VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

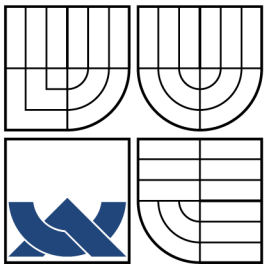
IMPLEMENTACE CDN A CLUSTERINGU V PROSTŘEDÍ GNU/LINUX S TESTY VÝKONNOSTI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

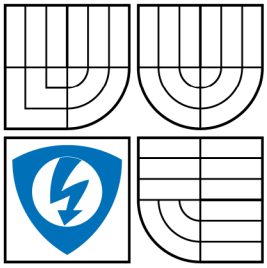
AUTOR PRÁCE
AUTHOR

BC. PAVEL MIKULKA

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE CDN A CLUSTERINGU V PROSTŘEDÍ GNU/LINUX S TESTY VÝKONNOSTI

CDN AND CLUSTERING IN GNU/LINUX

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. PAVEL MIKULKA
AUTHOR

VEDOUCÍ PRÁCE — ING. MILAN ŠIMEK
SUPERVISOR

BRNO 2008

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

ZDE VLOŽIT PRVNÍ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ZDE VLOŽIT DRUHÝ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ABSTRAKT

Odolnost proti chybám je v produkčním prostředí velmi důležitá. Jedna z možností řešení vysoké dostupnosti je vytvoření clusterového prostředí. Práce se zabývá možnostmi implementace serverového prostředí s důrazem na řešení vysoké dostupnosti poskytovaných služeb a rozložení záteže. Jsou zde popsány metody využití virtualizace, centrálního ukládáni dat, synchronizace a rozebrány použitelné technologie. Pro účely synchronizace dat je zde popsán nástroj DRDB, určený pro vytvoření synchronizovaných blokových zařízeni. Byly ověřeny praktické možnosti běhu těchto systémů za použití open-source nástrojů v prostředí GNU/Linux a otestovány možnosti programů Linux-HA, DRBD, Redhat Cluster Suite, LVS, Piranha. Sítě CDN replikují data na více geograficky rozložených serverů pro dosažení vyššího výkonu a schopnosti odolávat velkým zátěžovým špičkám. Práce se zabývá popisem základních mechanizmů replikace, přesměrování a implementací pomocí open-source nástrojů. Pro ověření funkce bylo použito sítě Planet Lab. U nástrojů Globule a CoralCDN byly provedeny testy na globálně rozmístěných serverech.

KLÍČOVÁ SLOVA

GNU/Linux, cluster, vysoká dostupnost, CDN, virtualizace, rozložení záttěže, drbd, Apache, Linux-HA

ABSTRACT

Fault tolerance is essential in a production-grade service delivery network. One of the solution is build a clustered environment to keep system failure to a minimum. This thesis examines the use of high availability and load balancing services using open source tools in GNU/Linux. The thesis discusses some general technologies of high availability computing as virtualization, synchronization and mirroring. To build relatively cheap high availability clusters is suitable DRDB tool. DRDB is tool for build synchronized Linux block devices. This paper also examines Linux-HA project, Redhat Cluster Suite, LVS, etc. Content Delivery Networks (CDN) replicate content over several mirrored web servers strategically placed at various locations in order to deal with the flash crowds. A CDN has some combination a request-routing and replication mechanism. Thus CDNs offer fast and reliable applications and services by distributing content to cache servers located close to end-users. This work examines open-source CDNs Globule and CoralCDN and test performance of this CDNs in global deployment.

KEYWORDS

GNU/Linux, cluster, High-Availability, CDN, Content delivery network, virtualization, loadbalancing, drbd, Apache, Linux-HA

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Implementace CDN a clusteringu v prostředí GNU/Linux s testy výkonnosti“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

OBSAH

Úvod **11**

1 Clustery **13**
1.1 Typy clusterů . . . 13
1.1.1 Vysoká dostupnost . . . 13
1.1.2 Výpočetní clustery . . . 17
1.1.3 Úložné clustery . . . 19
1.1.4 Rozložení zátěže . . . 20

2 Virtualizace **22**
2.1 Výhody a nevýhody . . . 22
2.2 Typy virtualizace . . . 22
2.3 Použití virtualizace v HA . . . 24

3 Datová uložiště **27**
3.1 DRBD . . . 27
3.2 Použití SAN . . . 28

4 CDN - Content delivery networks **31**
4.1 CDN - historie a souvislosti . . . 31
4.2 PlanetLab . . . 31
4.3 CDN - princip . . . 32
4.4 Komerční CDN . . . 33
4.5 Prvky CDN . . . 34
4.5.1 Přesměrování . . . 34
4.6 Open Source CDN . . . 36
4.6.1 CoralCDN . . . 36
4.6.2 Globule . . . 36

5 Praktické testy **38**
5.1 Testovací prostředí . . . 38
5.2 LinuxHA . . . 38
5.3 DRBD . . . 38
5.4 Linux-HA . . . 41
5.5 RedHat Cluster Suite . . . 43
5.5.1 LVS . . . 43
5.5.2 High Availabilty v RHCS . . . 49
5.5.3 GFS . . . 49

5.6 Apache - mod_proxy_balancer 49
5.6.1 Konfigurace LB 50
5.6.2 Zachování session 52
5.7 CoralCDN 52
5.7.1 Testování výkonu 52
5.8 Globule 53
5.8.1 Instalace 53
5.8.2 Konfigurace 55
5.8.3 Výsledky testování 58
5.8.4 Apache JMeter 59
5.8.5 Problémy s planetLabem 59

6 Závěr **61**

Literatura **63**

Seznam symbolů, veličin a zkratek **65**

SEZNAM OBRÁZKŮ

1.1 typické zapojení HA s částečnou redundancí a sdíleným diskem 15
1.2 zapojení HA se STONITH zařízením 16
1.3 typické zapojení s load balancerem 21
2.1 schema hardware virtualizace 23
2.2 schema virtualizace na úrovni OS 24
2.3 Active/active na jednom serveru 25
2.4 Active/active na více fyzických serverech 25
2.5 Kombinace fyzického a virtuálního serveru 26
2.6 Migrace virtuálních serverů 26
3.1 Pozice drbd v I/O vrstvě GNU/Linux [7] 28
3.2 schema clusteru s drbd replikací 29
4.1 PlanetLab - rozložení serverů 32
4.2 Způsob skládání URL v Akamai [?] 34
4.3 Typický průběh dotazu v CDN 35
5.1 Piranha - základní obrazovka 44
5.2 Piranha - globální nastavení 45
5.3 Piranha - nastavení redundance 45
5.4 Piranha - výpis balancovaných služeb 46
5.5 Piranha - nastavení balancované služby 46
5.6 Piranha - nastavení ip adres reálných back-end serverů 47
5.7 LVS - Graf rozložení doby odezvy serveru 48
5.8 Conga - architektura systému 49
5.9 Zobrazení stavu mod_proxy_balancer pomocí balancer-manager 51
5.10 Odezva na serveru v Brazilii 53
5.11 Odezva ze serveru v Tokiu 54
5.12 Odezva ze serveru v Austinu 54
5.13 Odezva ze serveru v Moskvě 55
5.14 Globule testovací topologie 56
5.15 Rozhraní programu jMeter 60

SEZNAM TABULEK

1.1 dostupnost ve ztahu k času výpadku 14
5.1 Význam zkratek ve výstupu `cat /proc/drbd` 42
5.2 Porovnání přenosových rychlostí 53
5.3 Doba odezvy serverů ve světě z master serveru(Praha) 55

ÚVOD

S rostoucím rozmachem Internetu do všech oblastí lidského života se vysoká dostupnost poskytovaných online služeb stává stále více důležitá. Nedostupnost serverů poskytujících důležité služby může totiž způsobit nejen velké finanční ztráty, ale může mít i sociální dopad. Proto je potřeba při návrhu nemyslet jen na pořizovací náklady, ale hlavně na dopady, které přinesou výpadky funkčnosti, které pak mohou několikanásobně přesahovat pořizovací částku.

Tato práce se zabývá možnostmi implementace serverového prostředí s důrazem na řešení vysoké dostupnosti poskytovaných služeb (high-avalability) a rozložení serverové zátěže (load-balancing) s využitím volně dostupných open-source nástrojů v prostředí operačního systému GNU/Linux. Rozebírá možnosti použití clusterových technologií k dosažení vyšší dostupnosti a vyššího výkonu. Právě pro dosažení vyššího výkonu je výhodné rozdělit zátěž mezi více uzlů sítě. Budou tu navrhnuty tedy i metody rozdělování požadavků mezi jednotlivé servery.

V případě použití řešení s více výpočetními uzly je jeden z klíčových bodů úspěšné implementace vyřešení správné replikace a sdílení dat mezi všechny prvky clusteru. Proto se tu nabídnou nástroje pro centrální ukládání a synchronizaci dat mezi uzly serverové farmy.

Protože v dnešní době mají velký rozmach virtualizační technologie, tak se tu popíší možnosti použití těchto moderních technologií k dosažení vyšší dostupnosti a výkonnosti s menšími celkovými náklady na implementaci a celkovou správu. Budou zde zmíněny modely zapojení tohoto progresivního řešení HA systému.

Překotný vývoj Internetu stále zvyšuje nároky na rychlosti přenášených dat a mění celkový pohled na způsob poskytování dat k cílovému klientovi, tak je potřeba přizpůsobit i původní model sítě. Zajímavou alternativu ke klasickému způsobu komunikace server-klient je metoda Peer-To-Peer. Tohoto se dá s výhodou použít právě pro rozložení síťové zátěže mezi více fyzických uzlů sítě. Budou zde tedy nastíněny možnosti použití CDN (Content delivery network) které využívají vlastnosti modelu P2P.

