionů řádků kódu a Xen okolo 500 tisíc, první stabilní verze KVM obsahovala jen něco málo přes 10 000 rádků kódu. [17]

Právě díky této historii a velmi rychlému zařazení do linuxového jádra se řada vývojářů domnívá, že KVM představuje budoucnost open-source virtualizace. Společnosti jako Red Hat a SuSE daly přednost KVM před XENem.

2.3.2 XEN

Původně vznikl jako projekt na univeritě v Cambridge a v roce 2003 byla vydána první veřejná verze. Dnes je vyvíjen jako open-source pod licencí GPL2. Během roku 2011, tedy po více než 8 letech, se s nástupem kernelu 3.0 dočkal zařazení do linuxového jádra.

Správu procesoru a paměti zajišťuje hypervizor, ostatní zařízení spravuje prigilegovaný virtuální systém, který označujeme jako Dom0. Podporuje živou migraci virtuálních serverů a umí pracovat ve dvou režimech:

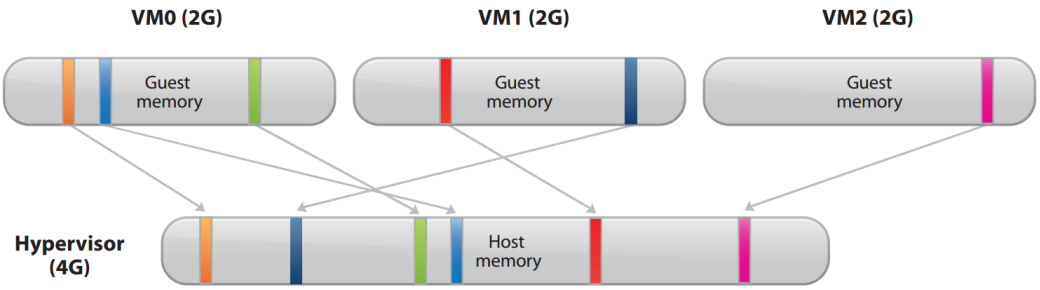
- **Paravirtualizace** (PV) – při paravirtualizaci není nutná podpora CPU, avšak virtualizovaný systém vyžaduje mít jádro a ovladače podporující PV. V praxi je jako hostovaný systém podporován každý Linux s jádrem 2.6.24 a novějším.
- **Plná virtualizace** (HVM) – dostupná od verze 3.0, vyžaduje podporu CPU, využívá QEMU (podobně jako KVM) na emulaci hardware (BIOS, IDE, VGA, USB, síťová karta, ...). Není vyžadována spolupráce hostitelského systému.

[8] Více informací o sloučení na: `http://www.redhat.com/promo/qumranet/`
[9] Detaily: `http://www.redhat.com/products/cloud-computing/virtualization/`

2.3.3 VMware

Společnost VMware se virtualizací zabývá již od roku 1998, přičemž první verze hypervizoru ESX byla vydána v roce 2001. Skládá se z vlastního jádra vmkernel a servisní konzole, přičemž ovladače ostatních zařízení, které jsou odvozeny z ovladačů pro OS Linux, jsou také součástí hypervizoru. Jelikož nepotřebuje k běhu jiný operační systém, označujeme ho jako typ 1. [41]

Velkou výhodou jsou pokročilé funkce s pamětí RAM[10], kdy je využíváno situace, že virtuální servery málokdy alokují veškerou přidělenou RAM. Hypervizor ESX umožňuje přidělit serverům v součtu více paměti, než je na serveru dostupné. Obr. 2.3 ilustruje, jak hypervizor do 4 GB fyzické paměti ukládá bloky paměti z virtuálních serverů.



Obrázek 2.3: Práce s pamětí u VMware ESX

Zdroj: [54]

Výše uvedené řešení je pak ještě vylepšeno technologií Transparent Page Sharing (TPS), která využívá faktu, že spuštěné virtuální servery mají často v paměti identické bloky (stejné aplikace, stejný OS, ...). Díky této technologii jsou eliminovány duplicity a ve fyzické paměti jsou uloženy identické bloky jen jednou. [54]

Jde o komerční software, jehož cena se pohybuje v řádech tisícovek EUR. Vzhledem ke své historii a technické vyspělosti je velmi silným hráčem na trhu virtualizace. VMware je populární především mezi takovými zákazníky, kterým záleží mnohem víc na značce, záruce a komerční podpoře než na ceně řešení. [17]

VMware dává k dispozici zdarma vSphere Hypervisor[11], který umožňuje

[10] Random-Access Memory

[11] Detaily zde: `http://www.vmware.com/cz/products/vsphere-hypervisor/`

provozovat virtualizaci v omezeném rozsahu (nefunkční živá migrace, omezení na počet jader CPU, omezení na velikost paměti RAM).

2.3.4 Hyper-V

Hyper-V je virtualizační technologie Microsoftu s hypervizorem typu 1, která umožňuje plnou virtualizaci a paravirtualizaci. První verze byla vydána v roce 2008 (což z ní činí nejmladší srovnávanou technologii) a je k dispozici buď jako součást produktu Windows Server (2008 i 2012) nebo jako samostatný OS Hyper-V Server (2008 i 2012).

Jak je patrné, jde o komerční produkt a běžná licence Windows Server 2012 R2 Datacenter stojí 6155 USD[12] (cena platná v dubnu 2014). Samotný Hyper-V Server je dostupný zdarma, avšak bez jakýchkoliv nástrojů pro správu a bez uživatelského rozhraní – je možná pouze vzdálená správa, avšak nástroje pro správu od Microsoftu jsou placené.

2.4 Porovnání a výběr

IBM publikovala studii o virtualizaci [27], ve které zjišťuje náklady TCO[13] na 3 roky provozu u enterprise řešení virtualizací. Bylo srovnáváno KVM, VMware a Microsoft Hyper-V. Výsledkem je, že ačkoliv byla u KVM zahrnuta cena licencí pro Red Hat Enterprise Linux a IBM Systems Director (který by se dal v našem kontextu považovat za zbytný), stále vychází v celkovém TCO levněji než srovnatelná konkurence. Dá se očekávat, že náklady pro XEN budou na podobné úrovni jako pro KVM.

Zadavatel ve svých požadavcích specifikoval přání, aby zvolené řešení nebylo zatíženo licenčními poplatky a omezeními. Pokud ho budeme respektovat, je nutné ze srovnání vyřadit VMware i Hyper-V (které sice mají také verze zdarma, ale jsou omezené). I kdyby tento požadavek neexistoval, na základě výše uvedeného porovnání vyplývá, že by tyto technologie nebyly ekonomicky výhodné.

[12] `http://www.microsoft.com/en-us/server-cloud/products/windows-server-2012-r2/buy.aspx`

[13] Total Cost of Ownership

	KVM (IBM + Red Hat)	VMware	Microsoft
Systems Management	**IBM Systems Director** ▪ $12,600 License + SnS for 3 yrs.	**VMware vCenter** ▪ $8,742 License + SnS for 3 yrs.	**Windows 2008 Server Data Center** ▪ $93,200 License + SnS for 3 yrs.
Virtualization Management	**IBM Systems Director VMControl:** ▪ $39,200 License + SnS for 3 yrs.	**ESXi via VMware vSphere** ▪ $201,280 License + SnS for 3 yrs.	
Hypervisor	**Red Hat Enterprise Linux** ▪ $185,200 Subscription/Support for 3 yrs.		**Microsoft Windows 2008 Server Standard (1 VM with 5 CALs)** ▪ $34,820 License + SnS for 3 yrs.
Operating System		**Red Hat Enterprise Linux** ▪ $185,200 per 2-socket server Subscription/Support for 3 yrs.	**Red Hat Enterprise Linux** ▪ $185,200 per 2-socket server Subscription/Support for 3 yrs.
Hardware / Server **	**IBM x3550 M3 server:** ▪ $143,680 for 20 servers	**IBM x3550 M3 server:** ▪ $143,680 for 20 servers	**IBM x3550 M3 server:** ▪ $143,680 for 20 servers
3-year TCO	**$380,680**	**$538,902** ***(+42% more than KVM)***	**$456,900** ***(+20% more than KVM)***

** Note: all prices shown are list prices via publically available information (websites, whitepapers)*
*** Server Configuration (Quantity 20): IBM x3550 M3, 2x6C, 2x500GB HDD, 48GB RAM, 2x 6Gb HBA, ServerRAID*

Obrázek 2.4: Srovnání 3letého TCO u zvolených virtualizačních technologií

Zdroj: [27]

Zbývá se tedy rozhodnout mezi KVM a XEN. V dnešní době jsou obě technologie na velmi podobné technické úrovni, avšak pro KVM hovoří fakt, že je již 7 let standardní součástí linuxového jádra a to mu zaručuje širokou podporu napříč různými distribucemi. Komunity firem i uživatelů také dávají stále častěji přednost KVM před XENem. [17]

Autor této práce doporučuje zvolit KVM jako virtualizační technologii pro použití v praktické části.

3 Cloud computing

V této kapitole bude představena historie a definovány základní pojmy z oblasti cloud computingu. Budou zde také rozebrány modely nasazení cloudu a popsány služby, které může poskytovat. Samostatná podkapitola je pak věnována výhodám, obavám a překážkám při využívání cloudu. Závěrečná část podává přehled softwarových platforem pro řízení cloudu, je provedeno jejich zhodnocení a zvoleno řešení k implementaci v praktické části.

3.1 Historie

Myšlenka sdílení výpočetních služeb přes počítačovou síť sahá až do 60. let 20. století, kdy byly výpočetní služby poskytovány na sálových počítačích (mainframech) vybrané skupině uživatelů, kteří se dělili o výpočetní čas.

Profesor John McCarthy v roce 1961 vyslovil na své přednášce názor, že (počítačové) „výpočty mohou být v budoucnu poskytovány jako veřejná služba stejně jako telefon“ [20] a také prvně použil pojem *utility computing*.

V roce 1966 publikoval Douglas F. Parkhill knihu *The Challenge of the Computer Utility*, ve které poskytování výpočetního výkonu přes síť přirovnává k poskytování elektrické energie. Domácnosti a firmy potřebují elektrickou energii, avšak téměř nikdo si kvůli této potřebě nebude pořizovat vlastní elektrárnu. V této době však šlo vesměs o vizionářské myšlenky, které nebylo možné vzhledem ke stavu infrastruktury realizovat. [40]

Po vypuknutí *internetové horečky*[1] na přelomu tisíciletí sehrála klíčovou roli ve vývoji společnost Amazon, která provedla zásadní modernizaci svých datových center. Zjistila totiž, že po většinu času bylo využíváno cca. jen 10 % instalované výpočetní kapacity a zbytek zůstával nevyužit pro pokrytí špiček. [25] Vyvinula a v roce 2006 oficiálně spustila platformu Amazon Web Services (AWS), která poskytuje výpočetní výkon široké veřejnosti. [1]

Cloud začíná nabízet stále více firem, v roce 2011 oznamuje IBM spuštění

[1] Označení pro období hromadného rozkvětu internetových firem, které neměly promyšlený obchodní model a brzy zkrachovaly. Toto období probíhalo přibližně v letech 1996 až 2001.

IBM SmartCloud[2] a o rok později Oracle spouští svůj Oracle Cloud[3]. V posledních letech roste odvětví tempem přes 10 % p.a. [18]

3.2 Definice a charakteristika

Pojem *cloud computing* (dále jen CC) poprvé definoval v roce 1997 prof. Chellappa jako „výpočetní paradigma, jehož hranice budou určeny spíše ekonomickými, než technickými limity.“ [8]

Inženýři z Kalifornské univerzity v Berkeley přišli v roce 2009 s mnohem obsáhlejší definicí. „Termín *Cloud Computing* odkazuje jak na aplikace poskytované jako služby přes Internet, tak na hardware a systémový software v datacentrech, které tyto služby poskytují. Samotné služby jsou označovány jako *Software jako služba* (Software as a Service; SaaS), termínem *Cloud* označujeme samotný hardware a software v datacentrech. Pokud je *Cloud* nabízen v režimu pay-as-you-go široké veřejnosti, nazýváme ho *Veřejným cloudem* (Public cloud), přičemž samotnou pronajímanou službu označujeme jako *Utility Computing*. Termín *Privátní cloud* (Private cloud) používáme pro datacentra firem či organizací, které nejsou nabízena široké veřejnosti. *Cloud Computing* se dá vyjádřit jako součet SaaS a Utility Computing.“ [2]

Zatímco definice *utility computingu* neobsahovala žádnou zmínku o úplatném provozu, téměř všechny definice CC již zahrnují model plateb *pay-as-you-go* (či *pay-per-use*). Tedy model, kdy uživatel platí pouze za skutečně spotřebované prostředky (výpočetní výkon, objem uložených dat, ...). Finanční aspekt zdůrazňuje Reese [44], který uvádí tyto parametry:

- Služba je přístupná přes webový prohlížeč nebo webové API[4].
- Pro spuštění služby není nutné mít počáteční kapitál.
- Platí se pouze za spotřebované zdroje.

Americký NIST (National Institute of Standards and Technology), zavedl definici [32] cloud computingu, která je uznávána vládou USA:

[2]Detaily na: `http://www.ibm.com/cloud-computing/us/en/`
[3]Více na: `https://cloud.oracle.com`
[4]Application Programming Interface

„Cloud computing je model, který umožňuje výhodný síťový přístup ke sdíleným výpočetním prostředkům (např. síť, servery, datové sklady, aplikace a služby) a který může být rychle zpřístupněn s minimální spoluprací poskytovatele služby.“

Pokud se podíváme na to, jak termín definují v České republice, narazíme na definici autorů Burkoně a Šebesty z Katedry informačních technologií na VŠE:

„Cloud computing zahrnuje dodávku virtualizovaných IT zdrojů jako službu přes Internet. Služby CC jsou dodávány škálovatelným a bezpečným způsobem ze vzdálených datových center a jsou zpoplatňovány modelem skutečné spotřeby *pay-as-you-use*. Dělíme je na služby infrastruktury (IaaS), služby platformy (PaaS) a služby softwaru (SaaS).“ [16]

Tato definice obdobně jako předchozí zmiňuje zpoplatnění podle skutečné spotřeby a přístup přes Internet. Navíc ale zdůrazňuje virtualizaci zdrojů, což je společným znakem a podstatou cloudových služeb.

Jak je patrné, nepanuje obecná shoda ohledně přesné definice CC a často jsou také velmi vágní - např. definice NIST uvádí „(ekonomicky) výhodný přístup“, což si lze vykládat různými způsoby. Pro účely této práce se rozhodl autor považovat za CC takovou službu, která splňuje tyto body:

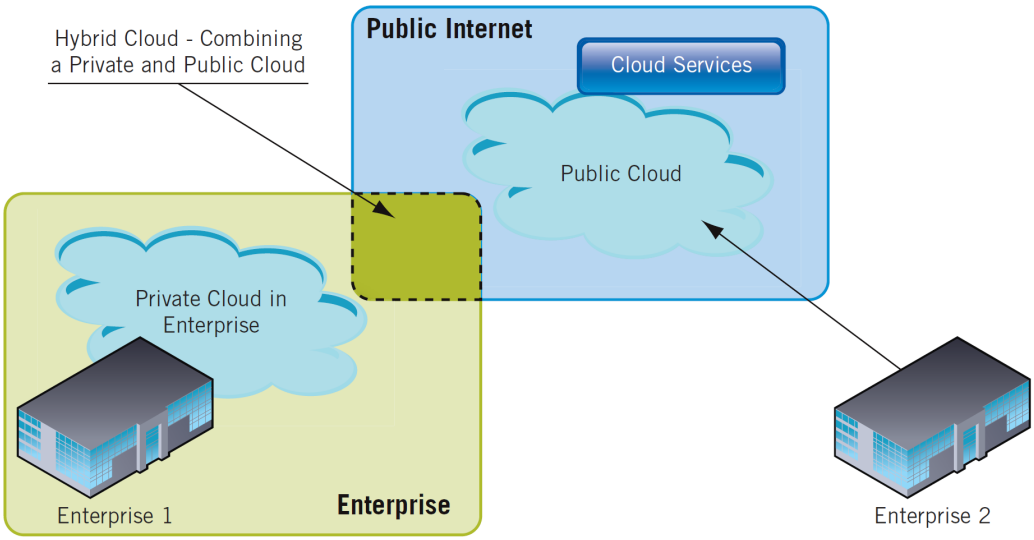
- Poskytuje virtualizované IT zdroje přes počítačovou síť.
- Je zpoplatněna za skutečné využití zdrojů.
- Přidělené zdroje lze snadno měnit (navyšovat či snižovat).
- Zřízení je plně automatické.

3.3 Dělení podle modelu nasazení

Cloudy můžeme rozdělit podle toho, kdo ho vlastní a spravuje. Toto rozdělení vychází z definice dle NIST [32] a ilustruje ho obr. 3.1:

- Public (veřejný) cloud
- Private (privátní) cloud

- Hybrid (hybridní) cloud
- Community (komunitní) cloud



Obrázek 3.1: Rozdělení cloudů podle modelu nasazení

Zdroj: [12]

3.3.1 Public cloud

Patrně nejpoužívanější variantu nasazení představuje veřejný, někdy také označovaný jako externí cloud. Koncoví uživatelé (organizace) přistupují přes Internet ke vzdálenému poskytovateli, který poskytuje cloudové služby. Obvykle jde o sdílený princip, kdy je služba poskytována většímu množství uživatelů (multi-tenancy) a zpoplatněna podle skutečného využití zdrojů.

Tento model je široce akceptován a adaptován řadou firem. Velcí hráči jako Amazon, Microsoft[5] nebo Google[6] disponují silnou infrastrukturou s mnoha datovými centry, čímž umožňují uživatelům flexibilně škálovat zdroje s nízkými náklady. [19]

Nelze určit, na jakém hardware a odkud je služba poskytována, což je zejména pro firmy a organizace často zásadní problém zejména v návaznosti

[5] `http://www.microsoft.com/`
[6] `http://www.google.com/`

na legislativu ČR a EU. Například Zákon o ochraně osobních údajů poměrně striktně specifikuje podmínky, pod kterými je možné osobní údaje zpracovávat mimo území ČR. Některé z nich, například nutnost ohlásit Úřadu pro ochranu osobních údajů všechna místa, kde může docházet ke zpracování (ukládání) citlivých údajů, mohou představovat poměrně zásadní problém v případě využití velkých cloudů rozmístěných po celém světě.

3.3.2 Private cloud

Tímto termínem je označován takový cloud, který je přístupný pouze danému uživateli (organizaci). Může být fyzicky umístěn v prostorách uživatele (in-house) nebo externě (hosted), spravován interně nebo externě. Uživatel je k němu obvykle připojen dostatečně rychlou a nesdílenou linkou.

Existuje názor, že nejde o cloud jako takový [19], protože nedochází ke sdílení fyzické infrastruktury mezi více uživateli a tím je snížena efektivita, které dosahuje veřejný cloud. Vzhledem k tomu, že je k dispozici pouze předem objednaná kapacita, odpadá také výhoda snadné škálovatelnosti ve srovnání s veřejným cloudem.

Naopak díky faktu že nedochází ke sdílení s jinými uživateli stoupá bezpečnost řešení. Toto řešení je užitečné především pro citlivé vnitrofiremní aplikace, kde by bylo narušení bezpečnosti nebo stability (např. sousedním uživatelem) velký problém.

Lze přesně určit, na kterém konkrétním hardware je služba poskytována a odkud je poskytována. Významově má tedy velmi blízko ke klasickému server hostingu, od kterého se ale odlišuje použitím virtualizace.

Je vhodný zejména pro velké organizace, které si mohou dovolit velké investice do IT a také tam, kde je nutné udržet naprostou kontrolu nad všemi prvky infrastruktury. Pro některé případy užití (intentivní výpočty, ...) se může ukázat, že je pořízení privátního cloudu vyjde v uvažovaném horizontu levněji než využití veřejného cloudu.

	Public cloud	**Private cloud**
Počáteční náklady	žádné	vysoké
Účtování za	využité zdroje	dostupnou kapacitu
Sdílení infrastruktury	ano	ne
Flexibilita škálovatelnosti	vysoká	nízká

Tabulka 3.1: Shrnutí rozdílů mezi public a private cloudem

Zdroj: vlastní zpracování

3.3.3 Hybrid cloud

Hybridní cloud představuje spojení veřejného a privátního cloudu. Uživatel využitím tohoto modelu spojuje výhody obou řešení. Citlivé vnitrofiremní aplikace s kritickými daty budou provozovány na privátním cloudu, aby nad nimi organizace udržela plnou kontrolu a méně citlivé části aplikace přesune na veřejný cloud, kde využije zejména výhody nižší ceny a lepší celosvětové dostupnosti. [53]

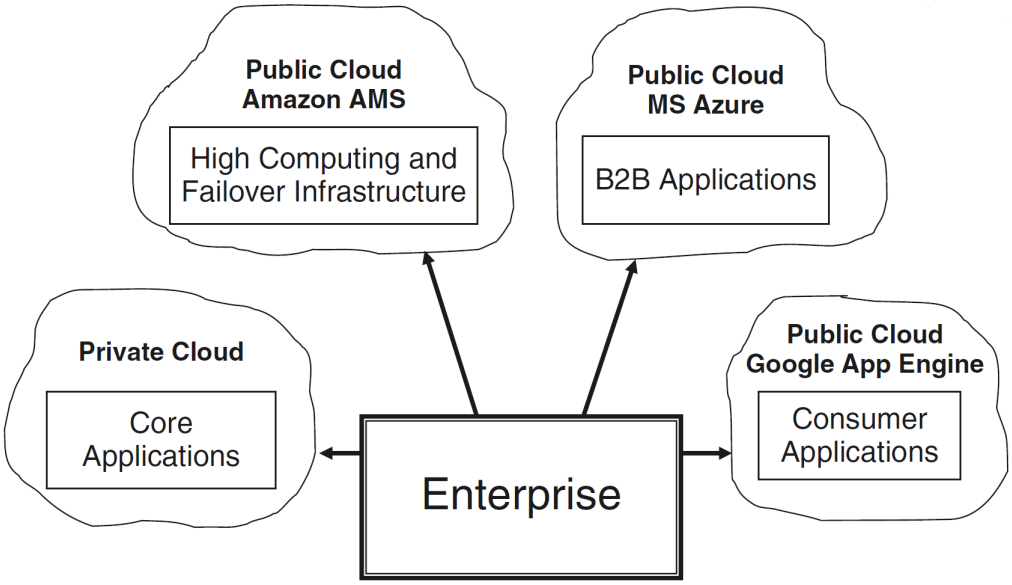
Dalším případem užití je situace, kdy běžný provoz je odbaven privátním cloudem a veřejný je využit pro pokrytí špiček, kvůli kterým by bylo nutné navyšovat kapacitu privátního cloudu.

Obrázek 3.2 ilustruje příklad hybridního cloudu, kde podnik využívá jeden privátní cloud a tři veřejné, přičemž každý cloud obsluhuje jinou část aplikací.

3.3.4 Community cloud

O komunitním cloudu lze uvažovat jako o mezistupni mezi veřejným a privátním cloudem. Je určen k výhradnímu užívání předem dané skupině organizací, které mají podobné zájmy a požadavky (např. v otázce zabezpečení). Často jde o firmy nějakým způsobem propojené či spřízněné.

Může být spravován externě, jednou z firem v komunitě nebo společnými silami. Výhodou je snížení nákladů oproti privátnímu cloudu.