CDN jsou převážně doménou komerčních institucí. Tato práce má za úkol ověřit možnosti implementace CND na platformě GNU/Linux s použitím pouze svobodného software a ověřit konkurenceschopnost ke komerčním řešením. Předmětem těchto nástrojů se stane i jejich výkon a porovnání s běžnými metodami rozložení zátěže. Tyto testy budou zaměřeny především na nasazení CDN v globálním měřítku. Tím se rozumí pokrytí a optimalizaci pro poskytování služeb na celém světě.

Protože má Vysoké učení technické v Brně přístup do vědecké sítě PlanetLab, která je vybudována právě pro vývoj a testování nových aplikací a protokolů budoucího Internetu bude zde popsán její princip a cíle. Protože nám tato síť poskytuje

přístup k serverům na mnoha světových lokalitách, tak této sítě využijeme pro testování CDN.

Co je GNU/Linux?

Jméno Linux značí jádro operačního systému původně vyvinutý Linusem Torvaldsem v roce 1991. Torwalds vycházel z klonu Unixu nazvaného Minix od Andrewa Tanenbauma. Toto jádro spolu s nástroji z GNU projektu založeného Richardem Stalmanem v roce 1983 tvoří operační systém. Správně by se tedy mělo hovořit o GNU/Linux, což značí jádro plus další nástroje, které dohromady tvoří operační systém. Pokud dále budeme mluvit o Linuxu, myslí se GNU/Linux.

Proč Linux?

V prvé řadě, protože má otevřený zdojový kód (Open Source). Tato vlastnost je důležitá v případě implementace složitějších systémů, kdy je potřeba konfiguraci upravit na míru a v případě komerčního uzavřeného systému to není většinou možné. Další důvod je, že většina clusterových systémů vzniká na akademické půdě, kde je právě Open Source oblíben z důvodu sdílení zdrojových kódů a možnosti jeho úpravy. Jasná převaha použití Linuxu v clusterech je vidět na stránkách projektu Top500 (http://www.top500.org), který udržuje statistiky nejvýkonnějších počítačů světa. OS Linux je použit na 76,8% [1] z 500 nejvýkonnějších počítačů světa.

[1] statistika z listopadu 2007

1 CLUSTERY

Cluster je skupina počítačů, které spolu úzce spolupracují tak, že navenek se jeví jako jeden počítač. Výměna dat mezi clustery je obvykle řešena přes síť.

Clustery se používají k zvýšení výpočetní rychlosti, nebo spolehlivosti v případech, kdy by tyto vlastnosti jeden počítač nezvládl, nebo by byl neúměrně nákladné jeho pořízení nebo provoz. V minulosti byly clustery doménou špičkových vědeckých a vládních pracovišť, ale dnes se běžně uplatňují v komerční sféře nebo školství.

Možné funkce clusteru jsou tyto:

- Škálovatelnost - cluster lze rozšiřovat přidáváním dalších uzlů, možnosti růstu obecného clusteru jsou omezeny pouze propustností sdílených komponent (hlavní uzel, propojovací síť)
- Odolnost vůči výpadku (fault tolerance) - při výpadku jednoho uzlu přestane cluster tento uzel používat. Sníží se výpočetní výkon, ale cluster funguje dál (do výpadku sdílené části nebo všech serverů)
- Vyšší výkon - paralelním zpracováním úkolu na více uzlech dosáhneme vyššího výkonu případně zlepšíme poměr cena/výkon

1.1 Typy clusterů

1.1.1 Vysoká dostupnost

Tyto clustery se budují s cílem zajistit pomocí několika počítačů nepřetržité poskytování služby i v případě výpadku některého prvku v clusteru. Známější výraz pro tento typ clusteru je HA-cluster[1].

High Avalability řešení představuje kompletní infrastrukturu složenou z hardware + software + konfigurace, která by měla být odolná proti výpadkům a to zejména proti nepředvídaným poruchám hardware. Tyto systémy mohou běžet v režimu Active-Passive, což znamená že na jednom serveru běží všechny služby a v případě výpadku se migrují na druhý záložní server. Druhá možnost je režim Active-Active, kde na jednom serveru běží například HTTP server a na druhém DB server a v případě výpadku jednoho servery se služba přesune na funkční uzel. Typické zapojení je na obr. 1.1.

Procentuální míra dostupnosti je důležitým měřítkem HA systémů. Abychom ji byly schopni ji určit musíme mít k dispozici statistiky z monitoringu dostupnosti.

[1] High Avalability

Dostupnost se tak nejčastěji udává v poměru počtu minut neplánovaného výpadku k počtu minut v roce.

$$Dostupnost = \frac{MTBF}{MTBF + MTTR} \tag{1.1}$$

kde MTBF[2] znamená střední dobu mezi poruchou a MTTR[3] střední dobu opravy. Dostupnost se pak typicky uvádí procentech s číslicí devět na konci. Počet devítek pak určí stupeň dostupnosti. V tabulce 5.1 uvádím maximální časy výpadků za rok pro dosažení určité úrovně dostupnosti.

Dostupnost	**výpadky za rok**	**výpadky za měsíc**	**výpadky za týden**
90%	36,5 dnů	72 hodin	16,8 hodin
95%	18,25 dnů	36 hodin	8,4 hodin
98%	7,30 dnů	14,4 hodin	3,36 hodin
99%	3,65 dnů	7,20 hodin	1,68 hodin
99,5%	1,83 dnů	3,60 hodin	50,4 min
99,8%	17,52 hodin	86,23 min	20,16 min
99,9%	8,76 hodin	43,2 min	10,1 min
99,95%	4,38 hodin	21,56 min	5,04 min
99,99%	52,6 min	4,32 min	1,01 min
99,999%	5,26 min	25,9 s	6,05 s
99,9999%	31,5 s	2,59 s	0,605 s

Tab. 1.1: dostupnost ve ztahu k času výpadku

SPOF

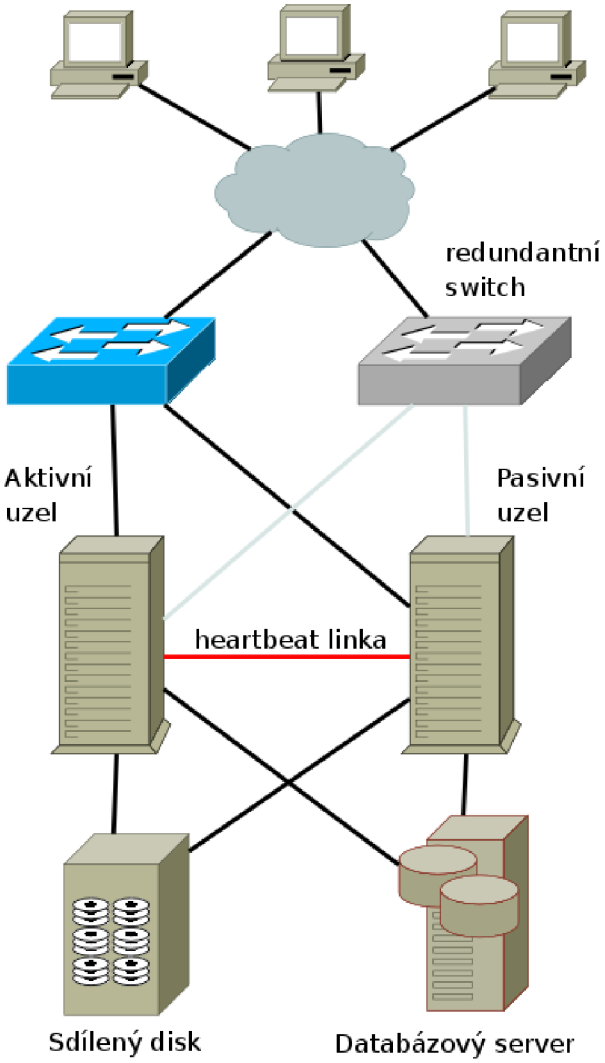
SPOF (A Single Point of Failure) je nejslabší článek systému, který když přestane fungovat, ohrozí běh celého systému. Důležitým předpokladem úspěšného návrhu systému s vysokou dostupností je použití kvalitních komponent systému, ale hlavně odhalení a vyloučení co možná nejvíce SPOF.

Správný HA systém eliminuje všechny možné SPOF. SPOF může být například sdílený switch, sdílený síťový disk, síťová linka vedená stejnou cestou/kabelem. Základní metoda odstranění SPOF je vytvoření redundance. Vždy by měla být snaha o vytvoření plné redundance, ale většinou se tak neděje z důvodu vysokých finančních nákladů. Na obrázku 1.1 tak představuje SPOF sdílený disk, proto se snažíme sdílený disk zabezpečit alespoň pomocí jiných technologií jako třeba použitím RAID polí.

[2]mean time before failure

[3]mean time to repair

K problému řešení vysoké dostupnosti je potřeba přistupovat komplexně, je zbytečné mít drahé hardware zařízení zapojené do HA clusteru, když se pak zapomene například na zálohované napájení, nebo použijeme diskové pole bez RAID.



Obr. 1.1: typické zapojení HA s částečnou redundancí a sdíleným diskem

Fencing

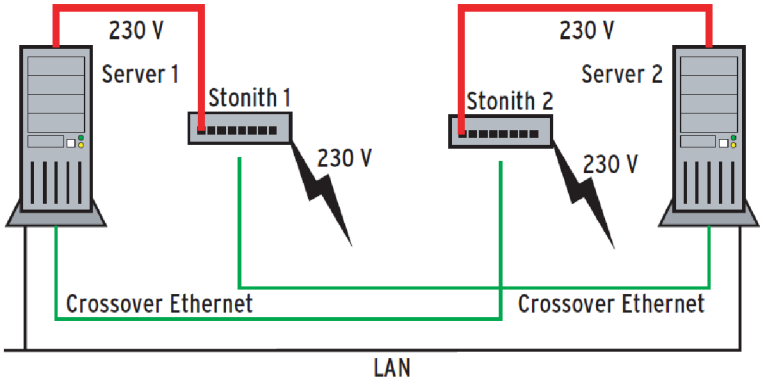
Je procedura k zajištění aktuálního stavu v nejednoznačných situacích. Všechny uzly sítě si musí mezi sebou vyměňovat informace který uzel je živý a který není, ale například při výpadku konektivity k serveru muže nastat problém. Nastane stav, kdy se situace vyhodnotí tak, že se začnou služby startovat na jiném serveru přestože ještě běží na původním. V případě, že více systémů sdílí jeden disk bez sdíleného filesystému, je potřeba zajistit, aby se nestalo, že se bude zapisovat z více míst současně. Toto může nastat pokud jeden server vyhodnotí že druhý server spadl

a začne zapisovat přičemž druhý server jede dál a zapisuje také. Tohoto se musí vyvarovat jinak dojde k destrukci konzistence dat. V případě, že jeden uzel neví o tom, že druhý je aktivní a vyhodnotí situaci že se sám stane aktivním, nastane tzv. stav **Split Brain**. Fencig lze zabezpečit více způsoby:

- FiberChannel Switch lockouts
- SCSI - funkce Reserve/Release
- Self-Fencing (např. IBM ServeRAID)
- STONITH – Shoot The Other Node In The Head (viz. dále)

STONITH (Shoot The Other Node In The Head)

STONITH je většinou HW zařízení které provede vypnutí serveru v případě že byl druhým (master) vyhodnocen jako nedostupný. STONITH je účinná metoda předcházení problému s tzv. SplitBrain. Pro tento účel lze použít například vzdáleně ovladatelné PDU[4] které zprostředkuje fyzické odpojení uzlu od napětí a tím ho vyřadí z provozu aby se nesnažil přistupovat ke sdíleným datům. Pro tuto činnost lze použít i například nastavitelný switch (managed). V tomto případě se vzdáleně provede odpojení od sítě a sdílených disků. Po odpojení administrátor dá věci do pořádku a vrátí systém do původního módu.



Obr. 1.2: zapojení HA se STONITH zařízením

Aplikace pro HA

Asi nepoužívanější balík programů pro řešení HA na GNU/Linux je dostupný na adrese http://www.linux-ha.org/. Je často nazývaný jako Heartbeat podle jména

[4]Power Distribution Unit

jeho hlavního modulu. Je vyvíjený od roku 1999 a dnes je součástí balíčků všech větších distribucí. Je vysoce přizpůsobitelný a je možné jej použít na téměř libovolné nasazení. Pro dobrou funkci ale vyžaduje velmi dobrou znalost celého systému.

Jeho konfigurace je možná dvěma způsoby. Starší systém konfigurace používaný ve verzích 1.x byl omezen na maximálně dva uzly, které si mezi sebou předávali služby. Ve verzi 2.x se přišlo s přepracovaným modelem kdy je možné použít neomezený počet uzlů, ale s tím souvisí i složitější celkové nastavení a vyladění migrace služeb. V nových verzích 2.x je možné podle potřeby použít jednoduchý model konfigurace pokud si vystačíme se dvěma uzly.

Heartbeat funguje tak, že do serverů umístíme navíc NIC[5] a propojíme servery do monitorovací sítě vyhrazenou pro heartbeat. Pokud máme jen dva uzly tak stačí propojit kříženým kabelem. Přes tuto linku si uzly vyměňují informace a zjišťují vlastní stav. Pro větší redundanci je možné tyto informace přenášet navíc po sériové lince.

Nástroje pro vytvoření HA clusteru nabízí i RedHat. Jedná se o celý balík programů který mimo jiné obsahuje řešení pro HA, rozložení zátěže, sdílený souborový systém (GFS). Toto řešení je nabízeno pod označením RedHat Cluster Suite (RHCS). Přestože jej RedHat nabízí spolu s podporou za nemalý peníz jedná se o OpenSouce. RHCS je dokonce součástí několika distribucí (CentOS, Debian, Ubuntu,...)

1.1.2 Výpočetní clustery

Výpočetní cluster [6] slouží k zvýšení výpočetního výkonu pomocí více počítačů, které na výpočtu spolupracují. Tímto způsobem vznikne vysoce výkonný celek, který je mnohonásobně levnější, než jeden vysoce výkonný počítač.

Nejpoužívanější Open Source řešení

OpenMosix Cluster je modifikace Linuxového kernelu, která se snaží clustering co nejvíce zjednodušit. OpenMosix přebírá kontrolu nad procesy ve chvíli volání například fork(), což umožňuje využití této technologie bez nutnosti větší úpravy zdrojového kódu. Pouhým štěpením na procesy lze dosáhnou vyššího výkonu migrováním těchto vláken na okolní body. OpenMosix se skládá z patche na kernel, obslužných programů, OpenMosix démona, OpenMosix kolektoru, který sbírá z pracujících nodů data. Nejčastěji se používal v podobě live distribuce bežící na jednotlivých uzlech. Stabilní verze podporuje pouze starou řadu Linuxového jádra 2.4.x.

[5] NIC - Network Interface Card

[6] High-performance computing (HPC) clusters

Přestože byla tato platforma značně oblíbená, vývojáři OpenMosix oznámily konec vývoje k březnu 2008. Dá se ale očekávat že vznikne nějaký fork[7] tohoto projektu.

Projekt Beowulf je projekt, který má umožnit složení výkonného výpočetního clusteru z běžně dostupného levného hardware. Vývoj začal v roce 1994 na pracovišti CESDIS (Center of Excellence in Space Data and Information Sciences), které je součástí centra vesmírných letů NASA. CESDIS bylo sponzorováno z projektu ESS (Earth and Space Sciences) k jehož úkolům patří i výzkum paralelních počítačů. Thomas Sterling a Donald Becker, dva zaměstnanci CESDIS, od konce roku 1993 pracovali na návrhu paralelního počítače postaveného pouze z běžně dostupných (levných) komponent a volně šiřitelného softwaru. Výsledkem jejich snahy byl cluster složený ze šestnácti PC s procesory 486DX4 propojených 10Mb vícekanálovým Ethernetem. FastEthernet a ethernetové přepínače byly tehdy poměrně drahé, proto Donald Becker přepsal ovladače Linuxového jádra tak, aby komunikace mezi jednotlivými uzly mohla být rozložena do dvou a více ethernetových sítí. Stroj i celý projekt byli pojmenováni podle hrdiny anglosaského eposu Beowulf.

Typický Beowulf cluster je složen z několika identických PC na kterých běží Open Source operační systém (Linux, BSD) a komunikace probíhá pomocí LAN. Pro počítače tohoto druhu se vžilo označení Beowulf Class Cluster Computer. S nárůstem výkonu a schopností počítačů PC a operačního systému Linux, který na Beowulf clusterech dominuje se odpovídajícím způsobem posunuly i možnosti počítačů typu Beowulf a staly se konkurencí pro malosériové superpočítače.

Komunita kolem projektu Beowulf se neustále vyvíjí a vývojáři i celá komunita se společně podílejí na dalším zlepšování a rozšiřování projektu. Důležitým faktorem pro současný rychlý nárůst výkonu počítačů typu Beowulf je především snižováním ceny a zvyšování dostupnosti vysokorychlostních síťových prvků, ale i zkvalitňování GNU softwaru, na kterém Beowulf stojí, a to nejen operačních systémů, ale i například překladačů C, C++ a Fortran.

Parallel Virtual Machine je software pro paralelní výpočty založený na myšlence použití sítě heterogenních počítačů jako jeden distribuovaný paralelní procesor. Vývoj PVM začal na univerzitě v Tenessee (Oak Ridge National Laboratory and Emory University) v roce 1989 a je vyvíjen dodnes. PVM je používán ve stovkách center pro vědecké, medicínské a průmyslové výpočty. Je to sada softwarových nástrojů a programátorských knihoven, která umožňuje a podporuje provoz paralelní aplikace na obecně nestejných počítačích propojených do sítě.

Základem PVM je démon pvmd3, který běží na každém počítači. Všechny uzly se spuštěným démonem pvmd3 mohou dohromady vytvářet virtuální stroj. Druhou

[7] pokračování vývoje v jiné větvi

částí PVM je knihovna, poskytující rozhraní pro paralelní operace. V současné době jsou podporovány jazyky C, C++ a Fortran.

OSCAR je v současnosti jeden z nejpoužívanějších HPC systémů na světě. Oscar je primární projekt skupiny Open Cluster Group[8]. Jedná se o balík nástrojů který zahrnuje vše pro provozování výpočetního clusteru. Vyznačuje se relativně jednoduchou instalací s dobrou dokumentací. Standartně podporuje tyto distribuce:

- Red Hat Enterprise Linux 4
- Fedora Core 4
- Fedora Core 5
- Mandriva Linux 2006
- SUSE Linux 10.0

Se zvyšujícím se počtem uzlů v clusteru se stává konfigurace a správa clusteru náročnější. Projekt Oscar byl založen za účelem zvládnout tento nedostatek. Balík programů obsažený v Oscar obsahuje vše potřebné pro běh a správu výpočetního clusteru. K základní instalaci používá nástroj SIS (System Instalation Suite).

System installer vytvoří nový diskový obraz který se přehraje na nový uzel pomocí SystemImager a pomocí System Configurator se nový uzel nastaví. Po rebootu tak máme funkční nový uzel. C3 je další obsažený nástroj používaný k spouštení programů na celém clusteru a distribuci souborů po clusteru. Ke správě uživatelů se používá nástroj Opium.