Obrázek 3.2: Rozdělení cloudů podle modelu nasazení

Zdroj: [19]

3.4 Dělení podle typu služby

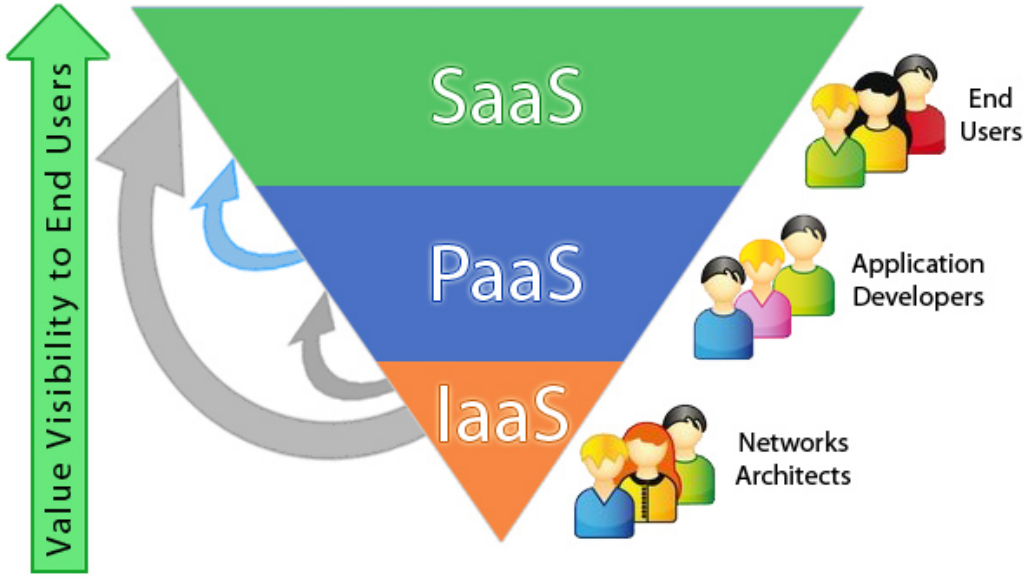
Na CC můžeme nahlížet jako na kolekci služeb, kterou lze znázornit vrstvenou architekturou, jak ilustruje obrázek 3.3.

3.4.1 Infrastructure as a Service

Pojmem IaaS[7] označujeme pronájem výpočetních prostředků, tedy garantovaného výkonu procesoru, operační paměti, diskového prostoru a připojení k internetu. Zákazníkovi je k dispozici virtualizovaná IT infrastruktura bez nutnosti zakupovat vlastní hardware. Výhodou je snadná škálovatelnost a platba pouze za využité zdroje.

V praxi si lze pod tímto pojmem představit například veřejnou službu Amazon EC2[8], Windows Azure[9] či Google Compute Engine[10]. Kapitola 3.7

[7] Infrastructure as a Service
[8] `https://aws.amazon.com/ec2/`
[9] `http://azure.microsoft.com/`
[10] `https://cloud.google.com/products/compute-engine/`



Obrázek 3.3: Vrstvy cloud computingu

Zdroj: [43]

se podrobněji zabývá softwarem pro realizaci IaaS v rámci privátního cloudu.

3.4.2 Platform as a Service

PaaS[11] je souhrn vývojových nástrojů, databází a výpočetních prostředků speciálně připravených pro vývoj a běh cloudových aplikací. Lze si ho představit jako IaaS + specifická softwarová platforma a primárně je určen pro softwarové vývojáře.

Příkladem jsou služby jako Heroku, Salesforce či Google App Engine.

3.4.3 Software as a Service

Nad ostatními vrstvami (viz obr. 3.3) se nachází SaaS[12]. Tento termín byl používán ještě před nástupem CC a označuje pronájem aplikací, které jsou

[11] Platform as a Service
[12] Software as a Service

poskytovány vzdáleně uživateli. Pro jejich využívání není nutné, aby měl uživatel nainstalován nějaký specifický software, obvykle si vystačí s běžným internetovým prohlížečem. Jako příklad lze uvést webové rozhraní pro práci s poštou (WebMail), kdy není nutné instalovat samostatný poštovní program a je možné pracovat s e-mailovou schránkou. Dále lze zmínit různé CRM[13] systémy či kancelářské balíky (Google Docs, Office 365). S nástupem CC se jako základ pro SaaS jeví virtuální infrastruktura a nižší vrstvy cloudu. [53]

3.5 Výhody

V této kapitole rozebereme výhody plynoucí ze zavedení cloud computingu.

Finanční úspory

Jak již bylo uváděno na příkladu firmy Amazon [25], v případě konvenčních datových center existuje nezanedbatelné množství nevyužitého výkonu. Nasazením CC a virtualizace se zvyšuje efektivita využití zdrojů.

Škálovatelnost

CC poskytuje velkou flexibilitu, kdy je možné snadno upravovat množství dostupných zdrojů s žádným nebo minimálním výpadkem služby. Díky modelu plateb za využité zdroje také odpadají fixní náklady a nutnost plánovat výkon na dlouhou dobu dopředu.

Snížení rizika

Použitím CC lze také přenést část rizika od zákazníka směrem k poskytovateli. Nejde jen o podcenění plánování, kdy v případě tradičních serverů není možné flexibilně navýšit kapacitu a tím pádem riskovat nedostupnost či výpadky služby.

Je tu také bezpečnostní riziko, kdy lze očekávat, že velcí poskytovatelé

[13] Customer relationship management

cloudu mají technologie zabezpečeny výrazně lépe než běžná menší firma a podstupují také pravidelně bezpečnostní audity. Nemusí to samozřejmě platit ve všech případech.

Snížení zátěže

Outsourcováním IT služeb je snížena zátěž zaměstnanců firmy, kteří se pak mohou specializovat na hlavní předmět svého podnikání.

3.6 Obavy, překážky

Nasazení CC sebou přináší také obavy a někdy také překážky, se kterými je třeba počítat a minimalizovat je.

Bezpečnost a riziko

Pravděpodobně největší obavu představuje fakt, že citlivá data jsou v rukou poskytovatele služby a to často v prostředí, které je sdíleno s několika (někdy i tisíci) dalšími klienty. Je třeba zvážit, jestli je zvolený poskytovatel dostatečně důvěryhodný, aby úmyslně či neúmyslně nezneužil svěřená data. [46]

Závislost a dostupnost

Uživatel je plně závislý na vybraném poskytovateli a v případě problémů na jeho straně se to projeví i v dostupnosti a kvalitě využívaných služeb. Ačkoliv se cloudové služby obecně považují za velmi stabilní, již existují případy, kdy i velcí poskytovatelé měli závažné problémy s provozem.

Zákonné požadavky

V kapitole 3.3.1 jsme podrobněji rozebrali situaci, kdy je ze zákona vyžadována určitá možnost kontroly nad místem, kde jsou data uložena. To v případě velkých geograficky rozdělených cloudů představuje problém.

3.7 Softwarové platformy pro IaaS

V této podkapitole postupně rozebereme několik platforem pro poskytování IaaS. Klíčem k jejich zařazení do tohoto přehledu byla v první řadě podpora KVM, protože vycházíme z kapitoly 2.4, kde jsme zvolili KVM jako virtualizační infrastrukturu pro náš návrh.

3.7.1 Apache CloudStack

Vznikl v roce 2008, tehdy ještě pod firmou VMOps (později přejmenovanou na Cloud.com). V roce 2010 byla většina kódu uvolněna pod licencí GPLv3. O něco později došlo k odkoupení firmou Citrix, která uvolnila zbytek kódu pod stejnou licencí. V dubnu 2012 pak celý balík změnil licenci na ASLv2 a byl věnován do správy Apache, kde je dodnes. [10]

Tento krok měl za cíl větší rozšíření a zrychlení vývoje, na kterém se Citrix nadále podílí a na základech CloudStacku postavil svůj produkt Citrix CloudPlatform.

CloudStack podporuje hypervizory VMware, KVM, XenServer, Xen Cloud Platform (XCP) a Hyper-V. Skládá se ze dvou hlavních částí:

- **Management Server**, který se stará o správu zdrojů (úložiště, IP adresy, hostitelské servery, ...).
- **Computing Server**, který představuje výpočetní část cloudové infrastruktury a který je řízen management serverem.

3.7.2 Eucalyptus

Eucalyptus je open-source platforma, která si klade za cíl slučovat existující virtualizovanou infrastrukturu k vytváření cloudových zdrojů pro poskytování služeb přes počítačovou síť. [15]

Byl založen na University of California, Santa Barbara a v roce 2009 se transformoval do firmy Eucalyptus Systems. V roce 2012 došlo k podpisu dohody s Amazon Web Services, která umožňuje přesouvat virtuální servery

mezi privátním cloudem Eucalyptus a veřejným cloudem Amazon EC2, což usnadňuje vytvoření, správu a využívání hybridního cloudu. [24]

Je kompatibilní s Amazon Web Services (AWS) a Amazon S3, jeho API je kompatibilní s Amazon EC2 a umožňuje využívat hypervizory VMware, XEN a KVM.

Skládá se z pěti hlavních komponent, které zachycuje obr. 3.4: [14]

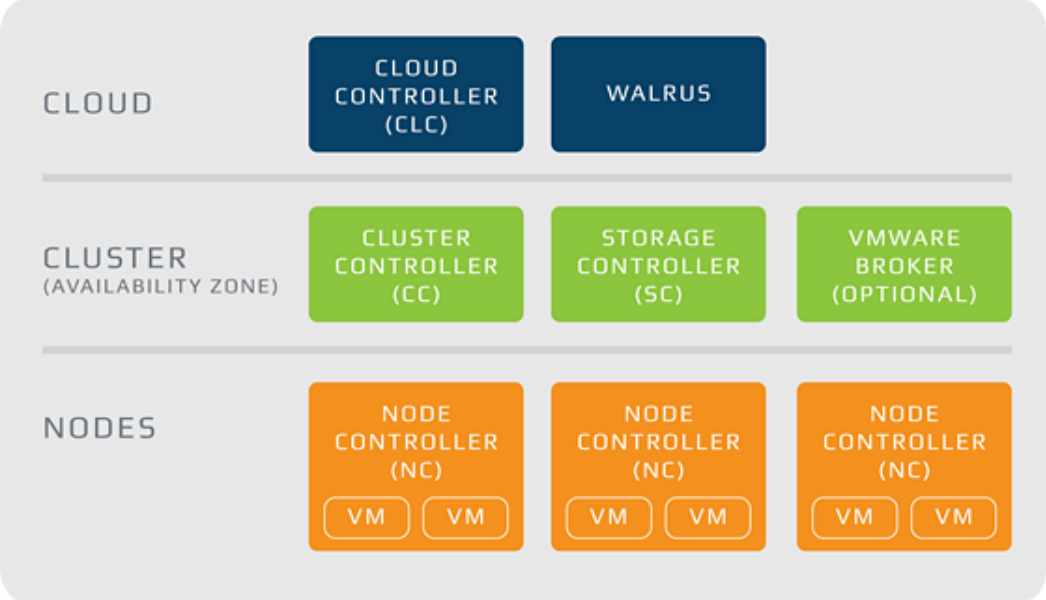
- Cloud Controller – zajišťuje rozhraní pro komunikaci ve formě API i webového portálu, plánování a účtování zdrojů.
- Cluster Controller – komunikuje se Storage a Node controllerem, spravuje instance virtuálních strojů a stará se o dodržování SLA.
- Node Controller – hostuje virtuální stroje a zajišťuje jejich vytváření, spravuje síťové rozhraní.
- Storage Controller – spravuje bloková zařízení a snapshoty v rámci cloudu.
- Walrus – ekvivalent k Amazon S3, zajišťuje perzistentní úložiště.

Až do roku 2011 Eucalyptus poháněl Ubuntu Enterprise Cloud (UEC), kdy byl nahrazen OpenStackem.

3.7.3 OpenNebula

OpenNebula je nástroj, který pomáhá virtualizovaným datovým centrům dohlížet na privátní, veřejné a hybridní cloudy. Není závislý na konkrétní technologii, platformě nebo API - umožňuje používat hypervizory KVM, XEN nebo VMware. [24] Obr. 3.5 zobrazuje jeho architekturu.

M. Llorente a Rubén S. Montero započali jeho vývoj v roce 2005 na University of Madrid, nyní je projekt spravován jako open-source. První veřejná verze byla uvolněna v roce 2008 a jde tak o nejstarší open-source software na správu cloudu. Podobně jako CloudStack je tvořen dvěma částmi – řídícím serverem (frontend) a výpočetními uzly.



Obrázek 3.4: Architektura Eucalyptu

Zdroj: [14]

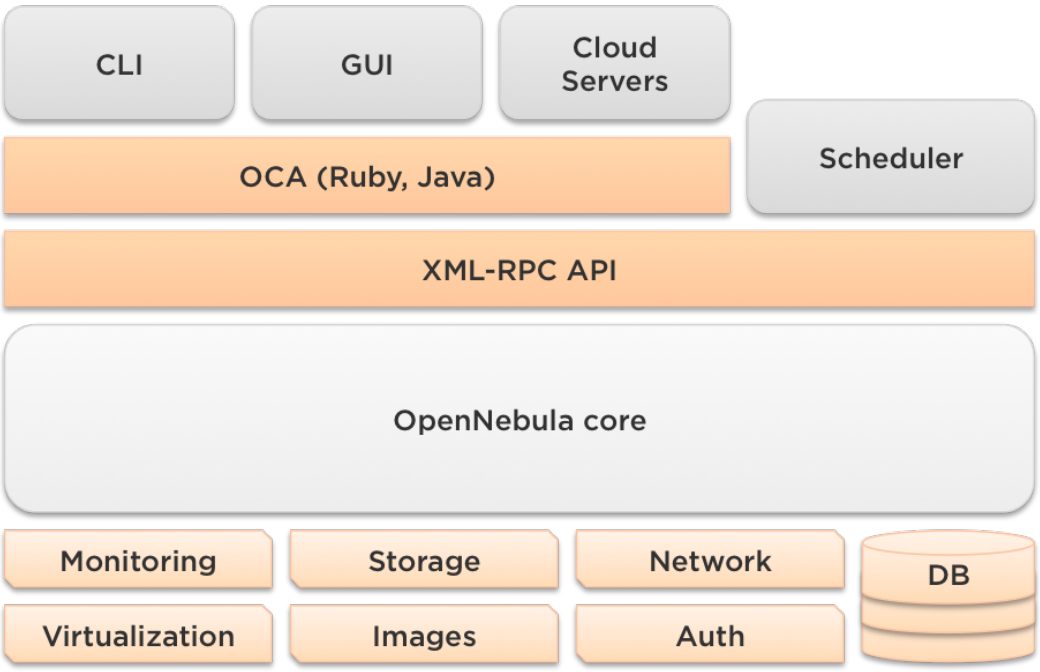
Podporuje sdílené (NFS spuštěné na SAN/NAS serveru) i nesdílené (lokální) úložiště, stejně jako distribuované souborové systémy jako GlusterFS[14] a MooseFS[15]. Umožňuje realizovat podobně jako OpenStack virtuální sítě s fixními a plovoucími IP adresami. Nabízí integrovaný firewall a podporu Open vSwitch. Jeho API je kompabitilní s Amazonem.

OpenNebula cílí především na uživatele vCloudu od VMware, kteří nyní tvoří až 70 % uživatelů. Verze 4.0 se pyšní prohlášením, že nabízí okamžitou náhradu vCloudu, která umožňuje kromě snížení nákladů také podporu jiných hypervizorů. [34]

3.7.4 OpenStack

Ze zpracovaného přehledu se jedná o nejmladší platformu, kterou v roce 2010 společně založili Rackspace a NASA. V současné době ji spravuje OpenStack Foundation a je pod licencí Apache. Klade si za cíl poskytnout řešení pro

[14]Detaily na: `http://www.gluster.org/`
[15]Více informací na: `http://www.moosefs.org/`



Obrázek 3.5: Architektura OpenNebuly

Zdroj: [36]

všechny typy cloudů díky snadné implementaci, vysoké škálovatelnosti a velkému množství funkcí. [39]

Aktuálně se na vývoji podílí přes 9000 jednotlivců a 200 firem jako je Cisco, Dell, HP, IBM, Intel, Red Hat, SuSE, VMware či Oracle a jejich počet stále narůstá. Ve srovnání s ostatními zmíněnými systémy pro IaaS čítá OpenStack největší komunitu přispěvatelů, vývojářů a uživatelů. [28]

Skládá se z několika samostatných modulů, přičemž každý z nich plní jinou funkci a společně propojené přes otevřené API tvoří komplexní řešení. Umožňuje se napojit a řídit řadu hardwarových prvků, od switche až po disková úložiště, přičemž pokud takové prvky nemáme k dispozici nebo je nechceme použít, lze vytvořit enterprise řešení pouze pomocí OpenStacku, který podporuje SDN (Open vSwich s OpenFlow) i softwarový storage (iSCSI).

Komponenty tvořící jádro (ve verzi Havana):

- **Horizon** (dashboard, webový portál)

- **Nova** (výpočetní uzel)
- **Neutron** (síťový uzel)
- **Swift** (objektové úložiště)
- **Cinder** (blokové úložiště)
- **Keystone** (správce identit)
- **Glance** (správce obrazů serverů)
- **Celiometer** (vyúčtování zdrojů)
- **Heat** (automatizovaná příprava instancí)

Ačkoliv je KVM výchozím hypervizorem pro OpenStack [45], je jich podporováno mnohem více. Podpora je rozdělena do několika úrovní: [38]

- **Skupina A**: plně podporované, prošlo unit testy a testováním funkčnosti, případné chyby jsou prioritně opravovány. Patří sem libvirt (qemu/KVM na architektuře x86).
- **Skupina B**: podporované, prošlo unit testy a testováním funkčnosti, chyby jsou hlášeny autorům a není záruka jejich oprav. Patří sem Hyper-V, VMware a XenServer.
- **Skupina C**: zastaralé, budou brzy vyřazeny. Patří sem baremetal, docker, Xen přes libvirt a LXC přes libvirt.

Začátkem roku 2014 se objevila oficiální podpora OpenStacku ze strany VMware[16], kdy byla provedena plná integrace do jejich infrastruktury. Díky této oficiální podpoře a před připravenému obrazu pro nasazení lze velmi snadno ovládat VMware cluster i přes OpenStack.

Platforma se neustále rozvíjí s cílem nabídnout výhledově i PaaS služby. Od verze Icehouse bude dostupná nová komponenta Trove, která poskytne uživatelům databázové služby (relační i nerelační databáze) bez nutnosti instalovat a spravovat vlastní databázový server. V přípravě je také komponenta Sahara zajišťující jednoduchou přípravu Hadoop[17] clusteru.

[16] Více informaci na: `http://www.vmware.com/files/pdf/techpaper/VMWare-Getting-Started-with-OpenStack-and-vSphere.pdf`

[17] `http://hadoop.apache.org/`

Další vyvíjenou komponentou je Ironic, která poskytne automatizaci správy serverů bez virtualizace (bare metal), čímž bude možné začlenit pod OpenStack i nevirtualizované servery.

3.7.5 Zhodnocení a výběr

Ze specifikace požadavků je patrné, že zvolené řešení musí být otevřené a modulární řešení bez licenčních omezení, jehož provoz lze automatizovat, snadno škálovat a případně upravit pro začlenění do infrastruktury Západočeské univerzity v Plzni.

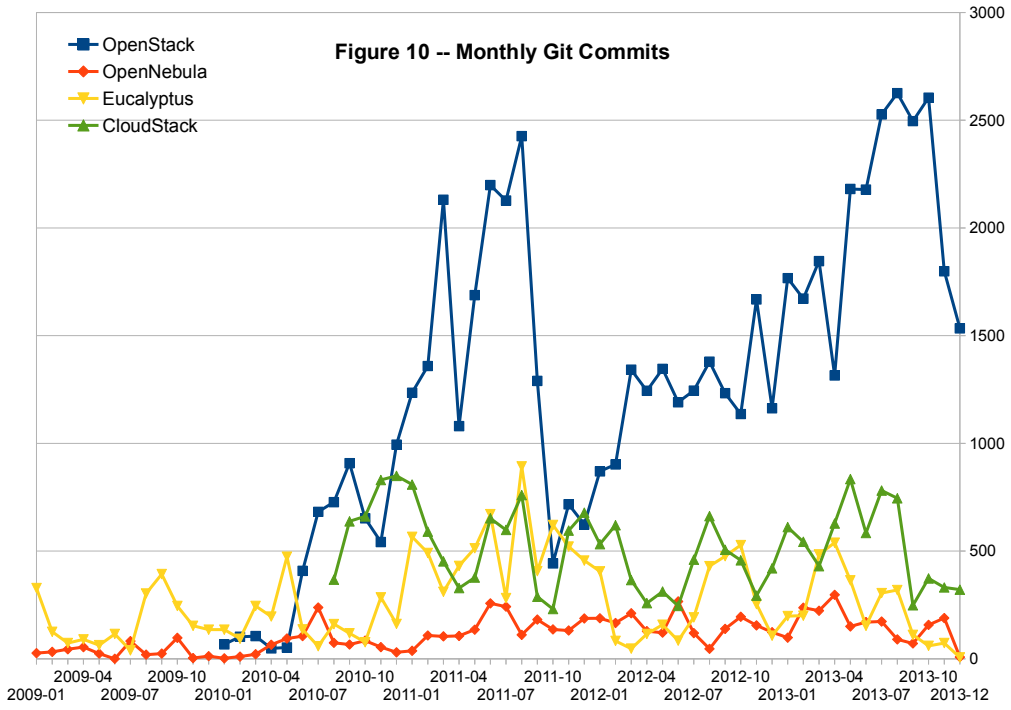
Jelikož v současné době na ZČU neexistuje žádný management cloudu, nejsme svázáni existujícím řešením. Zároveň se jedná v první řadě o provoz pro výukové a výzkumné účely, pro které chceme využít moderní a aktivně vyvíjenou platformu, která disponuje nejnovější technologií.

Pokud se zaměříme na aktuální dění v oblasti IaaS platforem pro privátní cloud, nejvíce zmiňovaný pojem je OpenStack. Ač je historicky nejmladší, za velmi krátký čas získal obrovskou popularitu a širokou základnu uživatelů i vývojářů. OpenStack využívají nejznámější firmy v oboru, je základem řady enterprise distribucí jako SUSE Cloud, HP Public Cloud, IBM SmartCloud, Oracle Solaris či Dell Red Hat Cloud.

Evropská organizace pro jaderný výzkum CERN nedávno oznámila [33], že svůj obří privátní cloud (500 hypervizorů, 11 000 virtuálních serverů) přesunula z OpenNebuly právě na OpenStack. Jako důvody přechodu zmínila, že chce systém s velkou škálovatelností a nechce využívat OpenNebuly, kde by byli první s takto velkým cloudem. Dalšími argumenty bylo využití širokého ekosystému – podpory load balancingu a možnost využití automatizované přípravy instancí (obdoba služby Orchestration v OpenStacku, viz kap. 4.10.2), kterými OpenNebula nedisponuje. K OpenStacku se hlásí také biomedicínská výzkumná instituce Broad Institute of Harvard and MIT, která nedávno oznámila [5], že na OpenStacku postavila svůj privátní cloud pro výzkumné účely – jedním z projektů je výzkum rakoviny, kde analyzují desítky TB dat.

Porovnáváním vybraných platforem se již zabývala řada studií a prací, jmenovitě například Klepáč, 2013 [29] nebo Jiang, 2014 [28]. Právě druhá zmíněná studie se dlouhodobě zabývá sledováním velikosti a aktivity komunity vývojářů u vybraných projektů. Výsledkem je, že OpenStack je největší a nejvíce aktivní open-source projekt pro cloud computing. [6] Jedním z mnoha

sledovaných faktorů v jeho studii je počet aktivních členů komunity (který je více než dvojnásobný oproti ostatním řešením) a měsíční počet commitů do zdrojového kódu (obr. 3.6).