1.1.3 Úložné clustery

Úložný cluster [9] zprostředkovává přístup k diskové kapacitě, která je rozložena mezi více počítačů z důvodu dosažení vyššího výkonu nebo pro zajištění vyšší spolehlivosti. Toho je dosahováno speciálními souborovými systémy, které jsou schopny zajistit rozložení zátěže, redundanci dat, pokrytí výpadků jednotlivých uzlů, distribuovaný mechanismus zamykání souborů a další doprovodné služby. Mezi tyto souborové systémy se řadí například Lustre [10] a PVFS[11]

[8] http://www.openclustergroup.org/
[9] anglicky Storage cluster
[10] http://www.lustre.org/
[11] http://www.pvfs.org/

1.1.4 Rozložení zátěže

Jsou to systémy používané na serverech s vysokými datovými přenosy nebo na systémech kde potřebujeme spotřebovávaný výkon serveru rozložit mezi více uzlů ale není vhodné nebo nelze použít tzv. výpočetní cluster. Nejčastěji se používá na webových serverech, kde se požadavky klientů přesměrovávají mezi více uzlů. V tomto případě se navenek cluster tváří jako jeden síťový server a na jednotlivé cílové (backend) uzly clusteru jsou přesměrovány požadavky od různých klientů. Správný výraz pro tento systém je Load Balancing (dále LB).

Typy LB

Round robin DNS je nejjednodušší metoda jak rozdělit požadavky mezi více serverů. Při dotazu klienta na doménu DNS server vrátí pokaždé jinou IP adresu. Tato jednoduchá metoda se ale nehodí na služby kde je nutné zajistit stálost sezení (tzv. session). Například pokud je klient připojen na server s internetovým obchodem je potřeba aby byl klient posílán stále na jeden server aby se mu zachovaly položky v košíku.

Jednoduše vytvoříme A záznamy s více IP adresami. Nastavení zóny pro server Bind pak vypadá takto:

```
; zone file fragment
ftp IN A 192.168.0.4
ftp IN A 192.168.0.5
ftp IN A 192.168.0.6
www IN A 192.168.0.7
www IN A 192.168.0.8
```

Přepínání na transportní vrstvě je metoda kde se použije router pracující na 4. vrstvě a ten rozhoduje, na který backend[12] server požadavek pošle. Jeden hlavní uzel tak přerozděluje požadavky dalším uzlům. LB pak poskytuje vysokou škálovatelnost v podobě jednoduchého přidávání dalších uzlů. Moderní load balancery rozhodují o cíli na základě několika parametrů (podle počtu dotazů, polde množství dat,...). Většinou se jedná o hardwarové zařízení.

Pro tento druh LB je možné použít i Linux Virtual Server (LVS) [13] open-source projekt který implementuje software LB na transportní vrstvě do GNU/Linux.

Přepínání na aplikační vrstvě se používá v případě že je potřeba se o cílovém cestě rozhodovat na základě složitějších mechanizmů použité služby. U http se tak

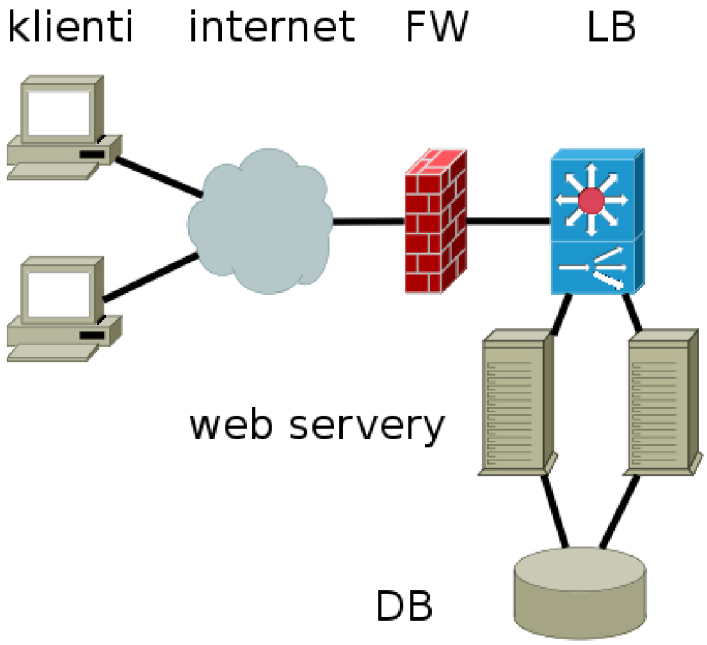
[12] vnitřní server

[13] http://www.linuxvirtualserver.org/

například může rozhodovat na základě URI[14], hostname, cookie[15]. Pro tento typ balancování se většinou používají software load balancery. V oblasti webových serverů jsou nejpoužívanější tyto nástroje:

- Pen - http://siag.nu/pen/
- Pound - http://www.apsis.ch/pound/
- HaProxy - http://haproxy.1wt.eu/
- mod_proxy - modul do Apache 2.2, buď jako reverzní proxy nebo balancer

Některé sofistikovanější HW balancery dokáží provádět i inspekci provozu na aplikační vrstvě a rozšiřují tak možnosti směrování na základě parametrů aplikačního protokolu oproti běžným hardwarově řešeným load balancerům.



Obr. 1.3: typické zapojení s load balancerem

[14]Uniform Resource Identifier
[15]řetězec uložený na klientově prohlížeči použitý např. pro jeho identifikaci

2 VIRTUALIZACE

Za pojmem virtualizace se obecně skrývají techniky uspořádání, ve kterých je možné k systémovým zdrojům přistupovat jako k množině výkonu bez ohledu na jejich fyzické charakteristiky. V kontextu serverů se myslí možnost provozu více logických instancí operačního systému nebo logických serverů v rámci jednoho fyzického serveru. Výhodné je to mimo jiné protože výkon dnešních serverů je většinou předimenzovaný.

Pojmem server se tak nemusí nutně rozumět fyzický stroj, ale může se jednat o server běžící na virtuálním prostředí. Cílem virtualizace je schovat hardware vrstvu systému pod virtualizační vrstvu.

2.1 Výhody a nevýhody

- Úspora nákladů - umožnuje ušetření hw prostředků a plné využití hardwarového výkonu
- Konsolidace - sloučení více poskytovaných služeb pod jeden fyzický hardware
- Použití více operačních systémů na jednom fyzickém stroji
- Jednodušší nakládání se zálohami
- Snadná obnova po pádu (Disaster Recovery), migrace
- Rozdělování výkonu

2.2 Typy virtualizace

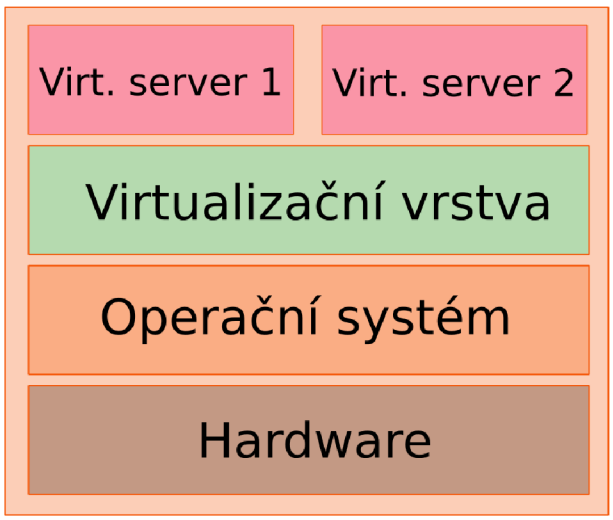
Full - plná virtualizace

Vytvoří se prostředí v němž operační systém nepozná, že nemá přístup k hardware. Operační systém ani aplikační programy nepotřebují žádné modifikace. Dochází tak k plnému oddělení fyzické vrstvy, veškeré programy běží pouze na virtuálním hardware a přístup k fyzickému vybavení je vždy zprostředkován. U plné virtualizace nemusí existovat žádná vazba mezi virtuálním prostředím a hardware, na němž je virtuální počítač provozován. To umožňuje plnou přenositelnost virtuálního stroje.

Tuto metodu používají tyto nástroje:

- VmWare (Server, ESX,...)
- VirtualBox

- QEMU



Obr. 2.1: schema hardware virtualizace

Virtualizační vrstva je umístěna mezi hardwarem a virtuálními servery. Tento typ virtualizace podporuje více operačních systémů na jednom serveru. Operační systém sám o sobě nepozná že běží jako virtualizovaný.

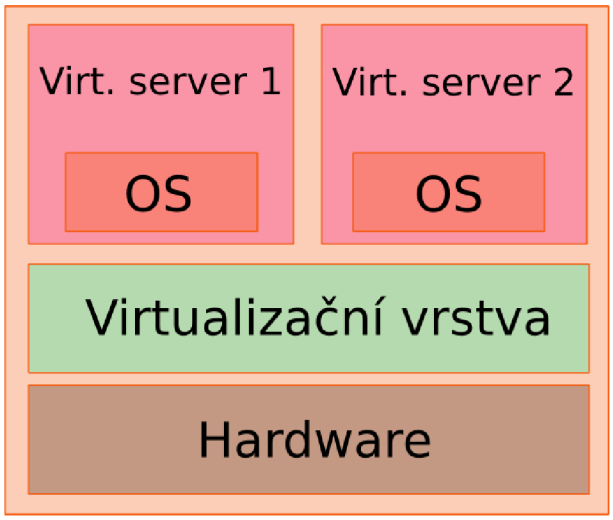
Paravirtualizace

Vychází z konceptu hardwarové virtualizace, ale virtuální stroj nemusí nezbytně simulovat hardware. Navíc totiž nabízí API vrstvu, která může být použita jen z upraveného OS hosta. Tudo metodu používá např. Xen. Nevýhoda je nutnosti použití upraveného OS na Guest[1] systému. Naopak ale může poskytnou větší výkon než plná virtualizace.

OS-Level

Virtualizuje se fyzický server na úrovni OS. Virtualizační vrstva je umístěna mezi operačním systémem serveru a virtuálními servery. Prostředí OS hosta sdílejí jeden OS s hostitelským systémem. Nevýhoda tohoto řešení je, že se podporuje jen jeden operační systém na jednom fyzyckém stroji. Používá např. OpenVZ, VServer.

[1] systém běžící pod virtualizací



Obr. 2.2: schema virtualizace na úrovni OS

2.3 Použití virtualizace v HA

Valná většina dnešních HA systémů je založena na fyzickém hardwaru. Spolu s tím jak se virtualizace stává více a více populární tak se ukazuje, že je výhodné spojovat ji do kombinace s HA systémy. Pomocí virtualizace se pak dá vytvořit několik topologií HA systémů a spojit tak výhody virtualizace s HA systémy.

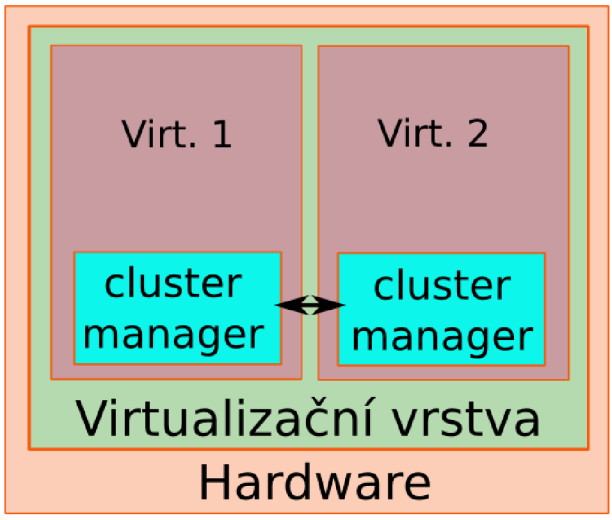
Active-Active na jednom HW

Toto nastavení je sestaveno ze dvou virtuálních strojů na jednom fyzické přičemž správce clusteru(heartbeat) běží na každém virtuální stroji zvlášť. Fyzický server tak jen poskytuje prostředí pro virtuální servery. Tato topologie není chráněná proti HW chybám, ale jen chybám software uvnitř virtuálního serveru.

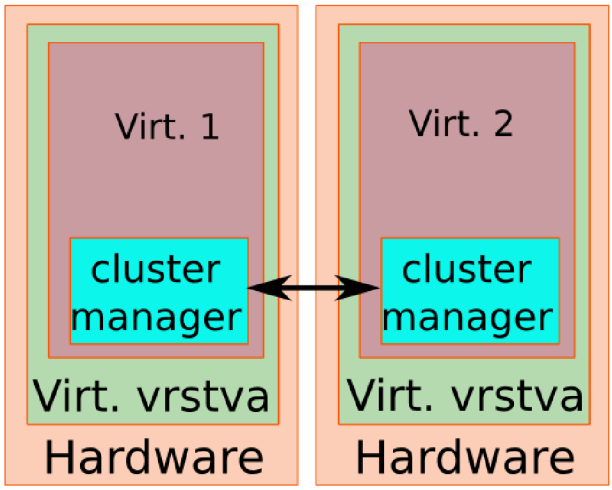
Výhoda tohoto řešení oproti klasickému spočívá pouze v ušetření HW prostředků. Je vhodná k otestování konfigurace HA ale do ostrého nasazení se příliš nehodí, protože obsahuje SPOF v podobě sdíleného HW.

Active-Active na více HW

V této konfiguraci běží HA cluster na dvou virtuálních serverech, které běží na dvou nezávislých fyzických strojích. Takto jsme se zbavily zmiňovaného SPOF. Základní výhoda spočívá v jednodušší správě serverů z důvodu použití virtualizace. Zálohy a obnova je jednodušší a rychlejší. V tomto případě ale nesmíme zapomínat i na zálohy hostitelského OS. Další výhoda je, že můžeme kromě HA clusteru provozovat ještě další nezávislé služby přívo na hostitelském OS bez virtualizace.



Obr. 2.3: Active/active na jednom serveru

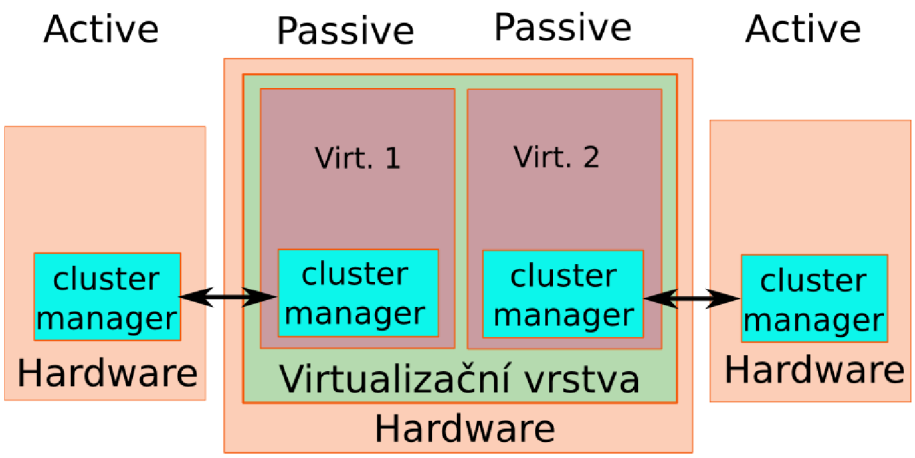


Obr. 2.4: Active/active na více fyzických serverech

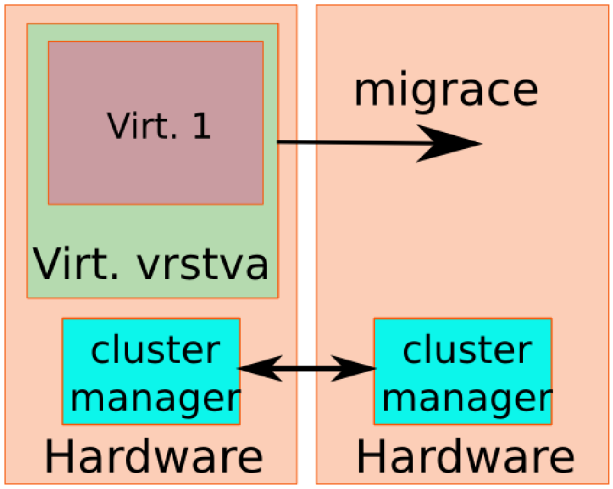
Kombinace fyzického a virtuálního serveru

Tady je HA cluster vytvořen pomocí aktivních fyzických serverů a záložních virtuálních. Touto konfigurací ušetříme HW prostředky oproti jiným active/passive řešením při zachování plné redundance.

Tento scénář je možné také obrátit, a mít jeden vysoce výkonný HW na kterém poběží dva active nody a passive nody poběží na záložních low-endových fyzických serverech. Toto řešení může ve výsledku vyjít levněji.



Obr. 2.5: Kombinace fyzického a virtuálního serveru



Obr. 2.6: Migrace virtuálních serverů

Migrace celých virtuálních serverů

Na rozdíl od předchozích kde běžel cluster manager uvnitř VM, v tomto případě umístíme cluster manager na OS hostitelského systému. V případě výpadku migrujeme celý image virtuálního serveru.

Toto nastavení nám poskytuje výhody v podobě zjednodušení nastavení HA, protože jedinou službu o kterou se cluster manager stará je zapínání/vypínání VM. Nemusíme se tak starat o celou hierarchii spouštených aplikací. Můžeme dokonce provozovat několik virtuálních HA systémů za pomoci jednoho cluster manageru a konzolidovat tak větší počet clusterů. Nevýhoda může být v delším čase migrace protože se musí spustit celý VM.

3 DATOVÁ ULOŽIŠTĚ

Ať už vytváříme LB, HA systém, nebo výpočetní cluster, vždy se musíme nějakým způsobem postarat o synchronizaci dat mezi uzly. Jiný přístup k synchronizaci vyžadují statická data které se mění jen zřídka na rozdíl od dat se kterými se nepřetržitě pracuje.

Pro synchronizaci pasivních dat jako jsou konfigurace programů se dá vystačit se standardními nástroji typu scp, rsync. Pro synchronizaci na více uzlů zároveň se dá s úspěchem použít csync2. Csync2 je nástroj vyvinutý pro pro synchronizaci více uzlů v clusterů který zvládne neomezený počet uzlů. Synchronizace může být šifrovaná pomocí SSL a podporuje různé možnosti řešení konfliktů mezi duplicitními soubory.

Některé aplikace poskytují vlastní model replikace jejich dat. Mezi tyto služby patří DNS, LDAP, DB2, Mysql, PostgreSQL(Slony).