Obrázek 3.6: Počet commitů za měsíc

Zdroj: [28]

Pokud budeme zkoumat využití cloudu na akademických institucích v ČR, pak kromě velmi rozšířeného VMware (který ale nesplňuje námi požadované parametry, viz kap. 2.4) už narazíme pouze na OpenNebulu, která je použita například v brněnském MetaCentru. Co se týče produkčního nasazení OpenStacku v ČR, jedná se zatím pouze o nepatrné náznaky, což může být zapříčiněno relativní mladostí technologie a omezeným počtem institucí, které si řešení privátního cloudu pořizují.

Jedním z představitelů jsou České Radiokomunikace, které v březnu 2014 oznámily, že během téhož roku postaví svůj cloud právě na OpenStacku. Generální ředitel firmy rozhodnutí zdůvodnil slovy: „OpenStack se nám líbí díky jeho skutečné škálovatelnosti. Když kus hardwaru vypadne, nic se neděje

a přidá se další. Je to skutečný cloud se všemi jeho vlastnostmi.“ [48] Vznikla také česká komunita uživatelů[18], což je vítaný impuls pro další rozšiřování v ČR.

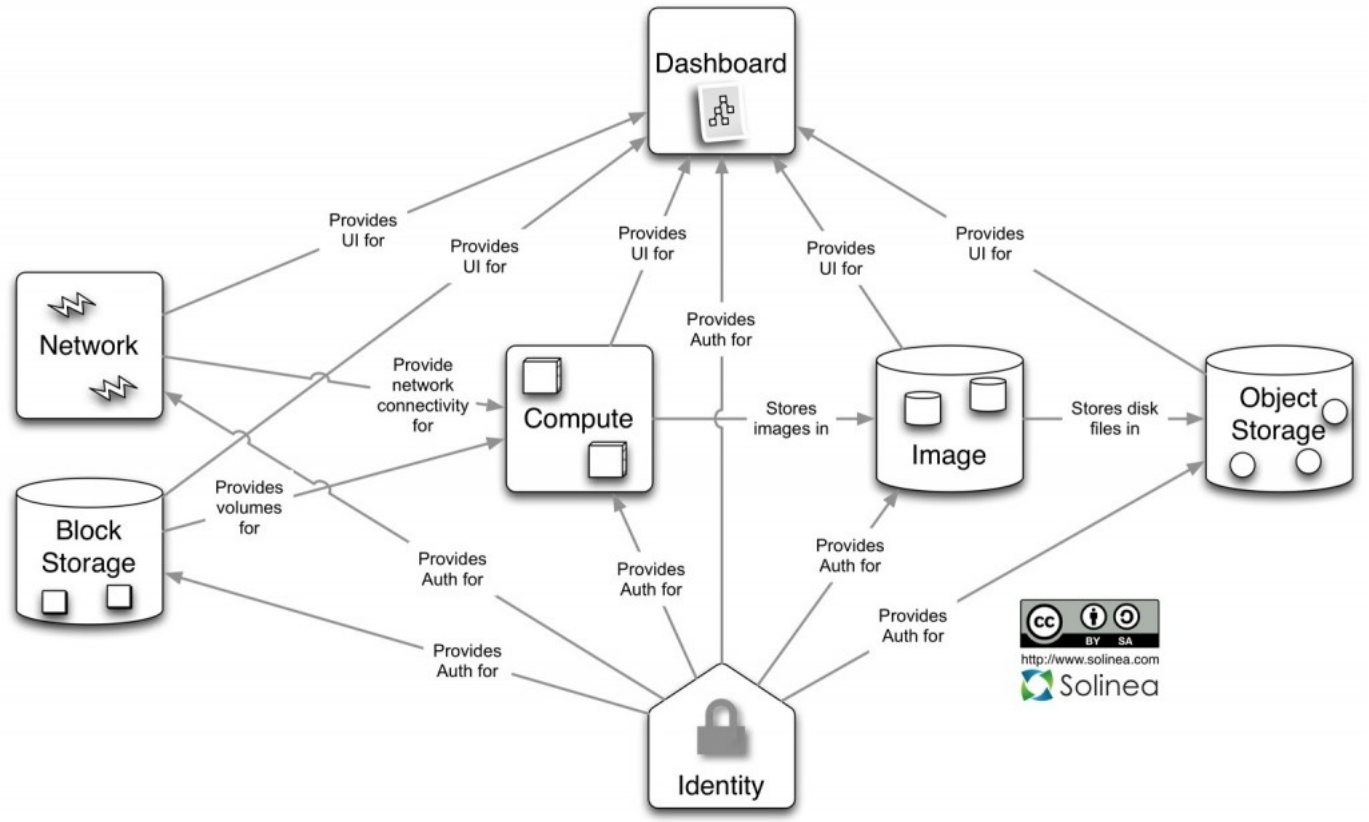
Tato práce nabízí kromě jiného přínos v tom, že nabídne hlubší pohled na implementaci řešení, které v českých akademických kruzích zatím nikdo neprovozuje a dosud neexistuje žádná studie proveditelnosti. Jak již ale bylo uvedeno výše, není to důvod jediný a samoúčelný. Ve světě open-source produktů se aktuálně těší největší přízni právě OpenStack a když na něj vsadili i ti největší a nejúspěšnější, je to nejen známka současné kvality, ale i příslib dobré budoucnosti.

V další části této práce se budeme zabývat návrhem řešení na platformě OpenStack.

[18]Dostupné na: `http://openstack.cz/`

4 Popis a návrh architektury

OpenStack je tvořen sadou komponent, které jsou mezi sebou propojeny pomocí otevřených rozhraní (API).



Obrázek 4.1: Konceptuální pohled na architekturu

Zdroj: [51]

V následujících kapitolách postupně rozebereme fungování jednotlivých komponent, jejich zapojení do celkové infrastruktury a provedeme návrh architektury pro potřeby ZČU.

4.1 Databáze

Databáze je zcela zásadní komponentou OpenStacku. V současné době jsou podporovány tyto databáze:

- MySQL
- PostgreSQL
- sqlite3

4.1.1 Návrh a zdůvodnění

V praxi se rozhodujeme mezi použitím MySQL a PostgreSQL, třetí zmíněný sqlite3 je vhodný spíše pro testovací a vývojářské účely. Jelikož CIV provozně podporuje Oracle (který zde ale nelze použít) a MySQL, pro náš návrh volíme MySQL. Pro toto rozhodnutí dále hovoří fakt, že jde o široce používaný software pro většinu instalací OpenStacku s velkou podporou komunity a oficiální dokumentace počítá s MySQL jako s primární databází. [39]

Ostrý provoz: databáze bude nasazena na samostatný virtuální server a bude zajištěna HA[1].

Pilotní provoz: databáze bude nasazena na virtuální server řídícího uzlu.

4.2 Keystone: správce identity

Keystone tvoří centrální bod, který zajišťuje ověřování uživatelů, vystavování, správu a ověřování tokenů pro přihlášené uživatele a katalog všech komponent vč. adres jejich rozhraní API.

Součástí téměř každého požadavku na rozhraní (API) libovolné služby je token vystavený Keystonem. Dotyčná služba ještě před začátkem vyřizování požadavku ověří jeho validitu a platnost, teprve pak pokračuje ve vyřizování samotného požadavku.

[1]High Availability, vysoká dostupnost; blíže v kap. 4.11.

4.2.1 Návrh a zdůvodnění

Z podstaty věci tvoří stejně jako databáze kritické místo (Single Point of Failure, SPOF), které je nutné ošetřit návrhem pro zachování vysoké dostupnosti. Touto problematikou se podrobněji zabývá samostatná kapitola 4.11.

Ostrý provoz: Keystone bude nasazen na virtuální server řídícího uzlu a bude zajištěna HA.

Pilotní provoz: Keystone bude nasazen na virtuální server řídícího uzlu.

4.3 Horizon: webový portál

Občas nazývaný také jako Dashboard tvoří webové rozhraní pro pohodlný přístup k poskytovaným službám. Je jako doplněk ke konzolovým klientům, které jsou naprogramovány jazyce Python. Využívá dostupná API jednotlivých komponent, není nijak těsněji svázán s jakoukoliv další službou.

4.3.1 Návrh a zdůvodnění

Portál představuje ústřední bod, přes který budou uživatelé provádět veškerou správu. Lze tedy očekávat vyšší nároky na výkon a navrhuji ho umístit na samostatný server.

Horizon standardně nepodporuje Single-Sign-On pro začlenění do systému jednotného přihlašování na ZČU, takže je nutné provést úpravu softwaru. Tímto se podrobněji zabývá kapitola 5.9.3.

Ostrý provoz: Horizon bude nasazen na samostatný virtuální server a bude zajištěna HA.

Pilotní provoz: Horizon bude nasazen na virtuální server řídícího uzlu.

4.4 Nova: výpočetní uzel

Výpočetní uzel má na starosti spuštění a provoz samotných virtuálních serverů. V navržené konfiguraci se nestará o diskové jednotky (o samostatné komponentě podrobněji v kapitole 4.6) ani o síťování.

4.4.1 Návrh a zdůvodnění

Ačkoliv lze OS využívat s více hypervizory a to dokonce souběžně, v našem návrhu se dále budeme zabývat implementací KVM z důvodů uvedených v kapitole 2.4.

Výpočetní uzly lze bez problémů škálovat a z povahy funkce je zřejmé, že se instalují na samostatné hardwarové (nikoliv virtuální) servery.

Ostrý i pilotní provoz: Výpočetní uzly (Nova-compute) budou instalovány na samostatné fyzické servery.

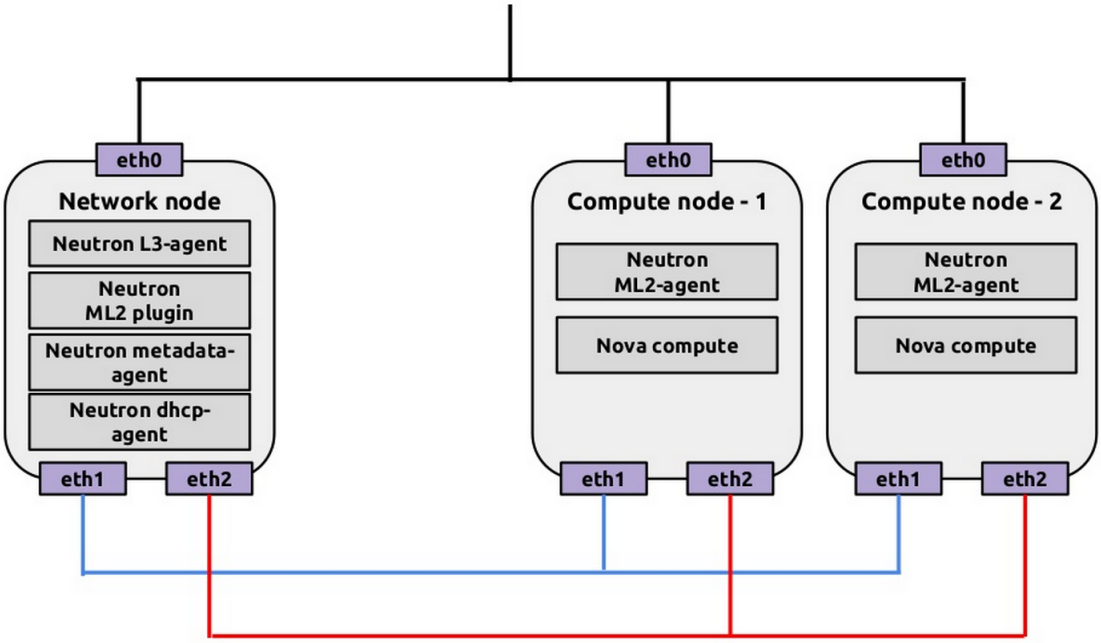
4.5 Neutron: síťování

Návrh a konfigurace sítě je nejkomplikovanější záležitostí v OS, zejména kvůli velkému množství variant, jak lze síť řešit.

Neutron se skládá z několika částí:

- **ML2 Framework** (viz níže).
- **Agent** pro zvolenou technologii networkingu, tedy např. Open vSwitch agent.
- **Firewall** pro centralizované poskytování Firewall as a Service.
- **DHCP agent** zajišťuje přidělování IP adres.
- **L3-agent** umožňuje uživatelsky vytvářet routery nad L2 vrstvou. Využívá Linuxový IP stack a iptables k provádění L3 forwardingu a NATu. Podporuje network namespaces[2], aby bylo možné přidělit např. dvěma

[2]Více informací o namespaces na: `http://www.opencloudblog.com/?p=42`



Obrázek 4.2: Schéma sítě s Neutronem

Zdroj: [50]

serverům stejnou síť a tyto byly při routingu odděleny. V každém definovaném namespace je také router a DHCP server.

Od verze Havana je součástí Neutronu framework ML2 (Modular Layer 2), který umožňuje souběžně používat řadu existujících technologií pro L2 networking. Podporuje například Linux Bridge, Open vSwitch a řadu externích prvků (Cisco Nexus, Arista, Tail-f NCS). Jsou možné následující typy segmentace: [3]

- Flat
- VLAN
- GRE
- VxLAN

Neutron rozlišuje dva typy IP adres, které je možné serverům přidělit. První z nich jsou *fixed* (pevné) IP, které jsou serveru přiřazeny napevno a

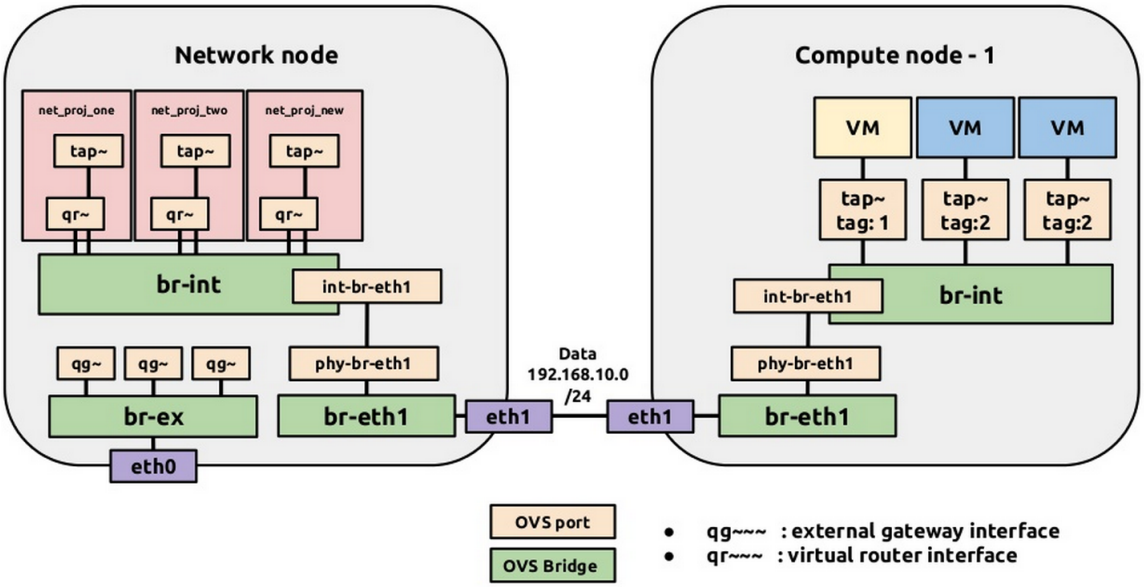
nepočítá se s jejich změnou. Tyto adresy jsou většinou neveřejné. Druhým typem jsou *floating* (plovoucí) IP, které jsou namapované na fixní. Tyto IP bývají obvykle veřejné a jsou přidělovány na žádost. Počítá se s tím, že mohou být v průběhu času přidělovány různým serverům dle aktuální potřeby.

4.5.1 Návrh a zdůvodnění

V reakci na zadávací požadavky návrh počítá s realizací pomocí softwarového řešení sítě (SDN, Software-defined networking).

Pro nejvíce izolovaný provoz bez nutnosti HW podpory na switchi navrhuji realizovat variantu s VLANy a GRE tunely. Mezi síťovým uzlem a výpočetními uzly jsou sestaveny GRE tunely po samostatné fyzické síti a na všech stranách je nainstalován Open vSwitch (s OpenFlow) řešící označování paketů podle VLAN.

Na obr. 4.3 je příklad takového návrhu. Mezi rozhraními eth1 na serverech (192.168.10.0/24) jsou realizovány GRE tunely.



Obrázek 4.3: Detail implementace Open vSwitche s GRE

Zdroj: [50]

Ostrý provoz: Neutron bude nasazen na samostatný virtuální server a bude

zajištěna HA. Vzhledem k očekávané zátěži doporučeno zvážit instalaci na fyzický server. Bude realizována varianta s Open vSwitchem, VLANy a GRE tunely a pouze v případě potíží s výkonem nasadit HW řešení spočívající v zakládání VLAN přímo na switchi přes dostupné ovladače.

Pilotní provoz: Neutron bude nasazen na samostatný virtuální server. Bude realizována varianta FLAT sítě.

4.6 Cinder: úložiště dat

OpenStack disponuje dvěma základními typy úložišť. Prvním z nich je dočasné úložiště (*ephemeral storage*), které je svázané s instancí virtuálního serveru a zanikne spolu se smazáním dotyčné instance. Fyzicky je umístěno na výpočetním uzlu a je to standardní úložiště OpenStacku.

Druhým z nich je tzv. perzistentní úložiště (*persistent storage*), které může být buď objektové (komponenta Swift, obdoba Amazon S3) nebo blokové (Cinder). Takto definované úložiště není závislé na existenci konkrétní instance, lze ho podle potřeby připojovat k libovolné instanci v cloudu. Jak je patrné, data jsou v tomto případě fyzicky umístěna mimo výpočetní uzel, což umožňuje dosáhnout vyššího výkonu i kapacity a dovoluje integraci s high-endovými externími úložišti.

Cinder se stará o správu perzistentního diskového úložiště a jeho hlavní funkcí je zprostředkovat připojení diskových jednotek k virtuálním serverům.

Sestává se ze tří hlavních komponent:

- **API** – slouží pro komunikaci s ostatními komponentami Cinderu a potažmo i OpenStacku.
- **Volume** – spravuje databázi jednotek, má přehled o jejich stavu a napřímo komunikuje s hardwarovými nebo softwarovými úložišti pomocí ovladačů.
- **Scheduler** – vybírá optimální úložiště, na kterém bude vytvořena nová jednotka.

Cinder je díky ovladačům schopen komunikovat s velkou škálou softwaro-

vých (Linux LVM[3], distribuované souborové systémy) i hardwarových (HP, IBM, NetApp, ...) úložišť.

Je podporována práce s více úložišti naráz a při vytváření nové jednotky se vybírá filtrováním (dle uživatelsky definovaných požadavků) a řazením (např. dle obsazenosti) nejvhodnější úložiště obdobně, jako když Nova hledá optimální hostitelský server při vytváření nové instance.

Od verze Havana je podporována řada užitečných vlastností:

- **Šifrování** – používá dm-crypt.
- **Migrace jednotek** – i mezi různými typy úložišť (Cinder je v roli proxy serveru, který spustí příkaz *dd*).
- **Omezování datového toku** – limituje I/O požadavky (odděleně pro čtení a zápis).

Pro náš návrh je důležitý zejména I/O limit, díky kterému můžeme zabránit situaci, kdy jedna instance zcela zahltí připojené úložiště I/O požadavky.

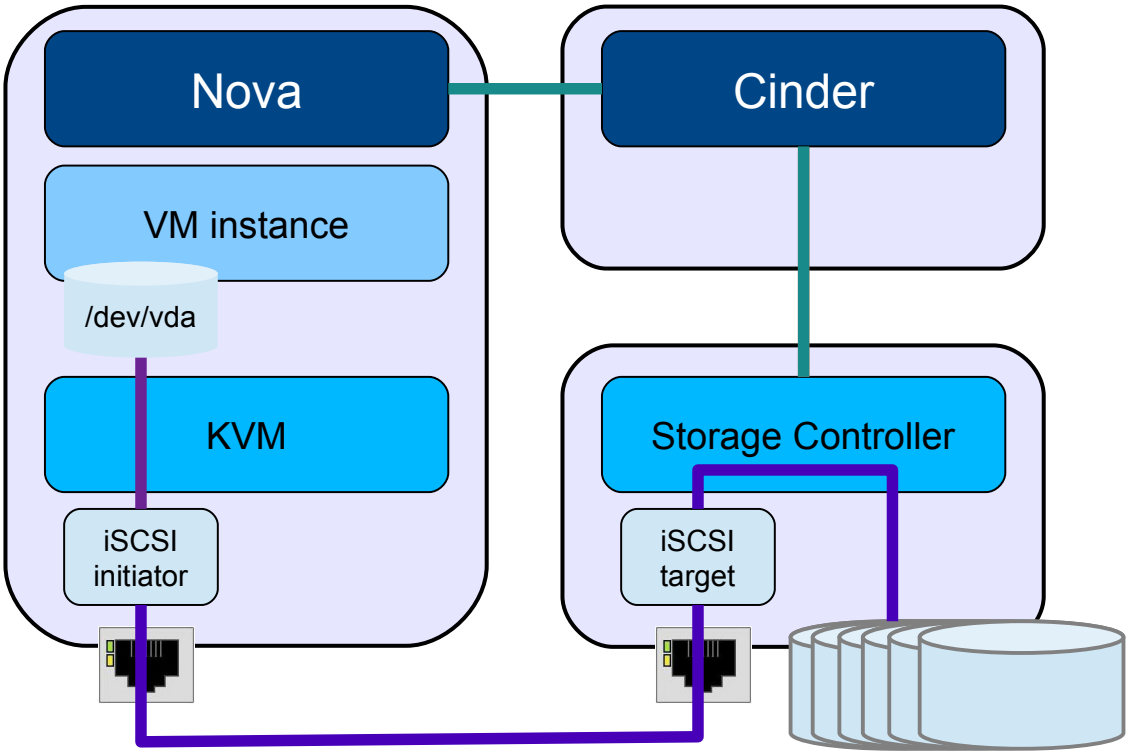
4.6.1 Návrh a zdůvodnění

Pokud chceme k serverům připojovat bloková zařízení, je nutné vybrat takový protokol, který to umožňuje. V praxi se nejčastěji setkáváme s rozhodováním mezi iSCSI a FC (Fibre Channel). Jsou sice podporovány i distribuované souborové systémy jako NFS, AFS či GlusterFS, ale technicky to funguje tak, že je na daném souborovém systému vytvořen soubor, který je následně namapován jako samostatná jednotka s vlastním souborovým systémem. To ale není z hlediska výkonu a uspořádání optimální, proto o této variantě nebudeme uvažovat.

V našem návrhu se budeme dále zabývat implementací iSCSI, což je protokol založený na IP, který umožňuje klientům (initiators) komunikovat s I/O zařízeními pro ukládání dat (targets) pomocí paketů TCP/IP. [26]

Výhodou iSCSI oproti FC je zejména cena, neboť pro FC je třeba pořídit speciální a drahý hardware (karty, přepínače, kabeláž) a iSCSI si vystačí s

[3]Logical Volume Management, detaily na: `https://sourceware.org/lvm2/`



Obrázek 4.4: Storage node s externím uložištěm

Zdroj: [52]

obyčejným Ethernetem. Lze ho dokonce provozovat na existující síti, ačkoliv to není s ohledem na výkon doporučováno. Navíc s nástupem 10 GbE přestává být problém rychlost rozhraní ve srovnání s FC. Studie Andrewa Reichmana [31] ukazuje, že při běžném nasazení je FC 3.8× nákladnější než iSCSI.