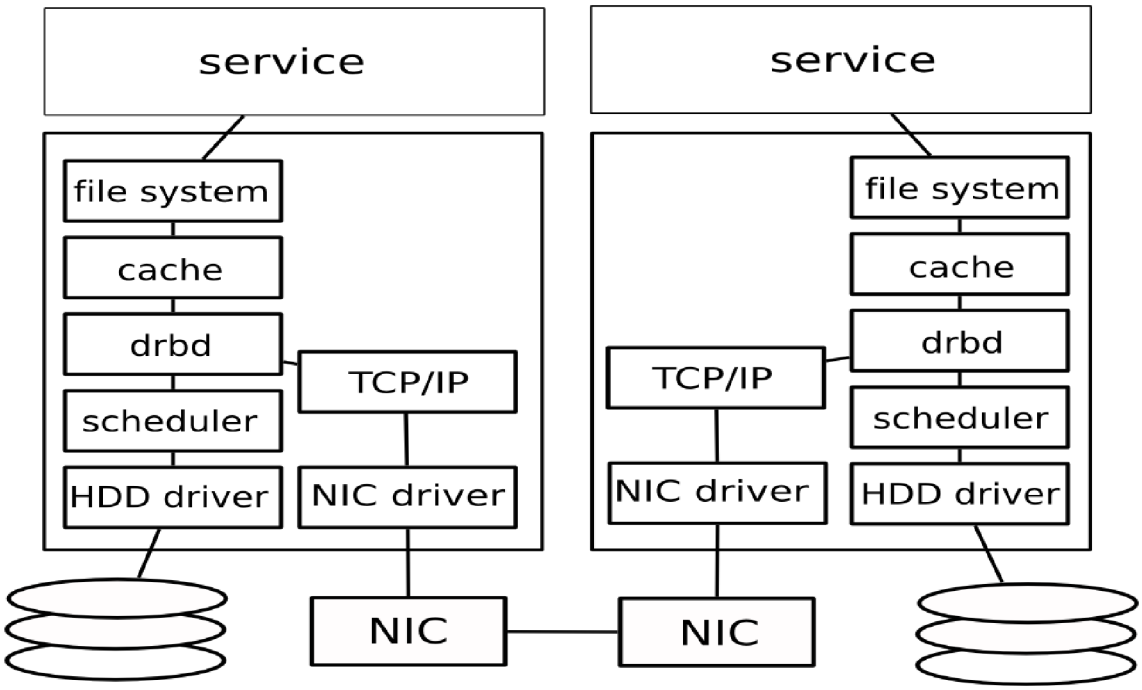
3.1 DRBD

Nevýhoda předešlé metody spočívá vtom, že synchronizace probíhá v předem stanovených intervalech a není tedy spojitá. V jednom okamžiku tak můžeme mít jinou verzi souboru na různých serverech. Pokud nám tedy v HA systému vypadne jeden uzel nemáme zaručenu konzistenci dat na obou serverech. Toto se dá řešit hardwarovými HA diskovými poli.

Tento problém lze v Linuxu efektivně vyřešit pomocí DRBD. Řešení s DRBD jsou pak překvapivě levná. Jedná se o nástroj pro online replikaci dat mezi dvěma servery zapojenými do HA clusteru. Skládá se z jaderného modulu, který vytváří virtuální blokové zařízení nad diskovým zařízením a uživatelskými (userspace) utilitami pro nastavování a administraci. DRBD tak vytvoří další vrstvu mezi hardware disku a souborovým systémem která se stará o replikaci dat na druhý uzel. Vytváří tak vlastně síťovou obdobu RAID1 pole rozloženého mezi dva servery.

DRBD velice dobře spolupracuje s projektem Linux-HA. Na obrázku 4.1 je tak znázorněna funkce a zapojení DRBD.

Nad blokovým zařízením DRBD se pak používají standardní souborové systémy EXT2, EXT3, XFS, JFS, ... Z tohoto důvodu může být pouze jeden uzel ve stavu primary na který se zapisuje a data se propagují na druhý uzel ve stavu secondary. Od verze 0.8 je možné povolit režim Active-Active kdy jsou oba servery v aktivním stavu ale je potřeba použít některý sdílený filesystém(GFS, OCSF2). V době uvedení verze 0.8 jsem tuto možnost zkoušel spolu s souborovým systémem GFS ale toto řešení nebylo příliš stabilní a docházelo k rozpadu filesystému



Obr. 3.1: Pozice drbd v I/O vrstvě GNU/Linux [7]

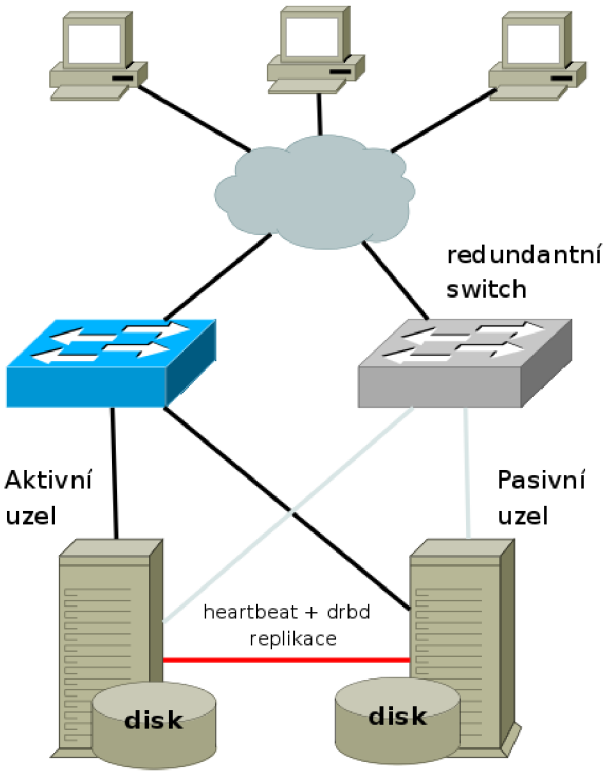
3.2 Použití SAN

Storage area network je síť sloužící k připojení vzdálených externích diskových polí. K tomuto účelu se používají tyto technologie:

- Fibre Channel - drahá technologie, robustní řešení
- SCSI - klasické řešení, omezená délka kabelů
- iSCSI - SCSI zapouzdřené do TCP paketů a přenášené gigabitovým ethernetem, levné a flexibilní řešení
- eSATA - externí verze SATA pripojení disku
- ATA over Ethernet - protokol pro přístup k blokovému ATA zařízení přes ethernet, daleko jednodušší standart jak iSCSI, nepodporuje routování

Pokud má k jednomu diskovému oddílu přistupovat více uzlů zároveň musíme zajistit použití některého sdíleného souborového systému. Standardní souborové systémy totiž vycházejí z předpokladu, že i při souběžné aktivitě několika procesů existuje jeden centrální bod, který jako jediný pracuje s úložným diskovým polem. Tímto je zajištěna konzistence dat i jeho metadata (vlastníci, přístupová práva, časy poslední změny apod.).

Tyto clusterové systémy dělíme na symetrické a asymetrické.



Obr. 3.2: schema clusteru s drbd replikací

Symetrické souborové systémy jsou takové, na jejichž klientských uzlech běží také správce metadat jako nedělitelná součást klientské části souborového systému.

Asymetrické souborové systémy mají na rozdíl od symetrických v clusteru vyhrazen jeden nebo více správců metadat, které spravují strukturu prvků souborového systému a následnou strukturu na sdíleném úložišti. Typicky tito správci běží na stejném stroji jako datové úložiště nebo klient.

GFS - Global File System

GFS je souborový systém navržený přímo pro použití v clusterovém prostředí vyvíjený firmou Redhathttp://www.redhat.com/software/rha/gfs/. Je standardní součástí Linuxového vanilla jádra od verze 2.6.19. V tomto souborovém systému se počítá s tím, že k jednomu úložnému zařízení (disk, diskové pole, atd.) je přímo (přes sdílené SCSI, Fibre-channel nebo jinou SAN) připojeno více uzlů. Souborový systém GFS umožní všem uzlům clusteru, aby přistupovali k jednomu diskovému prostoru jako by byl lokální. Je plně symetrický, takže zde není jeden centrální server, který by byl úzkým místem z hlediska výkonu.

Firma RedHat jej poskytuje spolu s ostatními nástroji (Cluster Administration GUI, GNBD, Conga, Piranha,...) pro tvorbu clusteru pod názvem *Red Hat Cluster Suite*. GFS se se dá s výhodou použít právě s GNBD což je upravené NBD(síťové blokové zařízení) optimalizované pro použití přímo s GFS.

Pro velké clustery a superpočítače s nutností rozložení diskové kapacity na více uzlů je vhodné použít filesystémy jako je např. Lustre[1], PVFS2[2]. Pro malé a střední clustery kde se použije jeden nebo menší počet sdílených disků je vhodnější GFS nebo OCFS2.

OCFS2

Souborový systém OCFS2 (Oracle Cluster File System) vyvíjí firma Oracle vznikl z původního systému OCFS, který byl určen pouze pro běh databáze Oracle RAC (Oracle Real Application Cluster). Je to přímí konkurent GFS. Od verze 2.6.16 je OCFS2 standardní součást jádra. OCFS2 je souborový systém určený pouze pro provoz nad jedním sdíleným blokovým zařízením. Toto blokové zařízení může být sdíleno přes LAN za pomocí NBD (Network Block Devices), iSCSI či ATAoE.

[1] http://www.lustre.org/
[2] http://www.pvfs.org

4 CDN - CONTENT DELIVERY NETWORKS

4.1 CDN - historie a souvislosti

Původní koncept Internetu spočívá na modelu, kdy na jedné straně existuje množina uživatelů (klienti) a na druhé straně stojí servery, které poskytují informace a zajišťují služby. Poslední vývoj Internetu ale ukázal, že i toto schéma bylo překonáno a začal masivně prosazovat provoz typu P2P ("Peer To Peer"), neboli provoz mezi samotnými účastníky Internetu, bez nutnosti služeb poskytovat služby z jednoho centrálního bodu. Podle některých odhadů tvoří dnes takovéto peer to peer aplikace 50-70% celkového objemu přenosu dat v Internetu.

Průkopník této technologie byla služba Napster, která umožnila masové přenosy multimediálních dat. Jeho popularita během několika málo měsíců vzrostla natolik, že vážně ohrozil celý multimediální průmysl a hudební nakladatelé nakonec dosáhli toho, že Napster musel skončit svoji činnost. I přes jeho ukončení ale odstartoval vlnu rozvoje Peer-To-Peer sítí. A stále více se ukazoval potenciál P2P pro další vývoj internetu.

4.2 PlanetLab

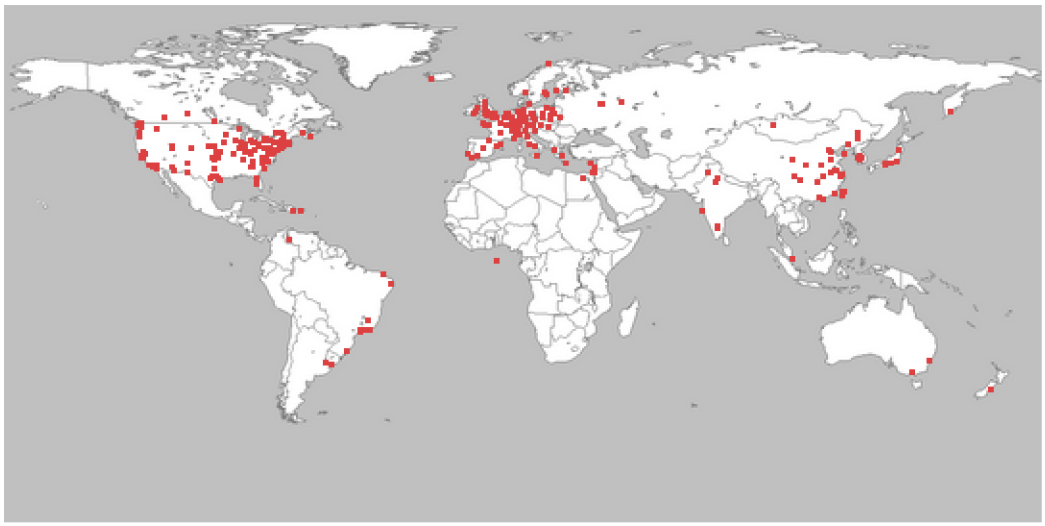
S neustále se rozvíjejícím se Internetem se také mění jeho využívání a stávající protokoly a systémy se stávají v některých ohledech nedostatečné. PlanetLab [1] je organizace, která vznikla s předsevzetím vývoje nových aplikací a protokolů Internetu. PlanetLab vznikla v průběhu roku 2002 spojením několika amerických universit University of California at Berkeley, Princeton University a University of Washington. V následujících obdobích se k nim připojilo mnoho dalších univerzit z celého světa a významných výzkumných pracovišť firem z oblasti IT (HP, Intel, France Telecom, Google), organizace zajišťující provoz Internetu (Internet2-USA, Canarie-Kanada, Cernet-Čína) a další (INRIA-Francie, GIST-Korea).

PlanetLab je otevřena vstupu dalších členů. Členem se může stát každá akademická organizace, která do ní vloží své uzly. Této příležitosti využil i CESNET. Přístup do PlanetLab má tak i ČVUT, Vysokého učení technické v Brně a Masarykova univerzita.

PlanetLab tak poskytuje první globální virtuální síťovou laboratoř, jež si ve svých cílech vytyčila změny Internetu. PlanetLab řeší aplikace budoucnosti, které by měly vyřešit hlavní problémy Internetu. Těch časem přibývá v závislosti na tom, jakým tempem se Internet rozvíjí.

[1] http://www.planet-lab.org

Výhoda sítě PlanetLab spočívá v tom, že umožňuje uživatelům vytvářet nezávislé síťové struktury, které mohou běžet v celé sítí vedle sebe, a vytvářet tak ucelené virtuální vrstvy sítě. Jednotliví uživatelé tak mohou používat stejné uzly bez vzájemného ovlivňování. Jednotlivé vrstvy používají společné uzly, aniž by se přitom jakkoli ovlivňovaly. Uživatel tak může najednou použít i stovky uzlů. Vlastní virtualizace je řešena pomocí implementace VServer[2] což je nástroj pro virtualizaci na úrovni operačního systému.



Obr. 4.1: PlanetLab - rozložení serverů

4.3 CDN - princip

Alternativou ke klasickému již zde zmíněnému rozdělování zátěže může být práve Content Delivery Networks. Jedná se o celkem nový pohled na rozložení zátěže webových serverů mezi více uzlů. CDN je systém serverů geograficky rozložených po internetu, které spolu spolupracují pro zajištění rychlého doručení dat klientovi.

Tyto servery tak představují virtuální vrstvu nad fyzickou infrastrukturou, která funguje na principu proxy serverů. CDN se primárně hodí jen pro data statická nebo měnící se v delších intervalech. Většinu datového provozu CDN tak tvoří statická multimediální data. Typickým využití CDN může být i distribuce aktualizací SW zákazníkům.

Jedna z nejznámějších komerčních CDN je Akamai (http://www.akamai.com/). Tuto síť využívá například Microsoft distribuci dat zákazníkům.

Požadavky uživatelů jsou tak rozprostřeny přeš všechny servery podle požadavků. V případě požadavku na co největší rychlost je uživatel přesměrován na

[2]http://linux-vserver.org/,

nejbližší uzel sítě nebo v případě optimalizace na cenu spojení půjde komunikace po levnějších linkách. Většinou je ale bližší uzel zároveň levnější.

V původním modelech CDN se používá centrální prvek, který přesměrovává požadavky podle určení a koncové uzly pak nemají žádnou vlastní rozhodovací váhu, ale jen poskytují data a v případě, že data samy neobsahují, tak se obrátí na předem definované místo, kde požadovaná data jsou. Vhodnější řešení je přímo použití záznamů v DNS. Na DNS serverech je pak nastavena rychlá expirace záznamů, aby neprobíhalo cachování. Klient tak dostane rovnou adresu farmy, která je mu nejblíže.

4.4 Komerční CDN

Akamai Technologie Akamai [3] původně vznikla jako projekt na MIT [4] a nyjí je lídrem na trhu s CDN. Společnost Akamai disponuje více než 18,000 servery rozmístěnými po celém světě v 70 zemích. K dynamické správě obsahu používá vlastní značkovací jazyk Edge Side Includes.

Akamai používá k rozlišení koncových serverů tzv. ARL (Akamai Resource Locator). ARL se skládá z polí uvedených na schématu 4.2.

Serial značí skmupinu webových objektů doručovaných ze stejných Akamai serverů. Nabívá hodnot 0 až 2047.

Pole **AkamaiDomain** určuje, že obsah půjde ze sítě Akamai a ne z původního serveru.

Type Code definuje způsob nakládání s ARL jako například časy expirace objektu, kontrolování otisků objektu,... Část ARL **Content provider code** určuje zákaznické číslo.

Object data field v závislosti na poli **Type Code** určuje čas expirace, definuje verzi objektu,...

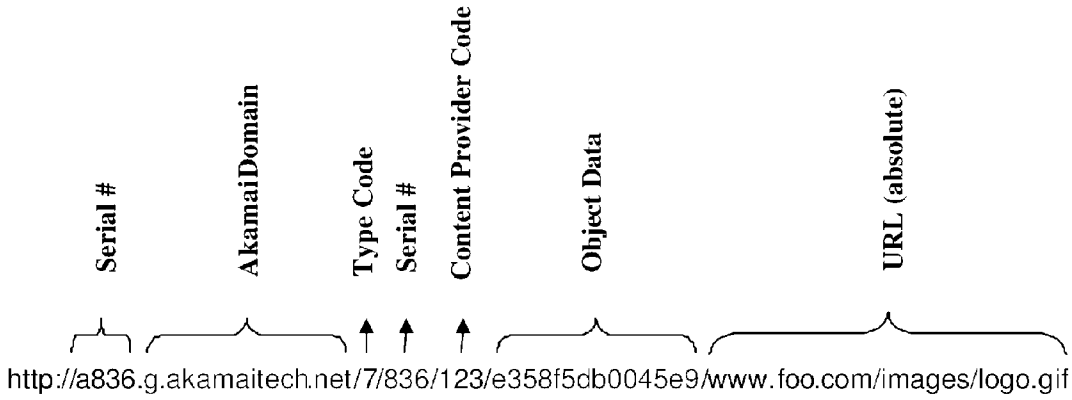
Absolute URL je původní absolutní cesta na serveru poskytovatele.

Mirror Image (http://www.mirror-image.com/) je další globální komerční CDN pro poskytování statických i streamovaných multimedií. Servery má rozloženyy v 22 státech světa a využívají ji společnosti jako třeba Creative, SiteRock,...

Limelight Networks (http://www.limelightnetworks.com/) je CDN pro poskytování videa, hudby a dalších dat, ke které používá servey an 72 lokalitách světa. Podporuje tzv. multivedia na vyžádání (On Demand).

[3] http://www.akamai.com/

[4] Massachusetts Institute of Technology



Obr. 4.2: Způsob skládání URL v Akamai [?]

4.5 Prvky CDN

4.5.1 Přesměrování

Plné přesměrování

V případě plného přesměrování se dotazy na veškerý obsah posílají na CDN. Nevýhoda je ve větším objemu přenášených dat.

Selektivní přesměrování

Poskytovatel obsahu sám určí, které objekty bude distribuovat pomocí CDN. Typicky se používá filtr na velké soubory, obrázky, atd. Výhoda je v dobré kontrole toku dat. Nevýhoda je v nutnosti manuálního nastavení značkování/filtrování obsahu.