Pro iSCSI také hovoří široká implementace napříč operačními systémy a fakt, že není omezen lokalitou (LAN), ale může fungovat po běžné síti na libovolnou vzdálenost.

Obr. 4.4 ukazuje navržený model, kdy Cider komunikuje s externím úložištěm tak, aby došlo k nastavení potřebných parametrů pro nově vznikající diskovou jednotku. Ta je pak propojena na přímo mezi externím úložištěm a virtuálním serverem – datové toky neprochází přes Cinder, což by bylo nežádoucí a mělo by to vliv na pokles I/O. Více informací o způsobu fungování podává kap. 4.6.2.

Ostrý provoz: Cinder bude nasazen na samostatný virtuální server a bude

zajištěna HA. Bude ovládat externě připojené diskové úložiště.

Pilotní provoz: Cinder bude nasazen na samostatný virtuální server. Cloudu bude poskytovat lokální diskový prostor, napojení na externí úložiště nebude řešeno.

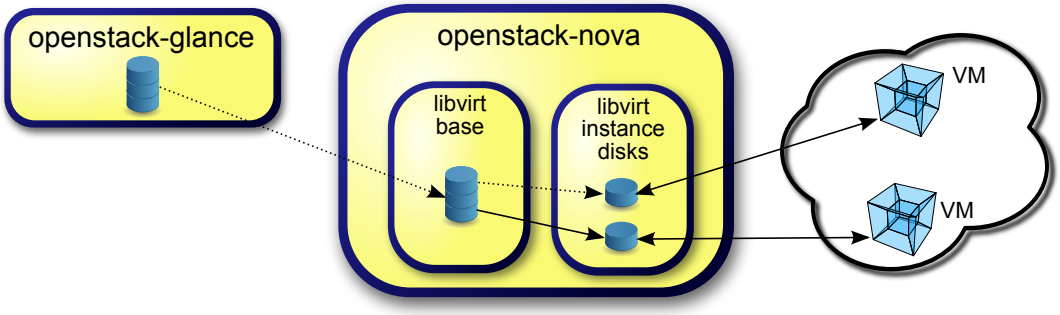
4.6.2 Tok požadavků při připojování jednotky

1. **Nova** zavolá přes API **Cinder** a předá mu požadavek s parametry (hostitel, jméno iSCSI initiatoru, ...).
2. **Cinder-api** předá zprávu do **cinder-volume**.
3. Proběhne kontrola chyb a zavolá se ovladač zařízení, na kterém se bude jednotka vytvářet.
4. **Ovladač zařízení** provede přípravu jednotky vč. nastavení práv pro připojení.
5. **Ovladač zařízení** navrátí informace pro připojení, tyto jsou pak předány do **nova**.
6. **Nova** vytvoří spojení na úložiště podle dodaných parametrů.
7. **Nova** předá připojenou jednotku hypervizoru virtualizace.

4.7 Glance: katalog obrazů

Glance zajišťuje centralizovanou správu katalogu obrazů virtuálních serverů. Má několik funkcí:

- Ukládá obrazy (images), které jsou používány při vytváření instancí.
- Poskytuje uživatelům katalog dostupných obrazů.
- Pro komponentu Nova zajišťuje dodávku obrazů.
- Zajišťuje vytváření snapshotů běžících serverů pro zálohovací účely.



Obrázek 4.5: Proces přenosu a použití obrazu

Zdroj: [4]

Samotné obrazy mohou být fyzicky uloženy buď na lokálním souborovém systému, objektovém úložišti (Swift, Amazon S3) nebo přístupné přes HTTP.

Při vytváření nové instance z existujícího obrazu funguje proces tak, jak je zachyceno na obrázku 4.5. Výpočetní uzel (openstack-nova) si z Glance stáhne obraz zvoleného operačního systému. Ten je uložen v nekomprimované podobě do tzv. base adresáře na výpočetním uzlu a z něj je následně vytvořen disk pro nově vznikající instanci. V případě opakovaného vytváření instance ze stejného obrazu je tedy uplatněn vliv lokální mezipaměti (cache) a minimalizován čas nutný pro vytvoření.

4.7.1 Návrh a zdůvodnění

Ostrý provoz: Glance bude nasazen na samostatný virtuální server a bude zajištěna HA.

Pilotní provoz: Glance bude nasazen na samostatný virtuální server.

Obrazy serverů budou v obou případech uloženy lokálně, vzhledem k tomu že součástí návrhu není objektové úložiště Swift (detaily a zdůvodnění v kap. 4.10.1).

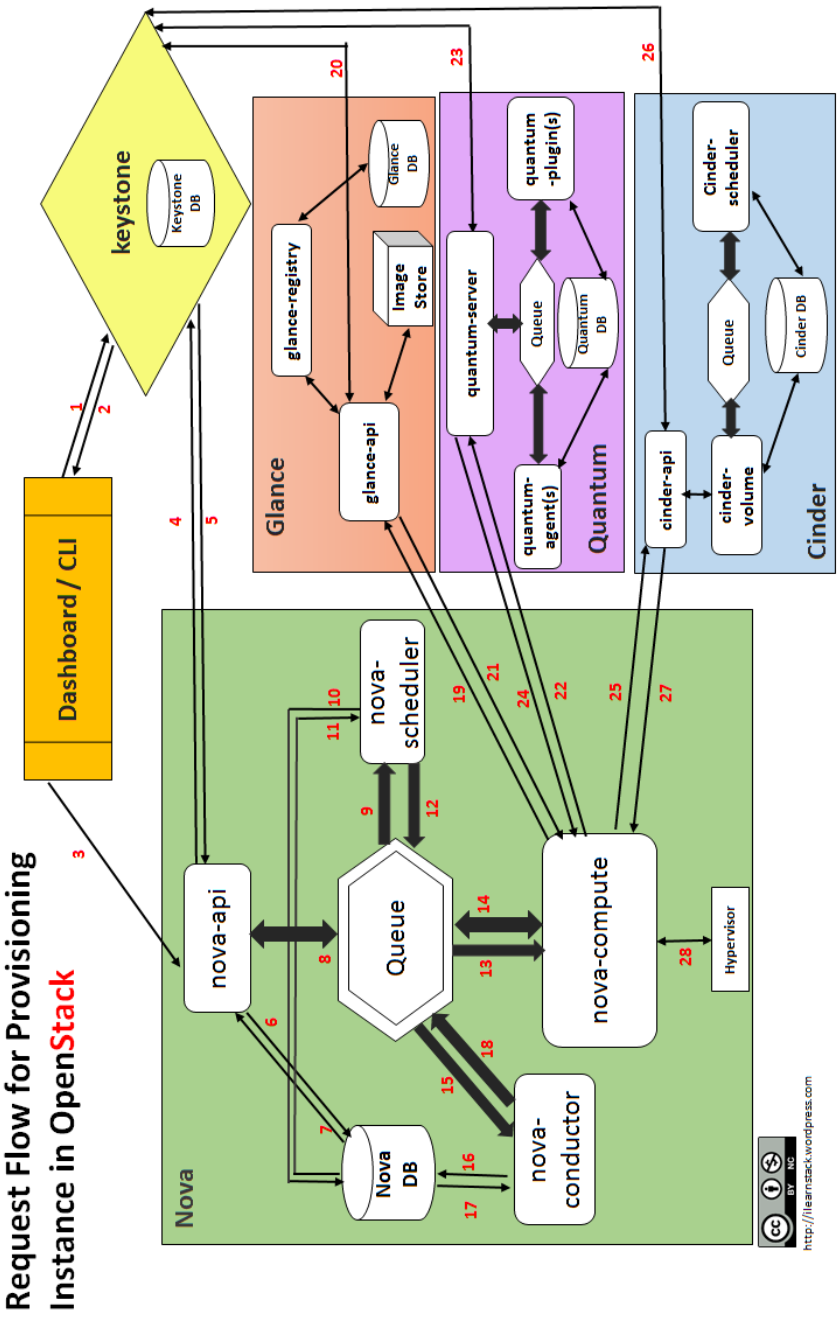
4.8 Tok požadavků při zakládání instance

Obrázek 4.6 znázorňuje pro lepší představu proces vytváření nové instance virtuálního serveru v OpenStacku. Dochází tam k interakci mezi jednotlivými komponentami a výsledkem je start virtuálního serveru.

V krocích 3–12 dochází k plánování, v krocích 22–24 je připravována síť a diskové úložiště se připraví v kroku 25–27. Níže následuje kompletní popis všech kroků: [22]

Každá komponenta (Nova, Glance, ...) je tvořena několika souvisejícími službami (nova-api, nova-scheduler, ...), které mezi sebou komunikují přes RPC. Komponenty mezi sebou využívají komunikaci přes REST API.

1. **Dashboard nebo CLI** obdrží přihlašovací údaje a požádá **Keystone**, aby je ověřil.
2. **Keystone** ověří, jestli jsou obdržené údaje platné a pokud ano, tak zašle zpět token, kterým se bude **Dashboard/CLI** nadále prokazovat.
3. **Dashboard/CLI** zpracuje zadané parametry pro novou instanci a pošle je přes REST do **nova-api** společně s tokenem.
4. **Nova-api** obdrží požadavek a požádá **Keystone** o ověření přihlašovacího tokenu a zaslání přístupových práv uživatele.
5. **Keystone** ověří token a zašle role i práva uživatele zpět.
6. **Nova-api** provádí interakci s **nova-database**.
7. Je vytvořen inicializační záznam v DB pro novou instanci.
8. **Nova-api** zašle rpc.call požadavek do **nova-scheduler** a čeká na doplnění ID hostitelského serveru.
9. **Nova-scheduler** vyzvedne požadavek z fronty (queue).
10. **Nova-scheduler** se dotazuje **nova-database** a pokud není uživatelsky specifikován konkrétní hostitelský server, hledá se pomocí filtrování a vah ideální hostitelský server, kde bude nová instance založena.
11. Je navráceno ID hostitelského serveru.



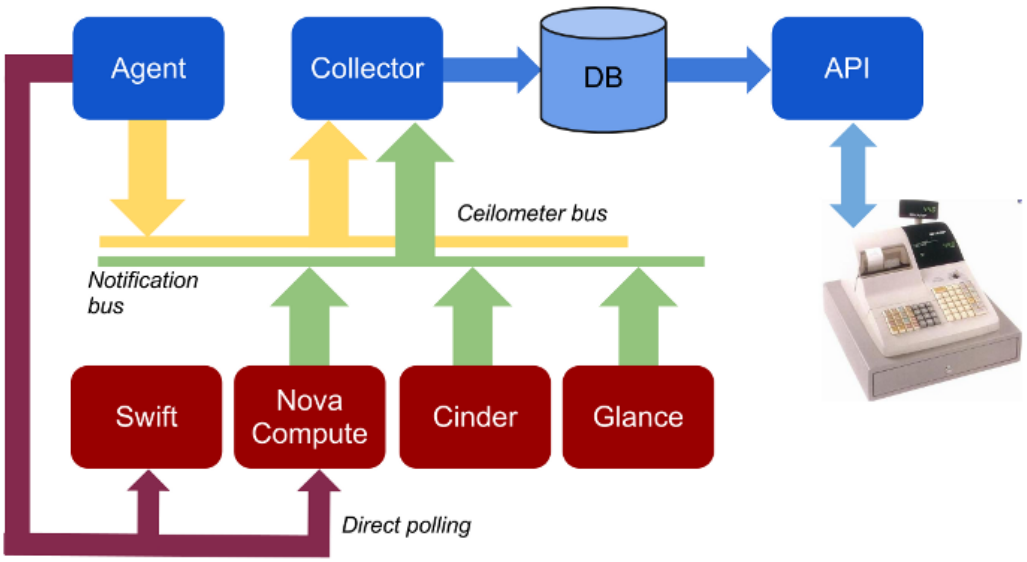
Obrázek 4.6: Tok požadavků

Zdroj: [22]

12. **Nova-scheduler** zašle rpc.cast požadavek na **Nova-compute** k založení nové instance na vybraném hostiteli.
13. **Nova-compute** vyzvedne požadavek z fronty.
14. **Nova-compute** zašle rpc.call požadavek na **nova-conductor** k vyžádání podrobnějších informací o hostiteli a parametrech (RAM, CPU, disk) nové instance.
15. **Nova-conductor** vyzvedne požadavek z fronty.
16. **Nova-conductor** se dotazuje **nova-database**.
17. Jsou navráceny informace o instanci.
18. **Nova-compute** si vyzvedne informace o instanci z fronty.
19. **Nova-compute** provede REST volání s tokenem uživatele do **Glance** (služba databáze obrazů) ke zjištění URL obrazu a nahrání obrazu z úložiště obrazů.
20. **Glance-api** požádá **Keystone** o ověření uživatelského tokenu a provede zadané úkoly.
21. **Nova-compute** získá metadata o obrazu (image) instance.
22. **Nova-compute** provede REST volání s tokenem uživatele do **Neutron** (síťová služba) k alokaci a konfiguraci síťových zdrojů tak, aby instance získala IP adresu.
23. **Neutron-server** (dříve quantum-server) požádá **Keystone** o ověření uživatelského tokenu a provede alokaci zdrojů.
24. **Nova-compute** obdrží informace o síti pro novou instanci.
25. **Nova-compute** provede REST volání s tokenem uživatele do Cinder (storage) k připojení svazku nové instanci.
26. **Cinder-api** požádá **Keystone** o ověření uživatelského tokenu a provede alokaci svazku (disku).
27. **Nova-compute** obdrží informace o blokovém zařízení (storage).
28. **Nova-compute** vygeneruje data pro ovladač hypervisora a spustí požadavek na vytvoření instance (přes libvirt nebo API).

4.9 Ceilometer: měření zdrojů

Slouží pro sběru informací o využívání zdrojů v cloudu a tvorbu podkladů pro vyúčtování. Architekturu ilustruje obr. 4.7.



Obrázek 4.7: Architektura Ceilometeru

Zdroj: [11]

Ceilometer sbírá data o využívání zdrojů od komponent v cloudu pro každého uživatele/projekt, zpracovává je a ukládá do své databáze. Následně poskytuje výstupy ve formě REST API pro použití v dalších aplikacích – jednou z takových aplikací je i portál Horizon, kde je možné pohodlně prohlížet výstupy měření.

Zaznamenává tři typy hodnot:

1. **Kumulativní** (cumulative), rostou v čase (např. počet hodin, po kterou je instance spuštěna).
2. **Diskrétní hodnoty** (gauge), např. počet přidělených plovoucích IP adres.
3. **Změnové** (delta), např. přenos dat po síti.

Jsou sledovány tyto parametry:

1. **Nova**: počet instancí, využití CPU, využití diskového I/O
2. **Neutron**: virtuální sítě, routery, porty, IP adresy
3. **Glance**: velikost obrazů, nahrávání a stahování obrazů
4. **Cinder**: počet a velikost diskových jednotek

4.10 Ostatní komponenty

Součástí OS Havana jsou ještě další komponenty, které nebyly součástí požadavků a proto nejsou zahrnuty do návrhu.

4.10.1 Swift

Swift je objektové úložiště (obdoba Amazon S3), které je navrženo pro bezpečné ukládání velkého množství dat. Data jsou přístupná přes API a jsou uložena ve více kopiích na různých serverech pro maximální bezpečnost. Úložiště je kompletně distribuované a velmi snadno škálovatelné.

Pro samotný provoz navrženého cloudu však není Swift vyžadován, data virtuálních serverů budou umístěna na existujících externích úložištích.

4.10.2 Heat

Obdoba AWS CloudFormation má za úkol přípravu spouštění aplikací na OpenStacku. Lze ji využít ke spuštění předdefinovaných scénářů, tedy např. spustit sadu webserverů s nastaveným load-balancerem a Drupalem. RedHat tuto komponentu využívá např. pro nasazování svého projektu OpenShift.

V návrhu není zařazen, neboť nejde o nezbytnou službu.

4.11 Škálovatelnost a High Availability

Pro zajištění maximální dostupnosti a možnosti rozšiřování služeb je nutné již v návrhu reflektovat tento požadavek. Systémy vysoké dostupnosti HA se snaží minimalizovat tato rizika: [23]

1. **Výpadek systému**: nastává, když je poskytovaná služba nedostupná.
2. **Ztráta dat**: zničení nebo poškození dat.

Důležitým aspektem projektování HA systémů je prevence rizikových míst, které mohou způsobit pád celého systému (Single Points Of Failure, SPOFs). Tato místa jsou zejména:

1. **Síť'ové komponenty**, jako routery, switche
2. **Aplikace**
3. **Datová úložiště**
4. **Služby** jako napájení, klimatizace, požární systém

Tato práce bude pokrývat zejména zajištění vysoké dostupnosti u jednotlivých komponent OpenStacku.

Zajištění HA pro jednotlivé aplikace se liší podle toho, jestli jde o *stavovou* nebo *bezstavovou* službu.

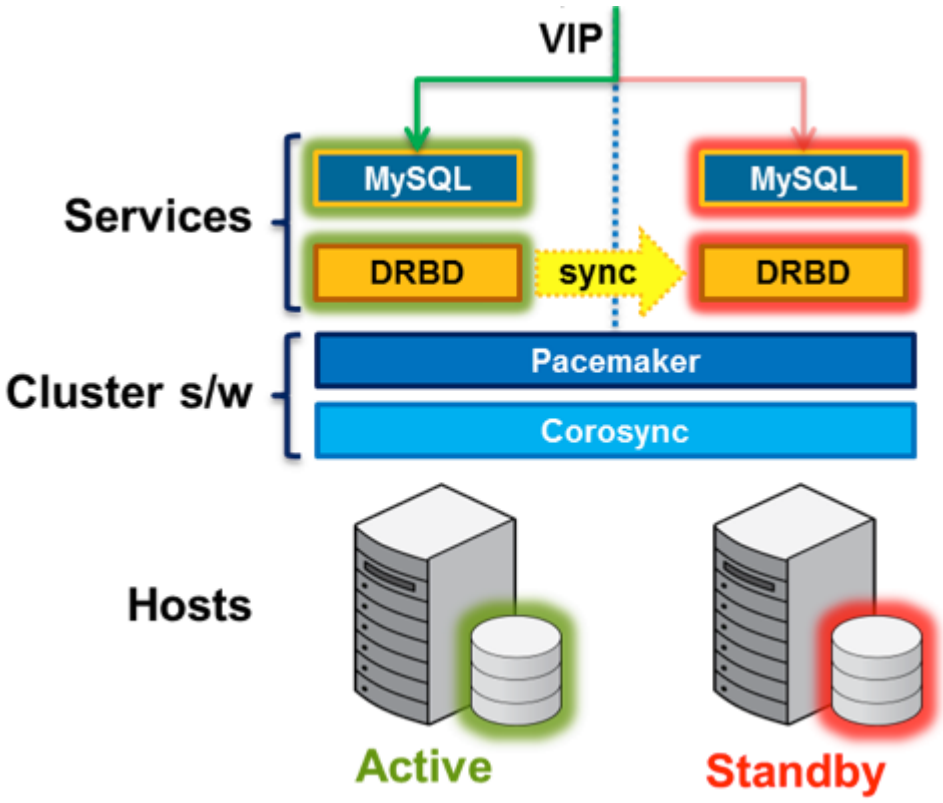
Bezstavová služba je taková, která po vyřízení dotazu neprovádí žádnou další akci. Typickým příkladem je protokol HTTP. V architektuře OS se jedná o služby: nova-api, nova-conductor, glance-api, keystone-api, neutron-api a nova-scheduler. U těchto služeb se HA zajišť'uje spuštěním více instancí na různých serverech, mezi kterými se provádí *load-balancing*, neboli vyvažování zátěže.

Pro stavovou službu je typické, že následující požadavek je závislý na prvotním požadavku, takže nelze HA řešit spuštěním více instancí s vyvažováním zátěže. Stavové služby v OS jsou centrální databáze a fronta zpráv.

4.11.1 HA pro databázi

Jako hlavní databázový server je použita MySQL. V ní lze HA řešit více způsoby:

Active/Passive – struktura počítá s nasazením dvou serverů, přičemž jeden je jako hlavní (Active) a druhý jako záložní (Passive). Za běžného provozu všechny požadavky vyřizuje hlavní server a v případě výpadku hlavního je provoz přepnut na záložní.



Obrázek 4.8: Realizace HA pro MySQL přes DRBD a Pacemaker

Zdroj: vlastní zpracování na základě [35]

Technicky je to řešeno tak, že se databáze ukládají na DRBD[4], což je blokové zařízení synchronizované přes síť (obdoba RAID[5]). Oba servery mají své IP adresy a krom toho ještě jednu virtuální (Virtual IP, VIP), kterou si

[4]Distributed Replicated Block Device, detaily na: `http://www.drbd.org/`

[5]Redundant Array of Inexpensive/Independent Disks

mezi sebou předávají podle toho, který server je v provozu. Právě na tuto IP se připojují ostatní služby. Na obou serverech je nainstalován Pacemaker, který kontroluje dostupnost druhého serveru a podle potřeby provádí nastavené akce (spuštění služby na záložním serveru v případě výpadku, převzetí virtuální IP, ...). Proces je znázorněn na obr. 4.8.

Tento způsob nicméně neumožňuje rozkládání zátěže a proto se nehodí pro systémy, kde potřeba škálování.

Active/Active – tento způsob počítá s vybudováním MySQL clusteru, který provádí synchronní replikaci InnoDB databází, jak naznačuje obr. 4.9. Lze použít MySQL cluster[6] nebo Galera cluster[7], přičemž druhý zmiňovaný nabízí téměř dvojnásobnou propustnost oproti MySQL clusteru [7]; volíme tedy řešení pomocí Galera clusteru.

Výhodou tohoto řešení je, že lze použít více serverů, všechny jsou v provozu a všechny vyřizují požadavky. Kromě zajištění vysoké dostupnosti lze pomocí takto vytvořeného clusteru i škálovat. Je ale nutné využít load-balancer, at' už hardwarový nebo softwarový (HAproxy + Keepalived).

Pro zajištění HA a snadné škálovatelnosti autor práce doporučuje realizaci formou databázového clusteru Galera.

4.11.2 HA pro frontu zpráv

Pro správu fronty (Advanced Message Queuing Protocol, AMQP) OpenStack využívá program RabbitMQ. Ten má podporu pro implementaci clusteru již zabudovanou, takže stačí provést instalaci na vybraný počet uzlů a provést drobnou úpravu nastavení[8].

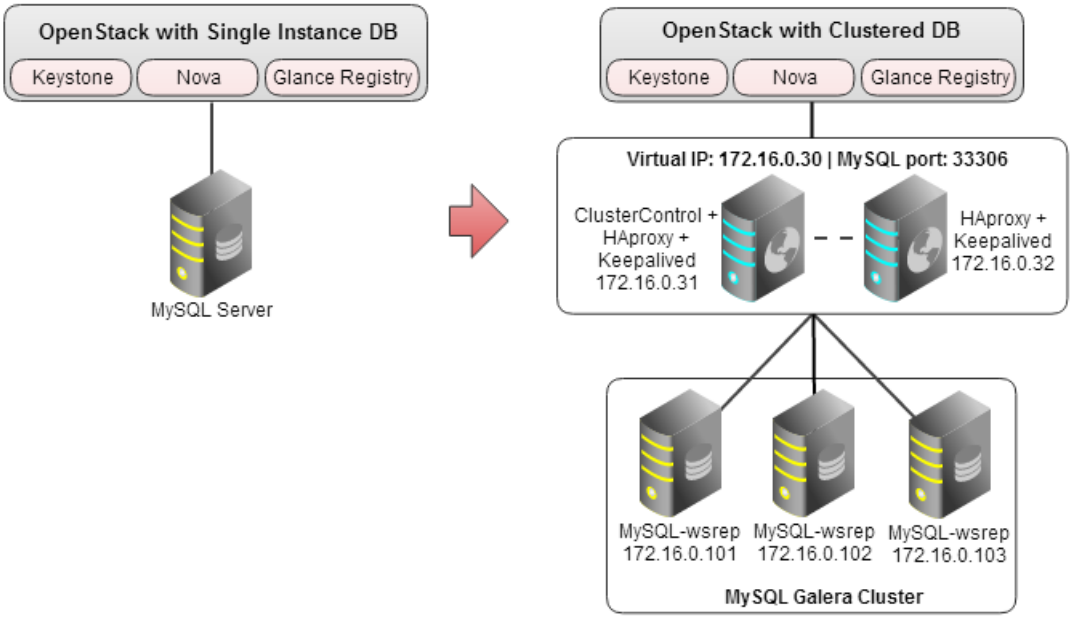
Není třeba využívat load-balancer, postačí upravit konfiguraci služeb OpenStacku, které s frontou pracují:

```
rabbit_hosts=rabbit1:5672,rabbit2:5672
```

[6] `http://www.mysql.com/products/cluster/`

[7] `http://galeracluster.com/`

[8] Detaily lze nalézt na: `http://docs.openstack.org/high-availability-guide/content/_configure_rabbitmq.html`



Obrázek 4.9: Realizace HA pro MySQL přes Galera a HAproxy.

Zdroj: [49]

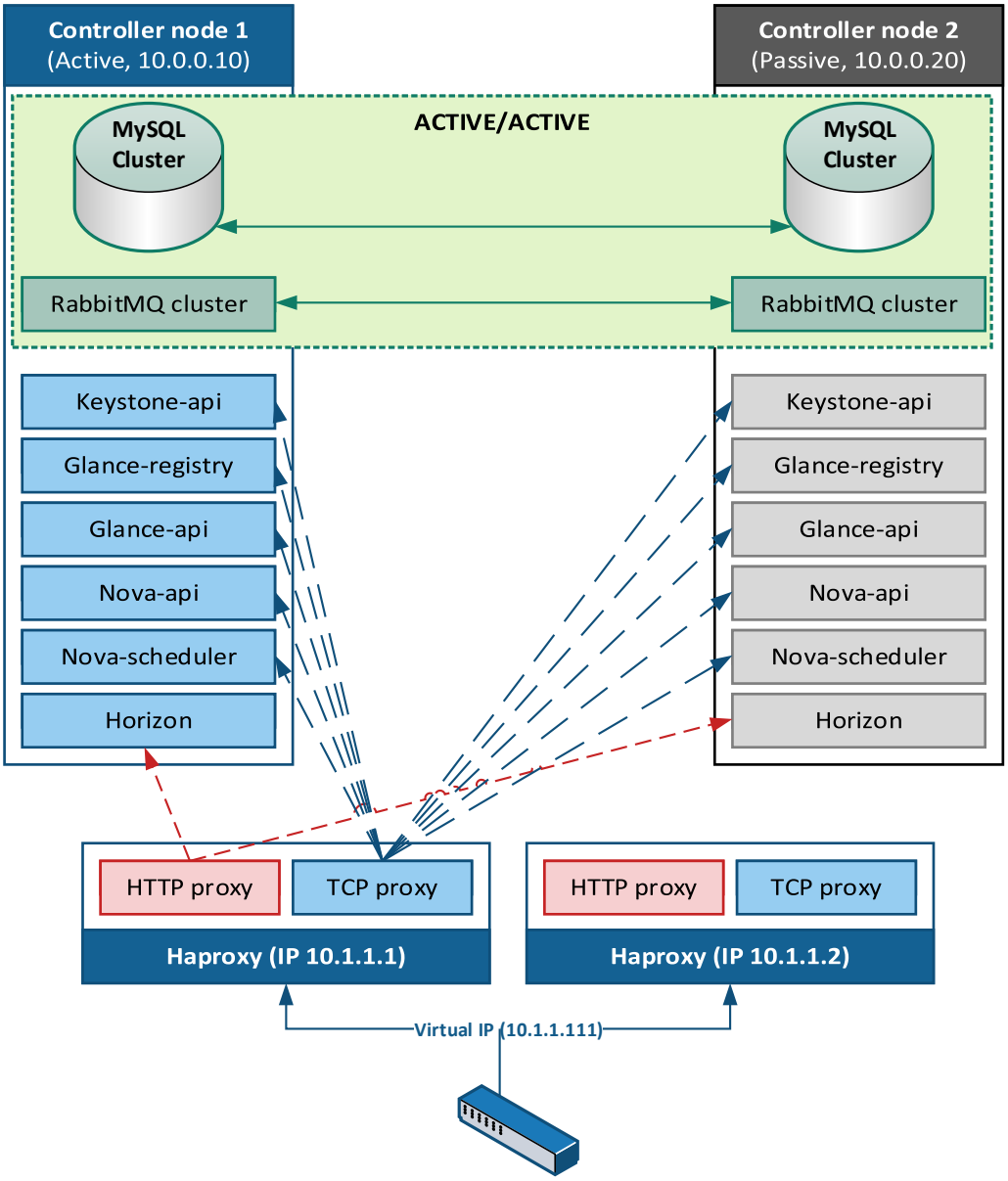
4.11.3 HA pro ostatní služby

Zajištění vysoké dostupnosti ostatních služeb je řešeno instalací na dva servery v režimu Active/Passive. Přes HAproxy prochází veškerá komunikace na dané služby. Ta se stará o správně nasměrování požadavků podle toho, který server je v provozu. HAproxy by měl být replikovaný, aby netvořil SPOF.

4.11.4 Schéma návrhu kompletního HA

Obrázek 4.10 shrnuje návrh HA řešení. Obsahuje dva řídící uzly (Controller node 1 a 2), na kterých běží služby, u nichž chceme zajistit HA. Jak již víme z předchozích kapitol, MySQL a RabbitMQ nemusí být v režimu Active/Passive, ale pro zjednodušení jsou zahrnuty do načrtnutých řídících uzlů, přičemž replikace je u nich Active/Active.

Pro přepínání Active/Passive služeb je nainstalován HAproxy. V návrhu je zdvojený, přičemž v případě nedostatku hardware je možné jednotlivé instance nainstalovat přímo na řídící uzly. Všechny služby, které se k HA



Obrázek 4.10: Návrh HA architektury OpenStacku

Zdroj: vlastní zpracování

OpenStacku připojují využívají virtuální IP (VIP) HAproxy, tedy 10.1.1.111.

MySQL a RabbitMQ cluster lze využít buď napřímo (např. přes round-robin DNS) nebo také přes HAproxy. Autor doporučuje použít HAproxy, protože již využita i pro ostatní komponenty návrhu.

4.12 Zálohovací schéma

Při zálohování komponent OpenStacku je třeba provést zálohu centrální databáze a pak také všech konfiguračních souborů jednotlivých služeb.

Zálohu zvolené databáze MySQL lze provést přes konzolový program mysqldump, který je součástí distribuce. V tomto případě však dojde k dočasnému uzamknutí tabulek po dobu provádění zálohy. Zálohu přes mysqldump provedeme následujícím příkazem, výstupem bude soubor openstack.sql obsahující data všech databází:

```
# mysqldump --all-databases > openstack.sql
```

Pokud potřebujeme zajistit dostupnost i při probíhající záloze, je možné využít open-source software Percona XtraBackup[9], který umožňuje provádět zálohy MySQL InnoDB bez přerušení provozu. Pro provedení zálohy všech databází ze standardního datového adresáře (je uveden v konfiguračním souboru /etc/my.cnf) do složky `/data/backups` spustíme příkaz:

```
# innobackupex /data/backups
```

Nova

Na výpočetním i řídícím uzlu je nutné provádět zálohu adresářů `/etc/nova` a `/var/lib/nova` s výjimkou `/var/lib/nova/instances` na výpočetním uzlu, kde jsou uloženy obrazy virtuálních serverů (pokud není využit Cinder) – není doporučeno takto zálohovat běžící servery, protože hrozí riziko, že bude záloha poškozena. Pro zálohování samotných serverů je vhodné využít LVM snapshoty, které lze provést i u běžící instance.

[9]Detaily lze nalézt na: `http://www.percona.com/software/percona-xtrabackup`

Snapshoty lze vytvořit přes portál (Project → Instances → Create Snapshot), nebo přes příkazovou řádku:

```
# cinder snapshot-create --display-name NÁZEV ID_JEDNOTKY
```

Glance

Na uzlu obsahujícím obrazy serverů je nutné zálohovat adresáře `/etc/glance` a `/var/lib/glance`.

Keystone

Pro službu identity stačí zálohovat adresář `/etc/keystone`.

Cinder

Blokové úložiště má uloženou konfiguraci v adresáři `/etc/cinder`, který je tedy nutný zálohovat.

Horizon

Konfigurační soubor portálu je ve složce `/etc/openstack-dashboard`.

4.12.1 Obnova

V případě obnovy databáze je nutné nejprve zastavit všechny služby, které ji využívají (tedy pro databázi *nova* všechny procesy *nova-**) a poté provést obnovu:

```
# mysql nova < nova.sql
```

Pokud je třeba obnovit ze zálohy celý server, je nutné ho znovu nainstalovat a ze zálohy obnovit potřebné konfigurační soubory nebo databázi (záleží na tom, jaký měl dotyčný server roli, tedy co na něm bylo nainstalováno).

4.13 Logování

OpenStack standardně zapisuje logy do adresáře `/var/log`. Při vzrůstajícím počtu serverů ale není vhodné mít logy uchované na každém serveru zvlášť, ale zapnout centrální logování. Kromě snadnější analýzy je tím zajištěna i větší bezpečnost logů (např. při kompromitaci konkrétního serveru). Autor doporučuje využívat centrálního logování.

Syslog, který OpenStack podporuje, umožňuje zapnout logování přes síť. Pro jeho aktivaci je nutné v konfiguraci (např. `/etc/nova/nova.conf`) zapnout použití syslogu:

```
use_syslog=True
syslog_log_facility=LOG_LOCAL0
```

Dále je nutné nakonfigurovat syslog, aby komunikoval přes síť se serverem. Provedeme to vytvořením souboru `/etc/rsyslog.d/client.conf` s tímto obsahem:

```
*.* @192.168.1.1
```

Namísto IP 192.168.1.1 vložíme adresu centrálního syslog serveru.

4.13.1 Historie IP adres

Jedním z požadavků zadavatele bylo znát historii přidělovaných IP adres, aby bylo možné vystopovat konkrétního uživatele (server), který měl přidělenou určitou IP v daném čase (např. kvůli případným stížnostem na nelegální aktivitu).

Databáze OpenStacku nedisponuje historií přidělených adres, proto je nutné situaci řešit jinak. Jsou k dispozici dva způsoby.

Prvním z nich je obyčejné ruční (případně skriptem zautomatizované) procházení textových logů, kam se zapisuje při vytváření serveru přidělená IP adresa. Jde o méně pohodlný způsob, ale je velmi snadno technicky realizovatelný a nevyžaduje žádné další úpravy.

Druhým způsobem je odchytávání AMQP zpráv z fronty OpenStacku, což umožňuje například program yagi[10]. Takto získané informace je nutné dále zpracovat a uložit si je do vlastní datové struktury, ve které se bude uchovávat požadovaná historie. Tento přístup je více pracnější, neboť vyžaduje instalaci dodatečného software, jeho konfiguraci a navržení vlastní datové struktury na ukládání dat včetně obslužných skriptů prot vyhledávání dat.

4.14 Monitoring

Pro zajištění bezproblémového běhu je důležité monitorovat jak provozní veličiny, tak stav služeb. Na ZČU je využíván monitorovací systém Nagios, proto budou v tomto návrhu zmíněny základní postupy pro tento systém. Kromě níže naznačených postupů existuje také balík `nagios-plugins-openstack` pro Debian, který obsahuje skripty pro testování některých komponent (Glance, Keystone, Nova-API a Swift). Další informace poskytne také článek *Some practical considerations for monitoring in OpenStack cloud*[11].

4.14.1 Procesy

Běh služeb lze monitorovat základním způsobem tak, že se zkontroluje tabulka procesů systému. Níže uvedený příklad zkontroluje, zda na výpočetním uzlu běží služba nova-compute. Do konfigurace Nagios serveru se přidá následující:

```
define service {
    host_name cloud-102.civ.zcu.cz
    check_command check_nrpe_1arg!check_nova-compute
    use generic-service
    notification_period 24x7
    contact_groups sysadmins
    service_description nova-compute
}
```

[10] Detaily lze nalézt na: `https://github.com/rackerlabs/yagi`

[11] `http://www.mirantis.com/blog/openstack-monitoring/`

Na výpočetní uzel (cloud-102.civ.zcu.cz) je nainstalován NRPE[12] a přidána tato řádka do konfigurace:

```
command[check_nova-compute]=/usr/lib/nagios/plugins/check_procs
 -c 1: -a nova-compute
```

Uvedený postup se zopakuje pro všechny monitorované procesy.

4.14.2 Provozní veličiny

Dále je vhodné monitorovat využití zdrojů na serverech, zejména zátěž, obsazenost disku, využití paměti RAM či síťové I/O. Toto lze monitorovat např. přes SNMP nebo přes již zmíněný Nagios. Příklad konfigurace Nagiosu pro varování o docházejícím místu na disku:

```
define service {
    host_name cloud-102.civ.zcu.cz
    check_command check_nrpe!check_all_disks!20% 10%
    use generic-service
    contact_groups sysadmins
    service_description Disk
}
```

Na monitorovaném serveru je do konfigurace NRPE přidána tato řádka:

```
command[check_all_disks]=/usr/lib/nagios/plugins/check_disk
 -w $ARG1$ -c $ARG2$ -e
```

Při poklesu volného místa pod 20 % se v Nagiosu zobrazí varování a při poklesu pod 10 % chyba.

[12]NRPE - Nagios Remote Plugin Executor

4.15 Bezpečnost, hesla

Otázka bezpečnosti OpenStacku je velmi rozsáhlá a je k tomuto tématu vydána samostatná příručka[13]. Tato kapitola se tedy bude věnovat jen základnímu popisu toho, jak je zabezpečení řešeno a zmíní několik doporučení.

RabbitMQ

Po instalaci serveru pro výměnu zpráv RabbitMQ je vytvořen automaticky uživatel *guest* s heslem *guest*, proto je nutné toto heslo před prvním použitím změnit:

```
rabbitmqctl change_password guest NOVÉ_HESLO
```

Výše nastavené heslo se pak musí uvádět v konfiguračním souboru každé komponenty, která se k serveru připojuje (parametr `rabbit_password`).

Webové služby

Jak webový portál Horizon, tak všechna REST[14] API jednotlivých komponent pracují na protokolu HTTP[15], který je standardně nezabezpečený a pro vyšší bezpečnost je doporučeno využít SSL[16] u veškeré komunikace. Jelikož u API není SSL nativně podporováno OpenStackem, je nutné využít libovolnou SSL proxy (Pound, Stud, Apache, nginx, ...).

Databáze

Jednotlivé komponenty se připojují k centrální databázi MySQL. Standardně je komunikace nešifrovaná, tudíž je doporučeno nakonfigurovat MySQL server

[13]Kompletní verze k dispozici na: `http://docs.openstack.org/security-guide/content/`

[14]Representational State Transfer

[15]Hypertext Transfer Protocol

[16]Secure Sockets Layer

na podporu SSL nebo využít pro spojení komponent s databází oddělenou privátní síť (případně VPN[17]).

Pro zapnutí SSL na MySQL je nutné mít vygenerován certifikát (veřejný i privátní klíč) a v souboru `/etc/my.cnf` nastavit cesty k tomuto certifikátu:

```
[mysqld]
ssl-cert=/path/to/ssl/server-cert.pem
ssl-key=/path/to/ssl/server-key.pem
```

Každá komponenta je registrována v Keystone a má vlastní uživatelské jméno a heslo, kterým se k němu připojuje. Toto heslo je (zašifrovaně) uloženo v databázi Keystone a v čisté podobě v konfiguračním souboru dané komponenty. Kromě toho většina z nich používá databázi MySQL, ke které má vlastní uživatelské jméno, heslo i databázi.

4.16 Aktualizace systému

Jak bylo naznačeno v předchozích kapitolách, OpenStack je velmi živě vyvíjený projekt. Vývojový plán zahrnuje každých 6 měsíců vydání nové hlavní (major) verze. Krom toho vychází také drobné aktualizace v rámci aktuální verze, které je nutné instalovat.

Drobné aktualizace nejsou problémem a jejich instalace je možná s minimálním přerušením služeb, což si autor ověřil v průběhu prací na praktické části.

Větším problémem je přechod následující hlavní (major) verzi, která kromě přidávání nových vlastností často zasahuje do současné struktury. To má za následek nutnost úpravy konfiguračních souborů apod.

Vzhledem ke složitostí celého systému je velmi důrazně doporučeno [37], aby si administrátor cloudu připravil kopii ostrého prostředí, na něm otestoval proveditelnost aktualizace a teprve poté provedl aktualizaci provozního cloudu s odstávkou služeb.

[17] Virtual Private Network

4.17 Možnosti přístupu

OpenStack poskytuje několik způsobů, jak ho spravovat:

- Webový portál Horizon.
- REST API jednotlivých služeb, které je možné využívat více způsoby:
  - Přístup z libovolné vlastní aplikace.
  - Konzolové programy v Pythonu, které jsou součástí distribuce.
  - Knihovny v Pythonu, které jsou součástí distribuce a které lze použít pro usnadnění přístupu z vlastních Python programů.

Předpokladem pro všechny metody přístupu je platný uživatelský účet v Keystone.

5 Realizace pilotního nasazení

Pro ověření realizovatelnosti navržené architektury bylo uskutečněno pilotní nasazení, během kterého byla ověřena jak funkčnost navrženého celku, tak i kompatibilita s existujícím prostředím na ZČU.

Pilotní nasazení si neklade za cíl být přesnou a kompletní realizací návrhu, nýbrž spíše konceptem, který se bude dále rozvíjet. I vzhledem k omezeným možnostem z hlediska dostupného hardwarového vybavení bylo nutné při nasazení přistoupit k některým kompromisům. Všechny odchylky od návrhu jsou v textu práce zmíněny a zdůvodněny.

5.1 Testovací hardware

Pro pilotní nasazení OpenStacku byly ze zdrojů CIV poskytnuty tři servery. První fyzický server (cloud-101) byl použit na umístění všech kontrolních součástí OpenStacku. Další dva servery (cloud-102 a cloud-001) posloužily pro hostování vytvořených virtuálních strojů.

Každý server má dvě síťové karty, první je připojen do Internetu a druhá slouží pro vnitřní komunikaci mezi servery. Obě rozhraní jsou připojeny na gigabitový switch.

	cloud-101	**cloud-102**	**cloud-001**
Účel	Controller	Compute	Compute
Veřejná IP	147.228.253.111	147.228.253.112	147.228.253.11
Privátní IP	192.168.1.111	192.168.1.112	192.168.1.11
Procesor	1× Intel E5-2420	2× Intel X5560	2× Intel 5150
Jádra/vlákna	6/12	4/8	2/2
Paměť RAM	16 GB	32 GB	8 GB
Disk	2× 300 GB	2× 300 GB	2× 250 GB

Tabulka 5.1: Parametry hardware pro pilotní nasazení

Zdroj: vlastní zpracování

5.2 Software

Pro provoz je využita standardní provozní instalace operačního systému na CIV, která je založena na Debian 7.

Ačkoliv je jako doporučená distribuce pro nasazení OpenStacku označováno Ubuntu LTS, použitý Debian patří rovněž mezi podporovaný a při nasazení nenastaly žádné potíže pramenící z volby této distribuce.

Pro pilotní nasazení byl použit OpenStack ve verzi Havana, která byla v době nasazení (listopad 2013) aktuální.

5.2.1 Základní konfigurace

Všechny servery, na kterých budou provozovány komponenty OpenStacku, musí být připraveny identickým výchozím způsobem. Zahrnuje to především instalaci NTP[1], díky kterému se bude čas v rámci clusteru správně synchronizovat. Je doporučeno, aby čas zvenčí přebíral pouze jeden centrální server a všechny ostatní přebíraly čas od něj. Toho lze docílit úpravou souboru `/etc/ntp.conf`:

```
server <IP_CENTRÁLNÍHO_SERVERU>
```

Ostatní servery z konfigurace odstraníme. Dalším krokem je přidání oficiálních repozitářů do `/etc/apt/sources.list`:

```
deb http://archive.gplhost.com/debian havana-backports main
deb http://archive.gplhost.com/debian havana main
```

Poté je nutné aktualizovat lokální databázi a nainstalovat klíč repozitáře:

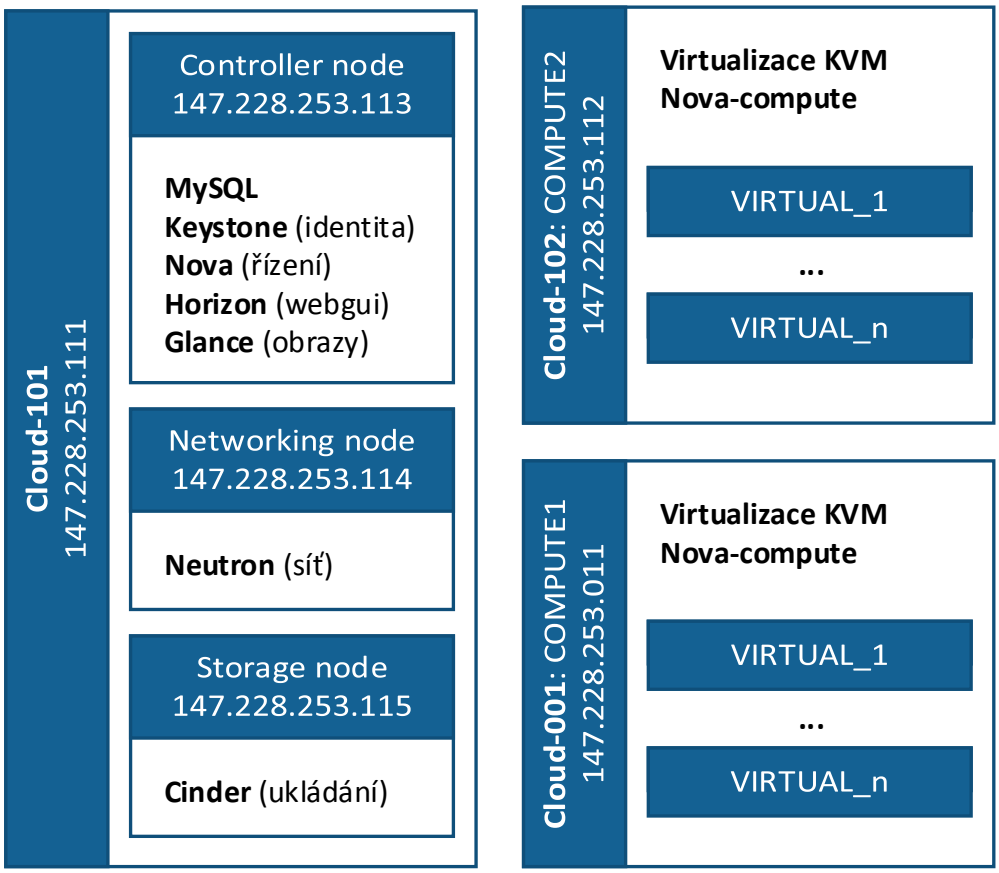
```
# apt-get update && apt-get install gplhost-archive-keyring
```

Dále je nutné nainstalovat balíky `python-mysqldb` a `python-argparse`.

[1] Network Time Protocol

5.3 Rozložení komponent

Jak již bylo zmíněno dříve, OpenStack je tvořen několika samostatnými komponentami. Tyto je vhodné v ostrém provozu umístit na samostatné servery, nicméně vzhledem k omezenému množství fyzických serverů pro pilotní provoz bylo nutné řídící servery virtualizovat, jak ilustruje obr. 5.1.



Obrázek 5.1: Rozložení služeb v realizované architektuře

Zdroj: vlastní zpracování

Technicky by sice bylo možné všechny služby nainstalovat na jeden server, avšak realizované řešení je praktičtější z hlediska budoucího rozvoje – jednotlivé virtualizované servery lze snadno přesunout na vlastní fyzický server nebo jim dle potřeby navýšit zdroje (paměť či CPU).

V pilotním provozu bylo upuštěno od řešení vysoké dostupnosti, jehož návrh byl detailně popsán v kap. 4.11. Autor doporučuje zvážit implementaci HA v ostrém provozu, ačkoliv vzhledem k projektovanému využití je možné se spokojit se současným modelem typu best-effort[2].

5.4 Controller node

Řídící uzel tvoří zásadní část celé infrastruktury. Jako první byla na server nainstalována centrální databáze MySQL, na které závisí všechny ostatní služby:

```
# apt-get install python-mysqldb mysql-server
```

Instalace balíčku neprovede inicializaci databáze, což je nutné provést ručně. Zároveň je vhodné ihned nastavit heslo správce (root), což se realizuje programem `mysql_secure_installation`:

```
# mysql_install_db
# mysql_secure_installation
```

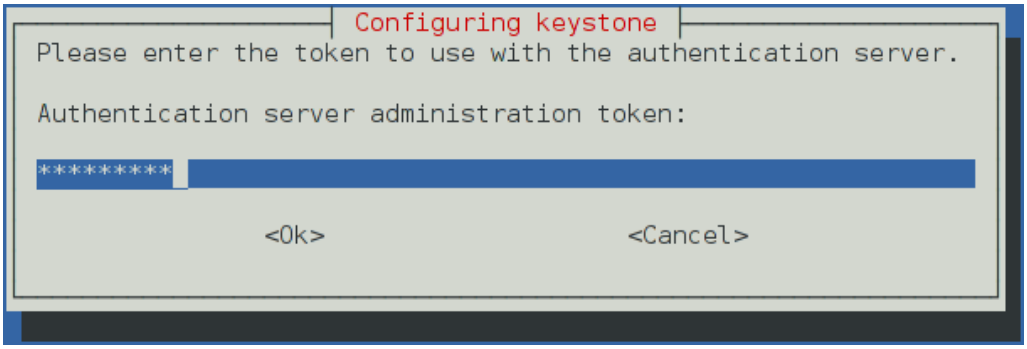
Druhý v pořadí byl následoval server pro výměnu zpráv v rámci clusteru RabbitMQ, kterému je vhodné ihned po instalaci změnit uživatelské heslo:

```
# apt-get install rabbitmq-server
# rabbitmqctl change_password guest NOVÉ_HESLO
```

Hned poté následoval správce identit Keystone vč. konzolového klienta:

```
# apt-get install keystone python-keystoneclient
```

Postinstalační proces konfigurace probíhá přes program debconf, který nás postupně vyzve k zadání řady potřebných proměnných – např. vlastní



Obrázek 5.2: Prostředí debconf pro konfiguraci

Zdroj: [39]

administrační token, jak ukazuje obr. 5.2. Debconf také zajistí napojení jednotlivých komponent OpenStacku do registru v Keystone, čímž usnadní část konfigurace.

Následně byl na server nainstalován Glance, který spravuje databázi obrazů (images) virtuálních serverů. Glance využívá v pilotním provozu pro ukládání obrazů virtuálních serverů lokální úložiště, ale pro ostrý provoz by bylo vhodnější využít externího. V aktuální verzi OpenStacku (Havana) však není plně dokončena podpora Cinderu jako úložiště dat pro Glance[3].

```
# apt-get install glance python-glanceclient
```

Řídící komponentu výpočetního clusteru (Nova) nainstalujeme včetně všech podpůrných částí příkazem:

```
# apt-get install nova-consoleproxy nova-api \
  nova-cert nova-conductor nova-consoleauth \
  nova-scheduler python-novaclient
```

Poslední nainstalovanou komponentou je webový portál Horizon:

```
# apt-get install memcached libapache2-mod-wsgi \
  openstack-dashboard openstack-dashboard-apache
```

[2]Není garantována spolehlivost služby.

[3]Více informací o (ne)podpoře: `http://docs.openstack.org/developer/glance/configuring.html#configuring-the-cinder-storage-backend`

Konfigurační soubory pro všechny instalované komponenty jsou vzhledem ke svému rozsahu umístěny na přiloženém CD v adresáři `/config`.

5.4.1 Obrazy serverů

Jak již bylo zmíněno, komponenta Glance spravuje obrazy virtuálních serverů. V základní instalaci není obsažen žádný obraz a je nutné si obrazy nejprve naimportovat. Jsou podporovány následující formáty obrazů:

- raw
- vhd (VMWare, Xen, Microsoft, VirtualBox, ...)
- vmdk
- vdi (VirtualBox, QEMU)
- iso
- qcow2 (QEMU, KVM)
- aki, ari, ami (Amazon kernel, ramdisk, machine)

V rámci testovacího provozu byly na cloud nahrány následující obrazy:

- CirrOS 0.3.1, který slouží jako minimální distribuce především pro otestování správné funkčnosti.
- CentOS 6.5
- Ubuntu 12.04 LTS

Obrazy pro další operační systémy lze buď stáhnout, k čemuž nabízí dokumentace OpenStacku řadu odkazů[4], nebo vytvořit svépomocí podle velmi podrobného návodu z dokumentace OS[5].

Import může probíhat buď přes portál nebo pomocí konzolového klienta:

[4]Detaily na: `http://docs.openstack.org/image-guide/content/ch_obtaining_images.html`

[5]Více informací na: `http://docs.openstack.org/image-guide/content/ch_creating_images_manually.html`

```
# wget -O cirros.img http://download.cirros-cloud.net/0.3.2/\
  cirros-0.3.2-x86_64-disk.img
# glance image-create --name=CirrOS --disk-format=qcow2 \
  --container-format=bare --is-public=true < cirros.img
```

Glance přes výše uvedené formáty podporuje širokou paletu operačních systému – Linux, Windows, FreeBSD, ...

Je dostupná kontextualizace obrazů ve formě nahrání uživatelských SSH klíčů při vytváření. Je nutné, aby měl uživatel předem připravený svůj klíč a veřejnou část nahrál přes portál nebo API.

5.5 Storage node

Uzel s diskovým úložištěm byl realizován jako samostatný virtuální server. Cinder pracuje přímo s interním úložištěm (v obrázku 5.3 /dev/hda), které dále rozděluje pomocí LVM a tyto svazky propojuje s compute node (Nova) pomocí iSCSI.

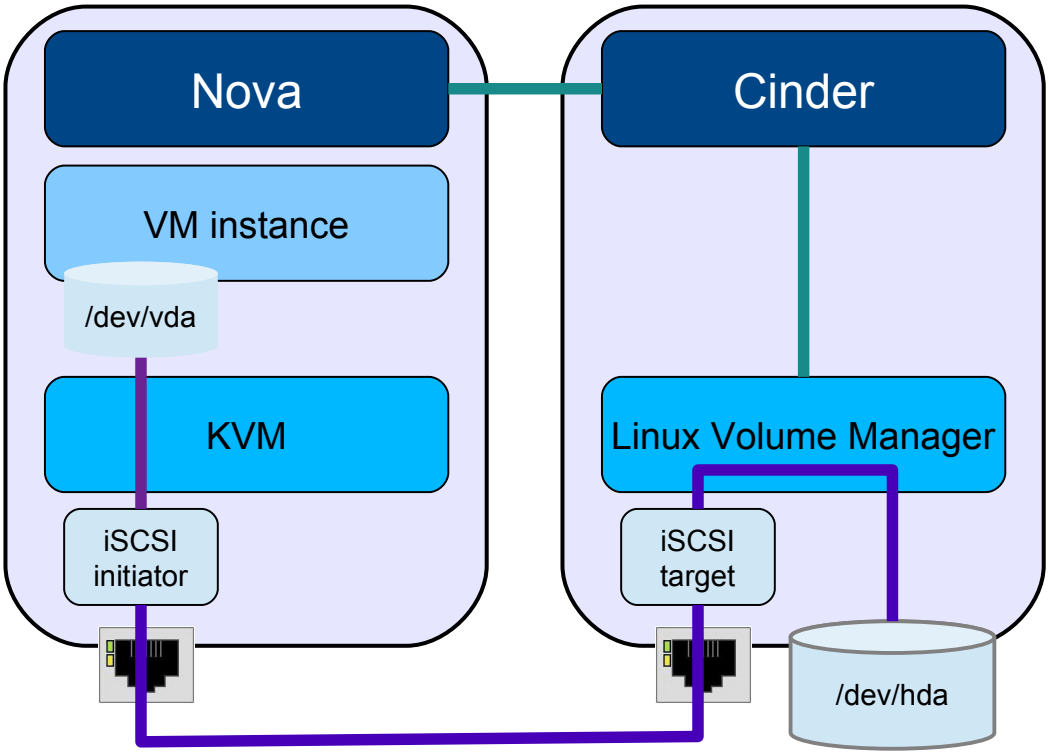
Prvním krokem bylo vytvoření LVM na serveru (/dev/xvdb je lokální disk) a vytvoření svazku (VG) se jménem *cinder-volumes*, na kterém pak bude Cinder vytvářet logické svazky (LV):

```
# pvcreate /dev/xvdb
# vgcreate cinder-volumes /dev/xvdb
```

Po základní instalaci (viz kap. 5.2.1) byly na server nainstalovány potřebné balíky:

```
# apt-get install cinder-api cinder-scheduler cinder-volume
```

Konfigurační soubor je vzhledem ke svému rozsahu umístěn na přiloženém CD v adresáři `/config`.



Obrázek 5.3: Storage server s interním uložištěm

Zdroj: [52]

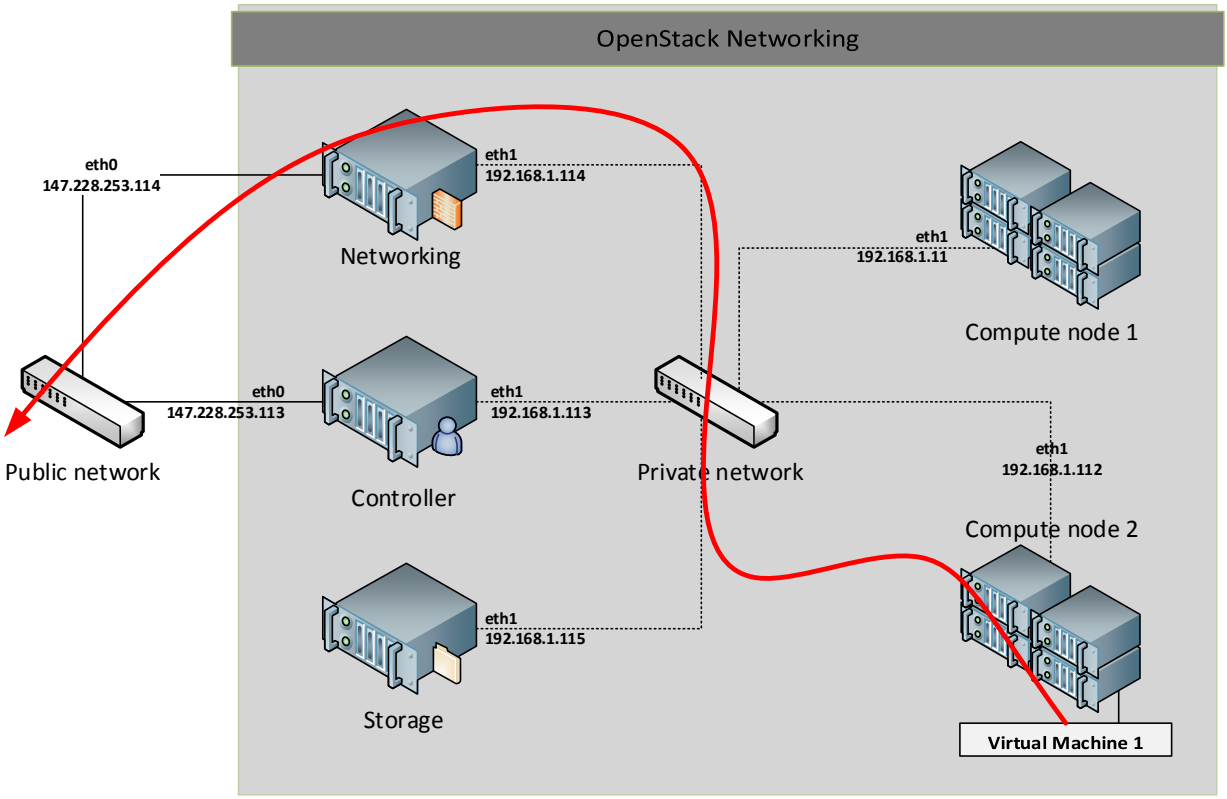
5.6 Networking node

Síťový uzel tvoří rozhraní mezi internetem a privátní sítí, na kterou jsou umístěné (nejen) samotné virtuální servery. Prochází jím tedy veškerá komunikace virtuálních serverů s vnějším světem. Je umístěn na samostatném virtuálním serveru, protože jde o komponentu, kde se očekává vyšší zátěž při provozu.

Obrázek 5.4 zachycuje navržené schéma sítě. Veškerý datový tok z virtuálního serveru prochází přes interní síť a síťový uzel (IP 192.168.1.114).

Při pilotním nasazení bylo upuštěno od realizace varianty s GRE tunely a VLANy. Realizavný pilotní provoz pracuje s FLAT modelem sítě, kde nejsou jednotlivé virtuální servery izolovány.

Po úvodní přípravě serveru (kap. 5.2.1) nainstalujeme balíky komponenty



Obrázek 5.4: Schéma sítě a zakreslení toku dat z virtuálního serveru

Zdroj: vlastní zpracování

Neutron:

```
# apt-get install neutron-server neutron-dhcp-agent \
  neutron-plugin-openvswitch-agent neutron-l3-agent
```

Dále je nutné upravit soubor `/etc/sysctl.conf`, aby síťový uzel mohl řídit provoz jednotlivých instancí:

```
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

K propagaci změn v souboru použijeme příkaz `sysctl -p`. Konfigurační soubor je vzhledem ke svému rozsahu umístěn na přiloženém CD v adresáři `/config`.

5.7 Compute node

V rámci pilotního provozu byly nainstalovány a připraveny dva výpočetní uzly (cloud-102 a cloud-001, viz. tabulka 5.1), oba s identickou softwarovou konfigurací. Po základní přípravě (kap. 5.2.1) provedeme instalaci potřebných balíků:

```
# apt-get install nova-compute-kvm python-guestfs \
  openstack-compute-node
```

Každý další výpočetní uzel se přidá tak, že po provedení výše uvedeného postupu z libovolného existujícího výpočetního nebo řídícího uzlu překopírujeme soubor `/etc/nova/nova.conf`, ve kterém upravíme IP adresu u položek `my_ip`, `vncserver_listen` a `vncserver_proxyclient_address`.

Konfigurační soubor je vzhledem ke svému rozsahu umístěn na přiloženém CD v adresáři `/config`.

5.8 Ceilometer

Instalaci komponenty pro měření využívání zdrojů je nutné provést ve více krocích. Hlavní část komponenty bude umístěna na Controller node. Ceilometer nepodporuje v současné době relační databází MySQL a proto je nutné nainstalovat MongoDB[6], což je jediná podporovaná databáze.

Standardní repozitáře Debianu obsahují velmi zastaralou verzi a proto je nutné před instalací do systému přidat oficiální repozitář MongoDB:

```
# apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
```

[6] `http://www.mongodb.org/`

```
# echo 'deb http://downloads-distro.mongodb.org/repo/debian-sysvinit \
  dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
# apt-get update
# apt-get install mongodb-org
```

V konfiguračním souboru `/etc/mongod.conf` upravíme tyto hodnoty, zbytek necháme výchozí:

```
bind_ip = 127.0.0.1
smallfiles = true
```

Server spustíme příkazem `/etc/init.d/mongod start`, připojíme se k němu a založíme uživatele:

```
# mongo
> use ceilometer
> db.addUser( { user: "ceilometer",
              pwd: "CEILOMETER_DB_HESLO",
              roles: [ "readWrite", "dbAdmin" ]
            } )
```

Dále již můžeme nainstalovat samotný Ceilometer:

```
# apt-get install ceilometer-api ceilometer-collector \
  ceilometer-agent-central python-ceilometerclient
```

Jeho konfigurace již není automatická jako u ostatních komponent. Budeme potřebovat vygenerovat `TAJNÝ_KLÍČ`, který slouží pro podepisování komunikace mezi uzly Ceilometeru a také vytvořit nový servisní účet v Keystone (uživatel *ceilometer*, tenant *service* a náhodné heslo).

Poté ručně upravíme soubor `/etc/ceilometer/ceilometer.conf`:

```
[database]
connection = mongodb://ceilometer:CEILOMETER_DB_HESLO@ \
  127.0.0.1:27017/ceilometer
```

```
[publisher_rpc]
metering_secret = TAJNÝ_KLÍČ

[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000
admin_tenant_name = service
admin_user = ceilometer
admin_password = CEILOMETER_SERVICE_HESLO

[service_credentials]
os_username = ceilometer
os_tenant_name = service
os_password = CEILOMETER_SERVICE_HESLO
```

Pro aplikaci nového nastavení služby restartujeme:

```
# service ceilometer-agent-central restart
# service ceilometer-api restart
# service ceilometer-collector restart
```

5.8.1 Úpravy v komponentě Cinder

Do souboru `/etc/cinder/cinder.conf` přidáme následující řádky:

```
[DEFAULT]
...
control_exchange = cinder
notification_driver = cinder.openstack.common.notifier.rpc_notifier
```

Pro načtení změn je nutný restart služeb:

```
# service cinder-volume restart
# service cinder-api restart
```

5.8.2 Úpravy v komponentě Glance

Do souboru `/etc/glance/glance-api.conf` přidáme následující řádky (za `RABBIT_HESLO` dosadíme heslo, které jsme nastavili v kap. 5.4):

```
[DEFAULT]
...
notifier_strategy = rabbit
rabbit_host = controller
rabbit_password = RABBIT_HESLO
```

Pro načtení změn je nutný restart služeb:

```
# service glance-registry restart
# service glance-api restart
```

5.8.3 Úpravy v komponentě Nova

Pro měření zdrojů na výpočetním uzlu musíme nejprve nainstalovat službu, která ho zprostředkuje:

```
# apt-get install ceilometer-agent-compute
```

Soubor `/etc/nova/nova.conf` upravíme následovně:

```
[DEFAULT]
...
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
notification_driver = nova.openstack.common.notifier.rpc_notifier
notification_driver = ceilometer.compute.nova_notifier
```

Upravíme také soubor `/etc/ceilometer/ceilometer.conf`:

```
[publisher_rpc]
metering_secret = TAJNÝ_KLÍČ

[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000
admin_tenant_name = service
admin_user = ceilometer
admin_password = CEILOMETER_SERVICE_HESLO

[service_credentials]
os_username = ceilometer
os_tenant_name = service
os_password = CEILOMETER_SERVICE_HESLO
```

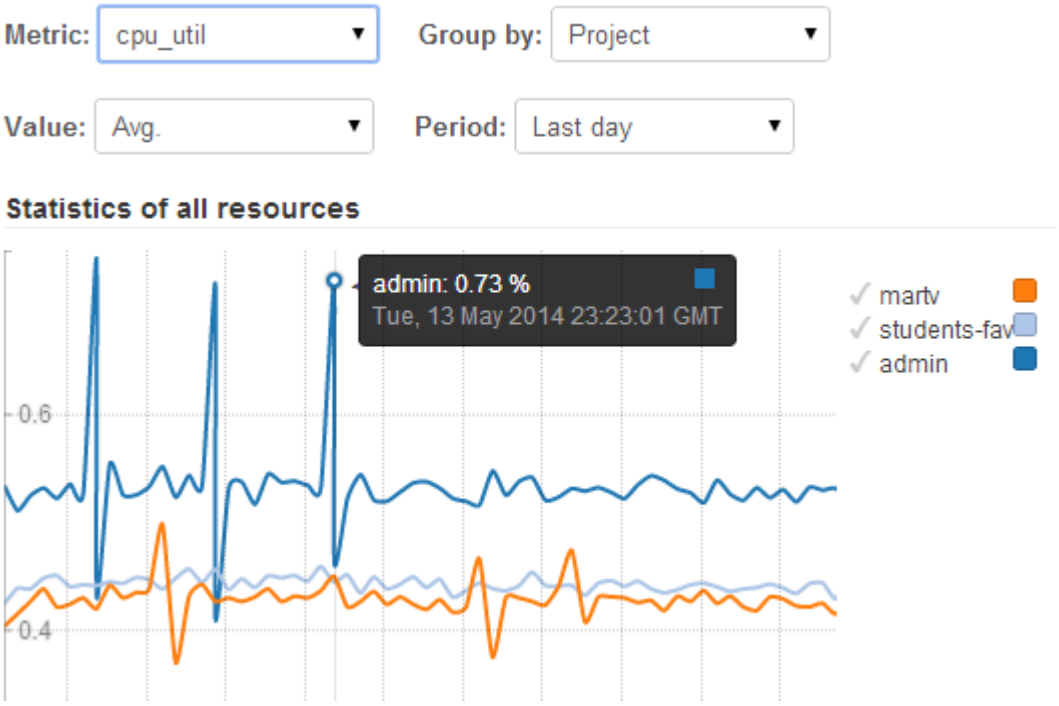
Pro načtení změn službu restartujeme:

```
# service ceilometer-agent-compute restart
```

Výsledkem je funkční komponenta, která sbírá data o využití uvedených služeb. Příkladem je obr. 5.5, který zachycuje zobrazení výstupů v portálu Horizon.

5.9 Integrace do prostředí ZČU

Při zapojení do existující infrastruktury ZČU bylo nutné se vypořádat především se správou uživatelských účtů a přihlašováním ke službě. Jak uvádí [21], doba, kdy každá aplikace přináší svůj vlastní způsob ověřování, se již pomalu stává minulostí. Bylo tedy nutné provést úpravy vedoucí k zapojení systému do Single-Sign-On architektury, aby bylo zajištěno pohodlné využívání uživateli bez nutnosti pamatovat si další hesla.



Obrázek 5.5: Výstup měření zátěže CPU v portálu Horizon

Zdroj: vlastní zpracování

5.9.1 Digitální certifikát

Aby bylo možné k webovému portálu a ostatním službám přistupovat zabezpečeně, je použit protokol TLS[7], což je kryptografický protokol zajišťující šifrovanou komunikaci po počítačové síti. Tento protokol umožňuje aplikacím typu klient/server komunikovat takovým způsobem, který zabraňuje odposlechu nebo padělání zpráv. [13]

Protokol využívá asymetrickou kryptografii, přičemž ale není zaručena identita druhé strany. Z tohoto důvodu existuje infrastruktura veřejného klíče (Public Key Infrastructure, PKI), která pro zajištění důvěryhodnosti využívá certifikační autority (CA). PKI je struktura, která se skládá z hardwaru, softwaru, osob, politik a procedur potřebných pro vytváření, správu, ukládání, distribuci a zneplatňování digitálních certifikátů. [55]

Pro vydání vlastního digitálního certifikátu je nutné vygenerovat privátní

[7]Transport Layer Security

klíč (zůstává utajen) a žádost (Certificate signing request, CSR). Tuto žádost poté předáme námi zvolené CA, která po ověření identity subjektu, pro který má být certifikát vystaven, vydá digitální certifikát.

Certifikačních autorit je celá řada, přičemž pro bezproblémový zabezpečený přístup z klientských počítačích je nutné, aby byl kořenový certifikát zvolené CA umístěn mezi důvěryhodnými na daném počítači.

Pro projektované akademické využití bylo možné využít serverové certifikáty TERENA Certificate Service (TCS), které jsou vydávány pod hlavičkou důvěryhodné CA COMODO pro akademické insitituce zapojené do sítě CESNET2. [8]

V rámci této práce byl vydán certifikát pro `https://cloud.civ.zcu.cz`.

5.9.2 Kerberos a WebAuth

Jelikož je přístup k aplikaci podmíněn existencí Orion konta v rámci ZČU, je pro ověření identity uživatele využit systém WebAuth založený na autentizačním systému Kerberos. Tento přístup eliminuje nutnost zakládat samostatné účty pouze pro účel OpenStacku s vlastním heslem a tak jde návrh v duchu Single-Sign-On (SSO), kde si uživatel vystačí s přihlášením k více službám na jednom místě. Princip fungování zachycuje obr. 5.6.

Technicky je fungování zajištěno přes modul *mod_webauth* do webového serveru Apache, který je vyvíjen na Stanfordově univerzitě[9]. Pro Debian existuje připravený balíček, instalace je tedy triviální a nainstalují se i všechny potřebné závislosti (zejména knihovny pro Kerberos):

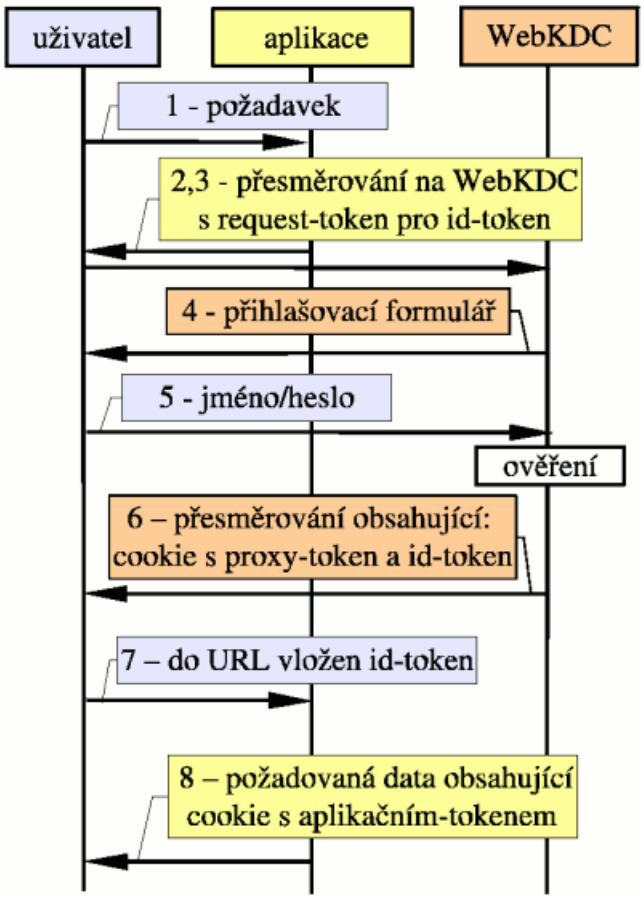
```
apt-get install libapache2-webauth
```

Modul (`/etc/apache2/mods-available/webauth.conf`) je nakonfigurován následujícím způsobem:

```
WebAuthLoginURL "https://webkdc.zcu.cz/login.fcgi"
```

[8] `http://support.zcu.cz/index.php/Z\%C3\%ADsk\%C3\%A1n\%C3\%AD_serverov\%C3\%A9ho_certifik\%C3\%A1tu_TCS_COMODO_pro_webov\%C3\%BD_server`

[9] `http://webauth.stanford.edu/`



Obrázek 5.6: Prvotní přihlášení uživatele

Zdroj: [21]

```
WebAuthWebKdcURL "https://webkdc.zcu.cz/webkdc-service/"
WebAuthWebKdcPrincipal webkdc/webkdc
WebAuthKeyring /etc/webauth/keyring
WebAuthKeyringAutoUpdate on
WebAuthKeyringKeyLifetime 30d
WebAuthKeytab /etc/webauth/keytab
WebAuthServiceTokenCache /etc/webauth/service_token.cache
```

V souboru `/etc/webauth/keyring` se nachází klíč Kerberos principalu aplikačního serveru.

Dále je nutné vyžádat autentizaci pro portál, což provedeme přidáním těchto řádků do souboru `openstack-dashboard-ssl.conf`, který najdeme v

adresáři `/etc/apache2/sites-available/`:

```
<Location />
    AuthType WebAuth
    Require valid-user
</Location>
```

Tímto je povolen přístup všem autentizovaným držitelům Orion konta.

5.9.3 Uživatelské účty

Základní identitou sytému oprávnění v OpenStacku je *uživatel* (user), který má přidělené heslo. Uživatel může patřit do jednoho nebo více *projektů* (tenants) v definované *roli* (uživatel, administrátor, ...).

Kvóty

Pro každý *projekt* se definují kvóty, zejména pro:

- Velikost diskových jednotek v GB.
- Počet vytvořených diskových jednotek.
- Počet spuštěných instancí serverů.
- Počet alokovaných jader procesoru.
- Počet alokovaných IP adres.

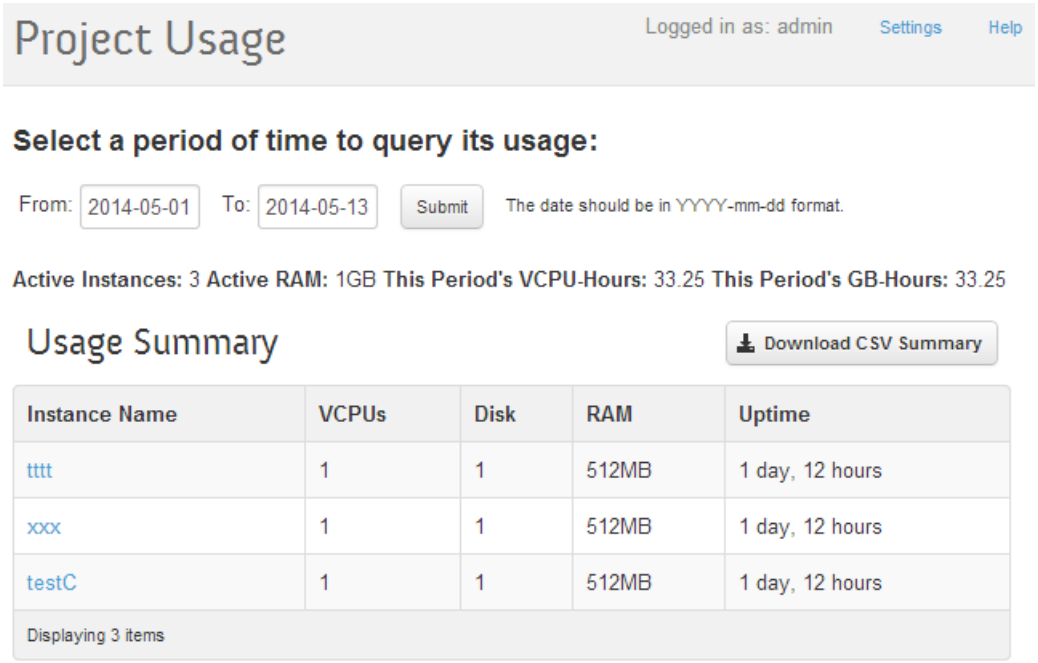
Na nově založený projekt se aplikují defaultní kvóty, které může upravit administrátor buď přes portál (Admin → Defaults → Update Defaults) nebo přes konzolový příkaz:

```
nova quota-class-update --instances 2 default
```

Výše uvedeným příkazem nastavíme defaultní limit na 2 vytvořené instance serveru. Výpis všech kvót provedeme příkazem:

```
nova quota-defaults
```

Přidělené kvóty lze projektům kdykoliv upravit buď přes portál (Admin → Projects → More → Modify Quotas) nebo konzolovým programem. Je možné také sledovat využití kvót a získat základní představu o využívání zdrojů konkrétním projektem. Výpis běžících instancí a souhrnné statistiky (počet instancí, celková velikost RAM a počet alokovaných „procesorohodin“ za zvolený časový úsek) je dostupný také přes web (Admin → Projects → More → View Usage), příklad výpisu ilustruje obr. 5.7.



Obrázek 5.7: Zobrazení využití zdrojů projektem

Zdroj: vlastní zpracování

Uživatelé na ZČU

Pro řízení uživatelských účtů máme k dispozici dvě možnosti. První možností je vytvořit pro každého uživatele vlastní projekt, kterému nastavíme limity.

Druhým přístupem je uživateli nezakládat automaticky vlastní projekt, ale zařadit ho do existujících větších celků.

Typickým scénářem může být situace, kdy budeme chtít uživatele sdružovat do projektů podle fakulty a katedry. Můžeme k tomu využít univerzitní LDAP[10] databázi, která poskytuje základní informace o studentovi:

```
# ldapsearch -x -H ldap://ldap.zcu.cz uid=martv
...(vynechané řádky)...
memberof: cn=stud-kat-kiv,ou=Groups,ou=rfc2307,o=zcu,c=cz
memberof: cn=students-fav,ou=Groups,ou=rfc2307,o=zcu,c=cz
```

O uživateli *martv* jsme schopni zjistit, že studuje na FAV[11] a má zapsaný libovolný předmět z KIV[12]. Více informací nám LDAP bohužel neposkytne a pokud bychom chtěli sdružovat uživatele podle přesnějších kritérií, například podle předmětů, které aktuálně studuje, musíme využít webovou službu univerzitního systému IS/STAG[13].

Pro zjištění aktuálně zapsaných předmětů uživatele potřebujeme nejprve zjistit jeho osobní číslo. To provedeme zavoláním webové služby a předáním uživatelského jména jako parametr:

```
https://stag-ws.zcu.cz/ws/services/rest/orion/
   getOsobniCislaByOrionLogin?login=martv
```

Webová služba nám vrátí osobní číslo (`<osCislo>A11N0016K</osCislo>`), se kterým můžeme dále pracovat a vyžádat si seznam předmětů, které má student v aktuálním semestru zapsané:

```
https://stag-ws.zcu.cz/ws/services/rest/predmety/
   getPredmetyByStudent?osCislo=A11N0016K&semestr=ZS
```

[10] Lightweight Directory Access Protocol
[11] Fakulta aplikovaných věd
[12] Katedra informatiky a výpočetní techniky
[13] Kompletní dokumentace k dispozici na: `https://stag-ws.zcu.cz/ws/help/`

Realizace napojení

Jak již bylo zmíněno, Keystone podporuje i externí autentizační služby a bylo by možné ho přímo napojit na Kerberos, který je na ZČU využíván.

Tento postup by však znamenal, že by uživatelé museli zadávat své přihlašovací údaje přímo do rozhraní OpenStacku, což není považováno zadavatelem za ideální v tom směru, že si nepřeje, aby se takto citlivé údaje zadávaly na přímo do aplikací třetích stran.

Bylo tedy nutné využít již zmíněný WebAuth. Na straně Keystone ale vznikl problém, protože jeho uživatelská databáze není nijak napojena na univerzitní seznam uživatelů. Takže i přes úspěšnou WebAuth autentizaci nebyl uživatel přihlášen, protože v databázi Keystone neexistoval jeho účet.

Tento problém je řešitelný několika způsoby. Jedním z nich je periodická hromadná aktualizace databáze Keystone tak, aby obsahovala všechny Orion účty na ZČU. Vzhledem k předpokladu, že OpenStack bude využívat pouze malý zlomek držitelů Orion konta (kterých je nyní cca. 20 tisíc), není ideální mít v databázi všechny oprávněné uživatele kvůli vyšší přehlednosti a jednoduchosti správy.

Autor práce navrhl a realizoval vlastní způsob spočívající ve vytvoření skriptu, který po úspěšném přihlášení přes WebAuth připraví v Keystone uživatelský účet a uživatele automaticky přihlásí na Horizon portál, který nativně neumí pracovat s WebAuthem. Výhodou je, že uživatelská databáze Keystone bude obsahovat jen osoby, které se k OpenStacku alespoň jednou přihlásily a odpadá nutnost synchronizace databází uživatelů.

Postup fungování skriptu je následující:

1. Uživatel *OrionUser* se úspěšně přihlásí přes WebAuth.
2. Skript ověří, jestli uživatel již existuje v databázi Keystone a vygeneruje jednorázové heslo pro vstup (toto heslo je pouze interní, uživatel ho nikde nevidí ani ho nezná).
   (a) Uživatel neexistuje: skript založí nový projekt *OrionUser* a uživatele *OrionUser* s vygenerovaným jednorázovým heslem.
   (b) Uživatel již existuje: skript nastaví existujícímu uživateli vygenerované jednorázové heslo.

3. Skript si z LDAP zjistí, jakých skupin je uživatel *OrionUser* členem. Z databáze Keystone zjistí, jaké projekty (tenants) jsou v OpenStacku založeny. Pokud je nalezena shoda (např. uživatel *OrionUser* je členem LDAP skupiny *students-fav* a v OpenStacku existuje tento projekt), je uživateli přidána role v tomto projektu.

4. Uživatel je automaticky přihlášen a přesměrován na hlavní stránku portálu.

Uživatel má tedy kromě primárního projektu automaticky členství ve všech existujících projektech, které se jménem shodují se skupinami, jejíchž je v LDAP členem. Pro správné fungování je nutné, aby byly projekty odpovídající skupinám v LDAP nejprve ručně založeny.

Pokud budeme chtít povolit zakládání virtuálních strojů jen vybraným uživatelům, lze to realizovat více způsoby. Jedním z nich je nastavit nulové výchozí kvóty (dle kap. 5.9.3) a potřebné kvóty nastavit až projektům, do kterých budou uživatelé automaticky přidáni na základě svého členství v LDAP.

Druhou variantou je omezení přístupu k portálu na úrovni WebAuth, které lze realizovat úpravou souboru `openstack-dashboard-ssl.conf` v adresáři `/etc/apache2/sites-available/` (podrobnější popis v kap. 5.9.2):

```
<Location />
    AuthType WebAuth
    #Require valid-user
    Require privgroup civ
</Location>
```

Výše uvedený příklad povolí přístup jen Orion uživatelům ve skupině CIV.

Vytvořený skript je napsán v jazyce Python, takže pro běh nevyžaduje žádné další softwarové vybavení než je nutné pro běh OpenStacku (který je také naprogramován v Pythonu). Pro funkčnost nicméně vyžaduje dva drobné zásahy do rozhraní portálu Horizon, pro které je vytvořen patch umístěný na CD v příloze této práce. Patche použijeme tak, že je nahrajeme na řídící uzel do adresáře `/tmp/horizon_patch` a spustíme:

```
# patch /usr/share/pyshared/horizon/templates/auth/_login.html < \
```

```
  /tmp/horizon_patch/_login.html.patch
# patch /usr/share/pyshared/openstack_auth/views.py < \
  /root/horizon_patch/views.py.patch
```

5.10 Ověření

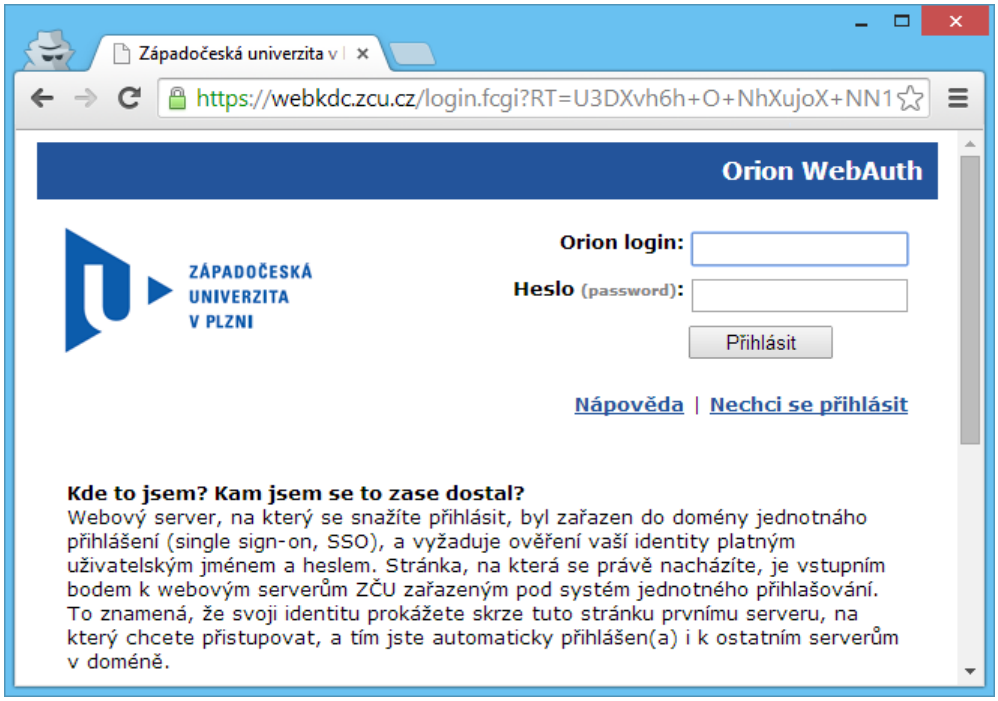
Cílem ověření je prokázat, že realizovaný pilotní provoz splňuje definované požadavky a představuje základ, díky kterému je možné dále rozvíjet realizovaný projekt.

5.10.1 Scénář: založení a přihlášení uživatele

Požadavky: uživatel s platným univerzitním Orion kontem, internetový prohlížeč.

Cíl: prokázat, že je funkční autentizace přes WebAuth a skript zajišťující propagaci uživatelských kont v Keystone pracuje správně.

1. Uživatel přistoupí na adresu Horizon portálu (`https://cloud.civ.zcu.cz`).
2. Modul mod_webauth detekuje, že uživatel není přihlášen a přesměruje ho na stránku Orion WebAuth (obr. 5.8).
3. Uživatel zadá svůj Orion login a heslo.
4. Po úspěšném ověření identity je uživatel přesměrován zpět na Horizon.
5. Je spuštěn námi vytvořený skript, který zjistí Orion login přihlášeného uživatele.
6. Skript se spojí s Keystone a ověří, jestli již uživatel v jeho databázi existuje. Pokud v databázi ještě není, založí se. Dále si z LDAP vyžádá seznam skupin, kterých je uživatel členem. Pokud taková skupina existuje jako projekt v OpenStacku, obdrží k ní uživatel přístupová práva.
7. Uživatel je automaticky přihlášen a zobrazí se mu hlavní stránka portálu (obr. 5.9). Pokud je členem více projektů, může mezi nimi přepínat.



Obrázek 5.8: Přihlášení přes WebAuth

Zdroj: vlastní zpracování

Po odhlášení z portálu jsou korektně vymazány cookies v prohlížeči[14] a uživateli se zobrazí stránka WebAuth informující o úspěšném odhlášení.

Závěr: bylo prokázáno, že funguje uživatelský přístup k portálu a do databáze se korektně propagují informace o členství ve skupinách.

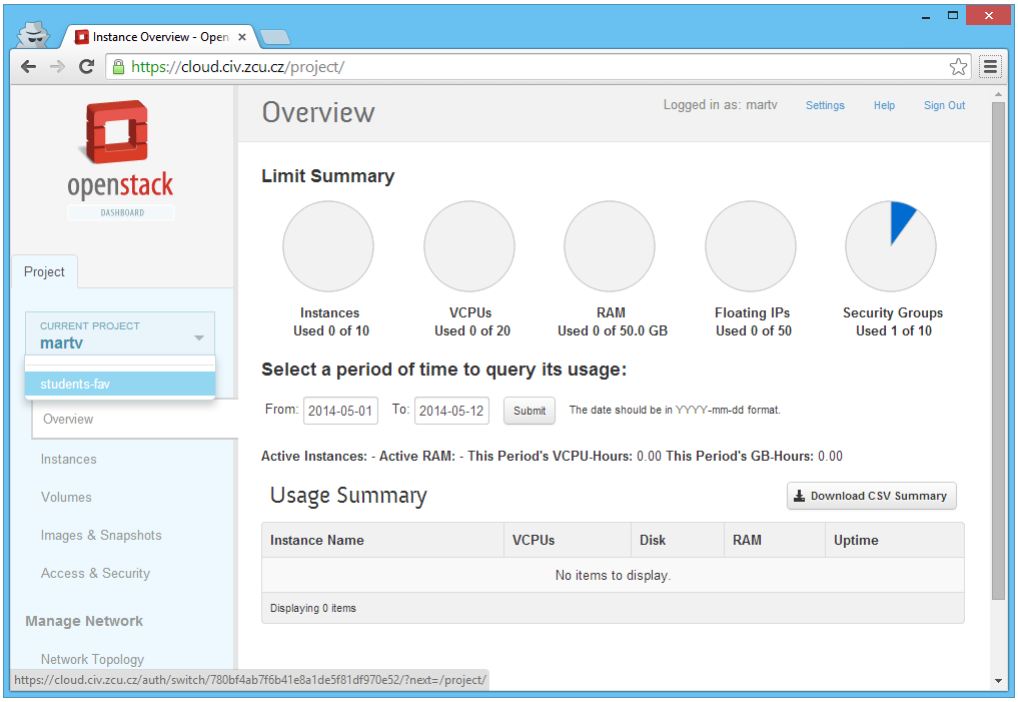
5.10.2 Scénář: založení a spuštění serveru

Požadavky: autentizovaný uživatel OpenStacku, internetový prohlížeč, založená síť v databázi OpenStacku.

Cíl: prokázat, že je možné založit a spustit virtuální server v cloudu.

1. Uživatel se přihlásí dle kap. 5.10.1. V menu se přepne na projekt, ve kte-

[14]malé množství dat, která WWW server pošle prohlížeči, který je uloží na počítači uživatele



Obrázek 5.9: Hlavní strana portálu Horizon, uživatel se může přepínat mezi projekty

Zdroj: vlastní zpracování

rém chce pracovat (pokud je členem pouze jednoho projektu, přepnutí není možné).

2. V levém menu zvolí položku INSTANCES a poté stiskne tlačítko LAUNCH INSTANCE.

3. V zobrazeném okně (obr. 5.10) vyplní detaily nově zřizovaného serveru. Je nutné zvolit název, vybrat jednu z připravených šablon parametrů (určuje počet CPU, velikost RAM a disku) a dále vybrat obraz, ze kterého bude server vytvořen. Je také možné volit mezi lokálním úložištěm (přímo na hostitelském serveru, nedoporučeno) a centrálním síťovým úložištěm (Cinder, doporučeno).

4. Uživatel volí vytvoření nové diskové jednotky na síťovém úložišti z existující šablony (Ubuntu 12.04 LTS).

5. Po stisku tlačítka LAUNCH se spustí proces popsaný v kap. 4.8 a pokud jsou splněny všechny podmínky (uživatel vyplnil všechna povinná pole,

nenarazil na horní hranici některé kvóty, . . .), výsledkem je start nového serveru v cloudu.

Závěr: bylo prokázáno, že uživatel může samostatně vytvořit a spustit virtuální server.

Testem bylo zároveň ověřeno, že funguje centrální síťové úložiště přes iSCSI a virtuální server je z něj schopen nastartovat.

5.10.3 Scénář: ověření funkčnosti kvót

Požadavky: autentizovaný uživatel OpenStacku, internetový prohlížeč, založená síť v databázi OpenStacku.

Cíl: ověřit, že funguje systém kvót.

V tomto scénáři bylo postupováno obdobně jako v kap. 5.10.2. Uživatel zakládal virtuální servery až do okamžiku, kdy narazil na definované limity zdrojů. Při pokusu přidělit serveru více paměti RAM než je dovoleno systém odmítl virtuální server založit, což dokládá obr. 5.11.

Pokud je vyčerpán limit na počet instancí, není uživateli umožněno vůbec otevřít dialogové okno pro založení serveru.

Závěr: testem bylo ověřeno, že nastavené limity v praxi fungují.

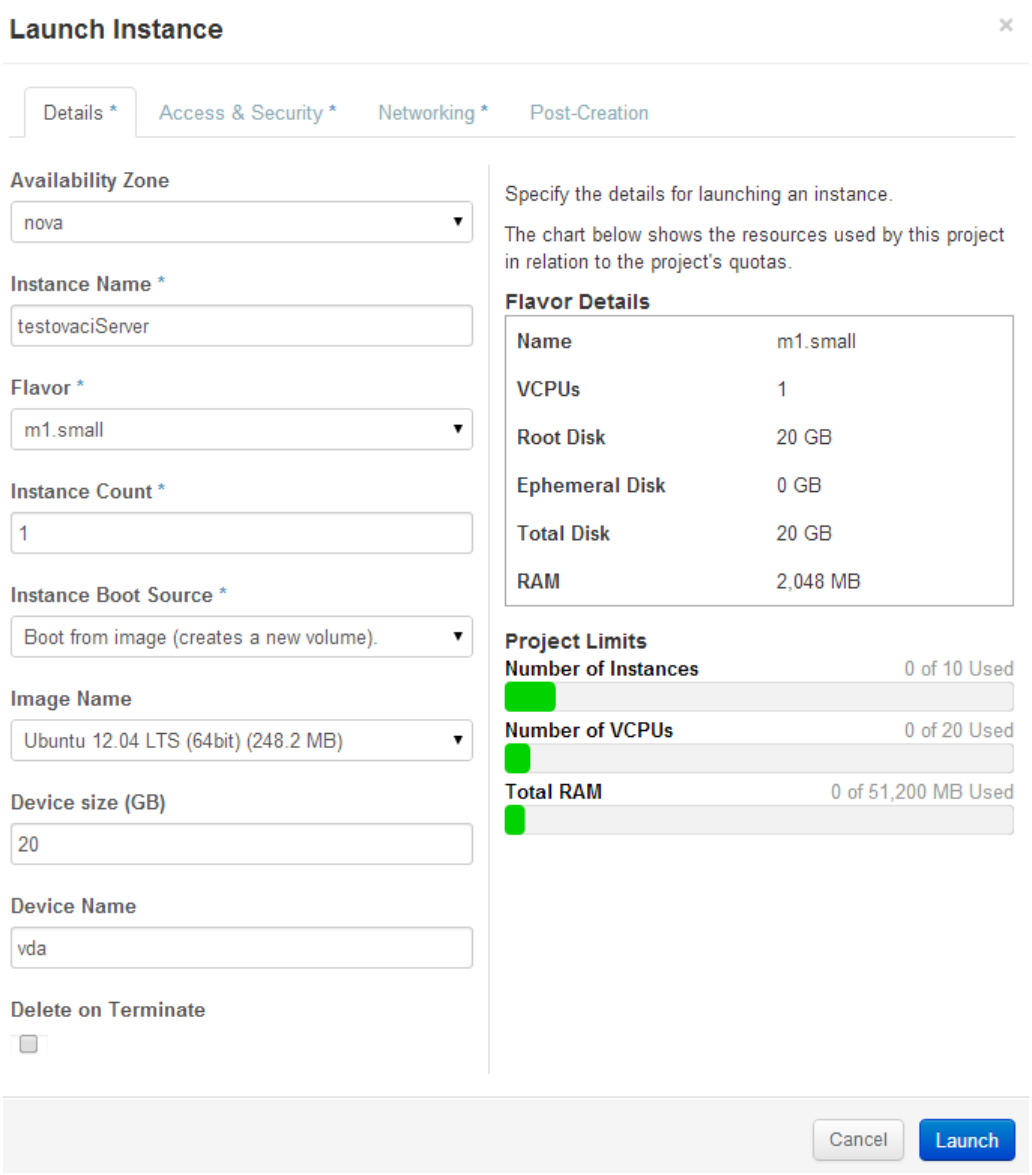
5.10.4 Scénář: spuštění více serverů najednou

Požadavky: autentizovaný uživatel OpenStacku, internetový prohlížeč, založená síť v databázi OpenStacku.

Cíl: ověřit, jak se bude chovat cloud v případě souběhu více náročných požadavků.

V tomto scénáři bylo postupováno obdobně jako v kap. 5.10.2, pouze s tím rozdílem, že bylo vytvořeno najednou 20 virtuálních serverů.

Závěr: testem bylo ověřeno vytvoření a spuštění velkého počtu instancí najednou. Test dopadl úspěšně; největší zátěž byla pozorována na uzlu, který

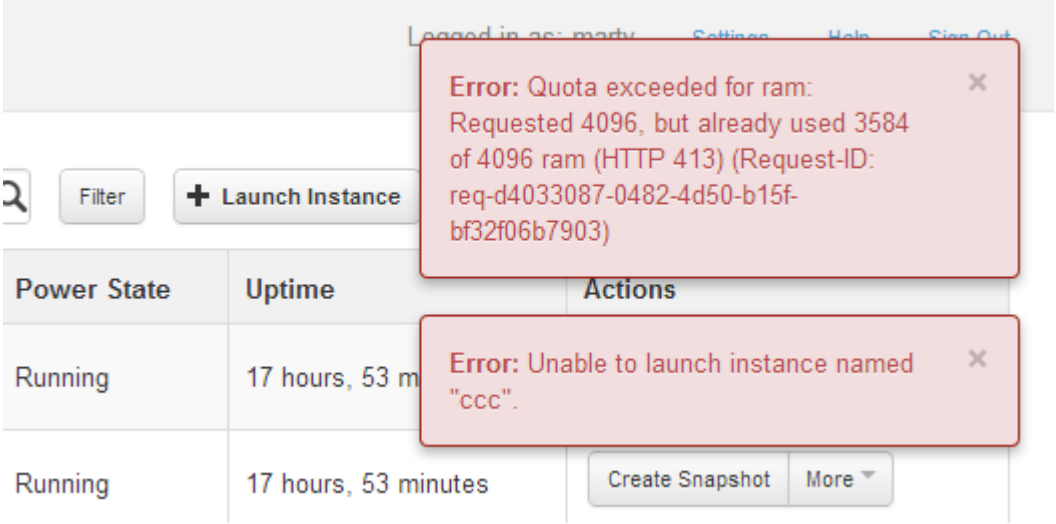


Obrázek 5.10: Volby při zakládání nového serveru

Zdroj: vlastní zpracování

měl na starost diskové úložiště.

Nejvíce výkonu vyžadovaly operace spojené s přípravou a kontextualizací obrazu serveru, vytvořením diskové jednotky a nakopírováním obrazu do nově vytvořené jednotky. Z výsledků testu plyne doporučení, že je třeba dostatečně dimenzovat výkon serveru s Glance a také samotného diskového úložiště.



Obrázek 5.11: Chybové hlášení při překročení kvót

Zdroj: vlastní zpracování

5.10.5 Naplnění požadavků

V rámci této kapitoly je proveden rozbor požadavků ze zadání a komentováno jejich splnění.

- **Požadavek**: Uzly budou plně ve správě uživatele a typicky nastartované z připraveného obrazu.

  Komentář: Splněno a prokázáno v kap. 5.10.2.

- **Požadavek**: Řešení musí splňovat provozní požadavky výpočetního prostředí Orion (monitoring, bezpečnost, zálohování, . . .).

  Komentář: Splněno, provozní částí se zabývají kapitoly 4.12 až 4.15.

- **Požadavek**: Je třeba napojení na bázi uživatelů a přístup všech uživatelů k nové službě.

  Komentář: Splněno, napojením na bázi uživatelů se zabývají kapitoly 5.9.3 a 5.9.3, prakticky je to ověřeno scénářem v kap. 5.10.1.

- **Požadavek**: Realizované řešení musí umožňovat stanovení limitů na využití zdrojů a měření využívání těchto zdrojů.

Komentář: Splněno, pro každý projekt jsou definovány výchozí kvóty (viz kap. 5.9.3), které je možné individuálně upravit. Dále je nainstalována telemetrická komponenta Ceilometer (viz kap. 5.8) umožňující měření, sběr, zpracování a vyhodnocování údajů o využití zdrojů.

- **Požadavek**: Součástí zadání není příprava obrazů virtuálních serverů, ale musí pro to být odpovídající podpora (import z jiných prostředí, podpora kontextualizace, podpora různých OS).

  Komentář: Splněno, možnostmi obrazů se zabývá kap. 5.4.1.

- **Požadavek**: Preferujeme otevřené řešení bez licenčních omezení a poplatků, modulární a založené na standardech. Důležitým parametrem je malá provozní náročnost a možnost rozšiřování (kapacita, vlastnosti).

  Komentář: Splněno, navržené řešení je svobodně dostupné pod licencí Apache 2.0. Disponuje značnou modularitou a je možno ho bez problémů rozšiřovat jak z hlediska kapacity, tak z hlediska výkonu.

6 Závěr

Cílem této práce bylo kromě seznámení s vývojem a přehledem technologií v oblasti virtualizace a cloud computingu také připravit návrh infrastruktury privátního cloudu pro potřeby Západočeské univerzity v Plzni a formou pilotního provozu ověřit jeho realizovatelnost.

Jednotlivé body ze zadání byly splněny, konkrétně byl ve spolupráci se zadavatelem vytvořen soupis požadavků, které musí navržené řešení splňovat. Byla požadována služba umožňující samoobslužné zřizování virtuálních serverů pro potřeby výuky a výzkumu, která nebude mít licenční omezení, bude využívat současné technologie a bude zapojena do prostředí Orion.

V teoretické části práce je kromě historického vývoje a přehledu dnešních technologických trendů a možností v oblastech virtualizace a cloud computingu také přehled konkrétních softwarových platforem, které lze využít pro realizaci infrastruktury privátního cloudu. Na základě zadaných kritérií jsou tyto platformy porovnány a je zvoleno vhodné řešení pro virtualizaci a také pro správu privátního cloudu.

Výstupem praktické části je popis a implementační návrh privátního cloudu na platformě OpenStack. Tento návrh je posléze realizován ve formě pilotního provozu, který probíhal od listopadu 2013 a během kterého bylo praktickými scénáři ověřeno, že je navržené řešení provozuschopné.

Je věnována pozornost aspektům běžného provozního nasazení, kdy jsou nastíněny otázky bezpečnosti, zálohování, monitorování, logování a zajištění vysoké dostupnosti. Pro splnění definovaných požadavků byly provedeny také úpravy zdrojových kódů na míru, které stejně jako všechny konfigurační soubory tvoří přílohu této práce ve formě CD.

Při realizaci pilotního provozu se vyskytla celá řada problémů, které souvisely především se složitostí OpenStacku. Ten je tvořen skupinou vzájemně kooperujících samostatných celků, které jsou mezi sebou propojené pomocí API a serveru pro frontu zpráv. Orientaci neulehčila ani dokumentace, která byla často roztříštěná a díky rychlému vývojovému cyklu plná zastaralých informací. Typicky je OpenStack nasazován v podobě komplexních komerčních distribucí nebo s podporou velkých firem, které si vytváří vlastní nástroje pro nasazení a správu. Bylo tedy velkou výzvou se o totéž pokusit vlastními silami v rámci této diplomové práce.

Seznam tabulek

Seznam obrázků

Literatura

[1] Amazon: Web Services - About AWS. [cit. 2014-02-10].
URL http://aws.amazon.com/about-aws/

[2] Armbrust, M.: Above the Clouds: A Berkeley View of Cloud Computing. 02 2009 [cit. 2014-03-03].
URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

[3] Bowen, B.; Gessau, H.; LeBlanc, D.; aj.: What's new in Neutron for OpenStack Havana. 10 2013 [cit. 2014-01-29].
URL http://www.slideshare.net/kamesh001/whats-new-in-neutron-for-open-stack-havana

[4] Brady, P.: The life of an OpenStack libvirt image. 02 2013 [cit. 2014-03-18].
URL http://www.pixelbeat.org/docs/openstack_libvirt_images/

[5] Brandon, J.: Harvard, MIT Broad Institute deploys OpenStack for private cloud. 04 2014 [cit. 2014-04-20].
URL http://www.businesscloudnews.com/2014/04/22/harvard-mit-broad-institute-deploys-openstack-for-private-cloud/

[6] Butler, B.: Open source cloud battles: OpenStack leveling off as CloudStack interest gains. 01 2013 [cit. 2014-02-28].
URL http://www.networkworld.com/news/2013/010713-openstack-cloudstack-265556.html

[7] Cannao, R.: Benchmarking NDB vs Galera. 10 2012 [cit. 2014-05-03].
URL http://www.palominodb.com/blog/2012/12/10/benchmarking-ndb-vs-galera

[8] Chellappa, R. K.: Intermediaries in Cloud-Computing: A New Computing Paradigm. In *INFORMS Annual Meeting*, 1997.

[9] Clark, C.; Fraser, K.; Hand, S.; aj.: Live Migration of Virtual Machines. In *NSDI*, USENIX, 2005.
URL http://dblp.uni-trier.de/db/conf/nsdi/nsdi2005.html#ClarkFHHJLPW05

[10] CloudStack: History. 2013 [cit. 2014-04-25].
URL https://cloudstack.apache.org/history.html

[11] Danjou, J.: Ceilometer, the OpenStack metering project. 07 2012 [cit. 2014-05-02].
URL https://julien.danjou.info/blog/2012/openstack-metering-ceilometer

[12] Dialogic: Introduction to Cloud Computing. 07 2010 [cit. 2014-01-19].
URL http://www.dialogic.com/~/media/products/docs/whitepapers/12023-cloud-computing-wp.pdf

[13] Dierks, T.; Rescorla, E.: RFC 5246: The Transport Layer Security (TLS) Protocol. 08 2008 [cit. 2014-04-12].
URL http://tools.ietf.org/html/rfc5246

[14] Eucalyptus: Cloud Computing Architecture. [cit. 2014-03-08].
URL https://www.eucalyptus.com/eucalyptus-cloud/iaas/architecture

[15] Eucalyptus: Datasheet. [cit. 2014-03-08].
URL https://www.eucalyptus.com/sites/all/files/ds-eucalyptus-iaas.en.pdf

[16] Feuerlicht, G.; Burkon, L.; Sebesta, M.: Cloud Computing Adoption: What are the Issues. *Systémová integrace*, ročník 18, č. 2, 2011: s. 187–192.

[17] Flexiant: Common considerations when selecting your hypervisor. 02 2014 [cit. 2014-03-01].
URL http://www.flexiant.com/2014/02/12/hypervisor-comparison-kvm-xen-vmware-hyper-v/

[18] FSN: The economy is flat so why are financials Cloud vendors growing at more than 90 percent per annum. 03 2013 [cit. 2014-02-25].
URL http://www.fsn.co.uk

[19] Furht, B.; Escalante, A. (editoři): *Handbook of Cloud Computing*. Springer New York Dordrecht Heidelberg London, 2010, ISBN 978-1-4419-6523-3.

[20] Garfinkel, S.: The Cloud Imperative. 10 2011 [cit. 2014-01-13].
URL http://www.technologyreview.com/news/425623/the-cloud-imperative/

[21] Grolmus, P.; Švamberg, M.: Single Sign-On řešení pro webové aplikace. *Vyšlo ve sborníku XXVII. konference EurOpen*, 2005: s. 87–100.

[22] Gupta, R.: Request Flow for Provisioning Instance in Openstack. 04 2013 [cit. 2014-04-12].
URL http://ilearnstack.com/2013/04/26/request-flow-for-provisioning-instance-in-openstack/comment-page-1/

[23] Haas, F.: OpenStack High Availability Guide. 04 2014 [cit. 2014-05-03].
URL http://docs.openstack.org/high-availability-guide/content/index.html

[24] Hefner, K.: Definition OpenNebula. 11 2013 [cit. 2014-01-25].
URL http://searchcloudstorage.techtarget.com/definition/OpenNebula

[25] Hof, R. D.: Jeff Bezos' Risky Bet. 11 2006 [cit. 2014-02-12].
URL http://www.businessweek.com/stories/2006-11-12/jeff-bezos-risky-bet

[26] Hufferd, J. L.: *ISCSI: The Universal Storage Connection*. Addison-Wesley Professional, 2003, ISBN 978-0201784190.

[27] IBM: Taking the Sting Out of Virtualization Licensing Costs with KVM. 09 2012 [cit. 2014-03-25].
URL https://www.ibm.com/developerworks/community/blogs/ibmvirtualization/entry/taking_the_sting_out_of_virtualization_licensing_costs_with_kvm?lang=en

[28] Jiang, Q.: CY13-Q4 Open Source IaaS Community Analysis. 01 2014 [cit. 2014-03-01].
URL http://www.qyjohn.net/?p=3432

[29] Klepáč, M.: *Private IaaS cloud comparison*. Diplomová práce, Czech Technical University in Prague.

[30] KVM: Main Page. [cit. 2014-02-25].
URL http://www.linux-kvm.org/page/Main_Page

[31] Marko, K.: iSCSI vs. Fibre Channel: A Cost Comparison. [cit. 2014-02-15].
URL http://www.processor.com/editorial/article.asp?article=articles/p3014/31p14/31p14.asp

[32] Mell, P.; Grance, T.: The NIST Definition of Cloud Computing. 09 2011 [cit. 2014-01-25].
URL http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[33] Meyer, D.: Here's why CERN ditched OpenNebula for OpenStack. 05 2013 [cit. 2014-03-13].
URL https://gigaom.com/2013/05/31/heres-why-cern-ditched-opennebula-for-openstack/

[34] Meyer, D.: OpenNebula 4.0 guns for the vCloud crowd. 05 2013 [cit. 2014-03-15].
URL https://gigaom.com/2013/05/08/opennebula-4-0-guns-for-the-vcloud-crowd/

[35] MySQL: Overview of MySQL with DRBD/Pacemaker/Corosync/Oracle Linux. [cit. 2014-03-12].
URL https://dev.mysql.com/doc/refman/5.0/en/ha-drbd.html

[36] OpenNebula: An Overview of OpenNebula. [cit. 2014-04-02].
URL http://docs.opennebula.org/4.6/design_and_installation/building_your_cloud/intro.html

[37] OpenStack: How to Perform an Upgrade from Grizzly to Havana. 12 2013 [cit. 2014-03-25].
URL http://docs.openstack.org/trunk/openstack-ops/content/ops_upgrades_grizzly_havana-ubuntu.html

[38] OpenStack: Hypervisor Support Matrix. [cit. 2014-01-25].
URL https://wiki.openstack.org/wiki/HypervisorSupportMatrix

[39] OpenStack: Documentation. [cit. 2014-03-20].
URL http://docs.openstack.org/

[40] Parkhill, D.: *The Challenge of the Computer Utility*. Addison-Wesley Educational Publishers Inc., 1966, ISBN 9782017700593.

[41] Patka, L.: *Virtualizační technologie*. Diplomová práce, Masarykova univerzita, 2009.

[42] Popek, G. J.; Goldberg, R. P.: Formal Requirements for Virtualizable Third Generation Architectures. *Communications of the ACM*, 1974.
URL http://www.logos.ic.i.u-tokyo.ac.jp/~tau/lecture/operating_systems/gen/papers/p412-popek.pdf

[43] QArea: Cloud Computing Outlook: IaaS, PaaS and SaaS. [cit. 2014-03-01].
URL http://www.qarea.com/articles/cloud-computing-outlook-iaas-paas-and-saas

[44] Reese, G.: *Cloud Application Architectures*. O'Reilly Media, Inc., 2009, ISBN 9780596555481.

[45] Renski, B.: VMware Vs. KVM: OpenStack Hypervisor Race Heats Up. 12 2013 [cit. 2014-02-27].
URL http://www.mirantis.com/blog/vmware-vs-kvm-openstack-hypervisor-race-heats-2/

[46] Rhoton, J.: *Cloud Computing Explained: Implementation Handbook for Enterprises*. 2009, ISBN 978-0956355607.

[47] Ryba, A.: ČR vede v zavádění ICT ve střední a východní Evropě. 03 2014 [cit. 2014-04-10].
URL http://www.ictmanazer.cz/2014/03/cr-vede-v-zavadeni-ict-ve-stredni-a-vychodni-evrope/

[48] Sedlák, J.: Radiokomunikace staví druhý cloud, poběží na otevřeném OpenStacku. 03 2014 [cit. 2014-04-30].
URL http://connect.zive.cz/bleskovky/radiokomunikace-stavi-druhy-cloud-pobezi-na-otevrenem-openstacku/sc-321-a-172912

[49] Severalnines: Clustering MySQL Backend in OpenStack. 08 2013 [cit. 2014-05-01].
URL http://www.severalnines.com/blog/clustering-mysql-backend-openstack

[50] Sim, P.: OpenStack Networking - with Open vSwitch VLAN, GRE. 12 2013 [cit. 2014-01-12].
URL http://www.slideshare.net/janghoonsim/open-stack-networking-vlan-gre

[51] Solinea: OpenStack Grizzly Architecture (revisited). 06 2013 [cit. 2014-03-29].
URL http://www.solinea.com/blog/openstack-grizzly-architecture-revisited

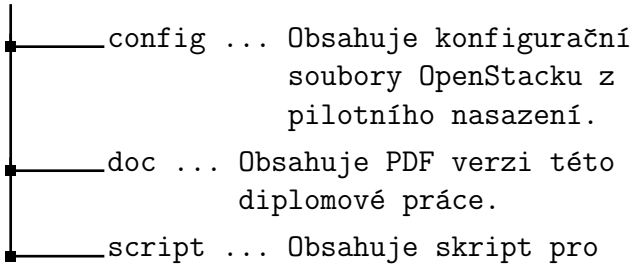
[52] Traeger, A.: OpenStack Cinder Deep Dive. 2013 [cit. 2014-02-20].
URL https://wiki.openstack.org/w/images/3/3b/Cinder-grizzly-deep-dive-pub.pdf

[53] Veber, J.: *Služby cloud computing v České republice [online]*. Disertační práce, Vysoká škola ekonomická v Praze, 2009 [cit. 2014-04-10].
URL http://theses.cz/id/58ov2m/

[54] VMware: Understanding Memory Resource Management in VMware® ESX™ Server. 2009 [cit. 2014-04-05].
URL http://www.vmware.com/files/pdf/perf-vsphere-memory_management.pdf

[55] William, S.; Stallings, W.: *Cryptography And Network Security, 4/E*. Pearson Education, 2006, ISBN 9788177587746.
URL http://books.google.cz/books?id=qKcrce0x_2YC

A Obsah CD


config ... Obsahuje konfigurační soubory OpenStacku z pilotního nasazení.
doc ... Obsahuje PDF verzi této diplomové práce.
script ... Obsahuje skript pro zakládání uživatelů z WebAuth/Orion.