DNS

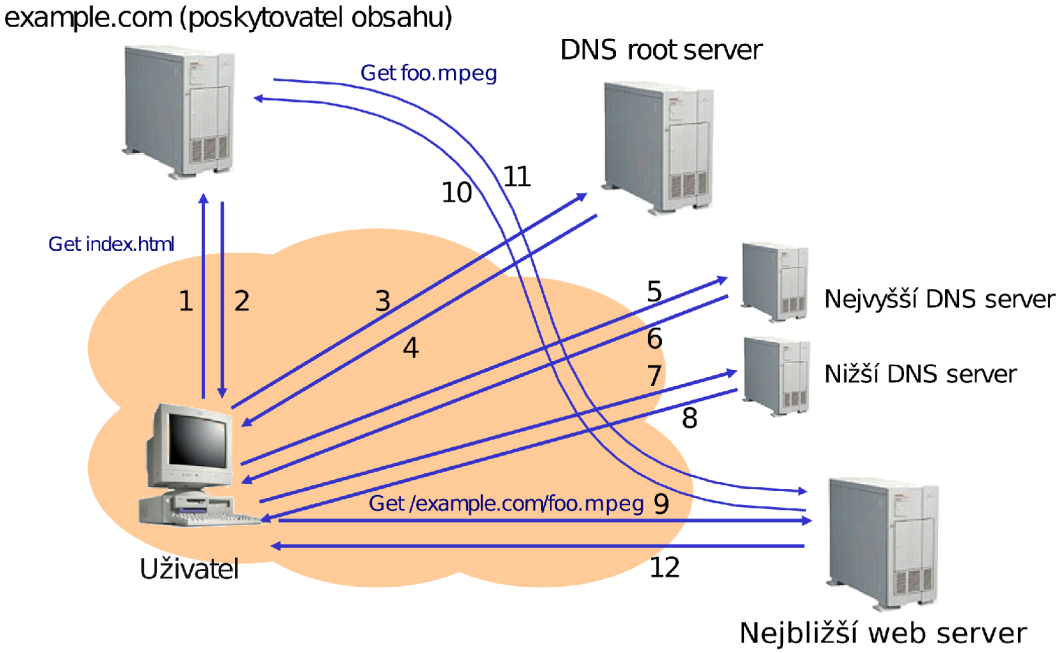
Toto přesměrování ma jistou analogii s již uvedeným DNS Round-Robin rozložení zátěže jen s tím rozdílem, že nepoužijeme cluster webových serverů ale CDN.

Typický průběh průběh DNS přesměrování u velkých CDN probíhá tak, že přijde požadavek od zákazníka na hlavní DNS server v CDN infrastruktuře. Požadavek jde na další bližší DNS server a nakonec se přesměruje na cílový server s požadovaným obsahem. DNS server je tak součástí CDN systému a musí mít přehled o topologii CDN sítě aby vybral nejbližší server a o stavu koncových serverů.

V případě DNS přesměrování můžeme narazit na jeden závažný problém. Rozhodování o cíly přesměrování se provádí na základě toho odkud přišel dns dotaz. Dotaz nejde přímo od klienta, ale ze jmenného serveru ketrý klient pro dotaz použil. To může znamenat problém u větších poskytovatelů internetu, kterí mají jeden DNS

server pro velké oblasti. Klient se totiz totiž může vyskytovat na zcela jiné lokaci než DNS server který použil pro dotaz.

Typický průběh při použití CDN Akamai je znázorněn na obrázku 4.3. Pro redirect je využito DNS. Kroky jsou prováděny podle uvedených čísel. Nejprve uživatel vznese dotaz na http://example.com/index.html vrátí se mu html kód s obsahem (soubor foo.mpeg) směrovaným do sítě pomocí URL rewrite. Klient se dále dotazuje na DNS záznamy. Napřed na root name-server a dále na DNS servery ve struktuře CDN postupně stále blíže klientovi. Pokud cílový server obsahuje replikovaný dokumet tak jej poskytne klientovi pokud ne tak se napřed stáhne z původního serveru.



Obr. 4.3: Typický průběh dotazu v CDN

HTTP

Druhá metoda přesměrování je založena na protokolu HTTP. Klient pošle požadavek na server a ten mu odpoví Request somehow intercepted by redirection service Redirection service forwards user's request to the "best" content server Content served from the content server

4.6 Open Source CDN

4.6.1 CoralCDN

Coral Content Distribution Network je otevřená peer-to-peer síť navržená pro zrcadlení www obsahu na více serverů.

Cíl projektu Coral je zbavit se zátěžových spiček na serverech poskytovatelů obsahu rozložením zátěže mezi všechny účastníky sítě. Nemá za cíl konkurovat velkým komerčním CDN, ale poskytnout CDN menším poskytovatelům obsahu, kterým tak umožní předcházet tzv. **slashdot efektu**[5].

Využívá metody DSHT [6], která spočívá v tvorbě samočinně se organizujících clusterů složených z uzlů které si předávají informace tak, aby se vyvarovali komunikace se vzdálenějšími uzly. Coral je tak rozdělen do zón ze soustředných krůhů. Každá DHT zóna představuje geograficky rozlehlejší území. Pokud je zaznamenáno mnoho přístupů na dva nejstřednější kruhy, tak se provoz začne přesměrovávat na další kruh dále od středu.

Projekt byl otevřen veřejnému testování v roce 2004. Coral hostuje na síti PlanetLab. Zdrojový kód je uvolněn pod licencí GNU GPL.

Protože je CoralCDN zveřejněn pod otevřenou licencí můžeme jej provozovat na vlastních serverech. Pojem CoralCDN neznačí jen tento software, ale již vytvořenou testovací síť, která je tvořena uzly PlanetLabu. Testovací síť CoralCDN používá k přesměrování metodu DNS. Pokud chceme přistoupit ke stránce example.com přes CDN, stačí jen změnit url na example.com.nyud.net.

4.6.2 Globule

Globule je další druh řešení realizace content delivery network. Globule je modul pro webový server Apache, které poskytuje možnost replikovat dokumenty po sítí dalších serverů pro účinné rozložení zátěže. Zajišťuje konzistenci dat na backend serverech a přesměrovává požadavky klientů. Je šířen pod svobodnou BSD licencí. Projekt se vyvíjí na Vrije Universiteit v Holandsku.

Globule řeší tyto funkce CDN:

- Replikace dat mezi backend servery.
- Přesměrování požadavku klienta na správný backend server. Pro tuto funkci využívá HTTP nebo DNS redirect.

[5] přetížení a paralizování stránek v důsledku uvedení url na vysoce navštěvovaných portálech(slashdot.com, del.icio.us,...)

[6] Distributed Sloppy Hash Table

- Odolnost pro ti výpadku backend serveru. Globule průběžně hlídá stav serverů a pokud některý neběží nebo je nesprávně nakonfigurován přestane na něj přesměrovávat. Podporuje také vytvoření plné kopie hlavního uzlu pro případ jeho havárie.
- Monitorování chování celé sítě serverů a agregaci logů z celé sítě. Ukládání statistik s možnostmi filtrování různých parametrů.
- Adaptivní replikace s možností rozdílného chování podle typu obsahu.
- Replikace dynamického obsahu. Podporuje PHP a MySQL databáze.

Přestože se v dokumentu [13] uvádějí metody výběru nejvhodnějších cílů přesměrování na základě RTT a dalších parametrů, ukázalo se, že poslední vydaná verze tyto metody ještě nepodporuje.

5 PRAKTICKÉ TESTY

5.1 Testovací prostředí

V této části budou popsány postupy zprovoznění jednotlivých nástrojů a praktické zkušenosti z instalace a provozu. Pokud nebude uvedeno jinak byly instalace prováděny na Linux/GNU distribuci CentOS [1] http://www.centos.org/. Jedná se o klon distribuce RedHat (Red Hat Enterprise Linux, RHEL), který vzniká kompilací originálních zdrojových kódů RHEL a tím je zajištěna plná binární kompatibilita. Přestože je RHEL komerčně nabízená distribuce je šířená pod licencí GNU/GPL, je tak možné po odstanění licencovaných prvků (firemní loga, atd.) provést volné šíření této distribuce. Přicházíme tak ale o podporu produktu ze strany společnosti RedHat.

Na rozdíl od druhého komunitního clonu RedHatu, distribuce Fedora Core, nevzniká CentOS jako prostředí pro test nových technologií, ale naopak zachovává všechny vlastnosti původního RHEL (vývojový cyklus, updaty). RHEL je vysoce stabilní a spolehlivou distribucí, která je určena primárně k serverovému nebo podnikovému nasazení, kde se hledí spíše na stabilitu než na nejnovější technické vymoženosti. Bezpečnostní updaty vydávané pro RHEL jsou převáděny s malým spožděním i do CentOS. CentOS je zkompilován pro platformy i386, x86_64, ia64, s390x, ppc.

5.2 LinuxHA

5.3 DRBD

Instalace

Ke zprovoznění Drbd mámě dvě volby, buď použijeme instalaci z balíků distribuce, nebo přímo ze zdrojových kódů. Instalaci z balíků provedeme jednoduše pomocí `yum install drbd kmod-drbd`.

Pokud chceme použít nejnovější vydanou verzi, je často výhodné instalovat přímo ze zdrojového souboru.

```
yum install make gcc glibc kernel-dev kernel-headers
wget http://oss.linbit.com/drbd/drbd-8.2.5.tar.gz
tar -xzf drbd-8.2.5.tar.gz

cd drbd
```

[1] The Community ENTerprise Operating System

```
make clean
make KDIR=/usr/src/linux

make install
make install-tools
```

Příprava hdd

Jelikož drbd pracuje jako přídavná vrstva nad v blokovým zařízením, je nutné pevný disk připravit. Drbd je možné provozovat přímo nad oddílem pevného disku, nad RAID oddílem, LVM nebo EVMS. Přestože je možné použít i tzv. loop device používané například pro šifrování tak se to nedoporučuje.

Nejjednodušší řešení je tedy vytvořit na obou serverech stejně velké partition, které se dále použijí pro Drbd replikaci. Konkrétní strategie rozdělení disku se bude řídit konkrétním účelem nasazení clusteru. Pro jednoduchý HA webový server je výhodné vytvořit dva oddíly, na kterých pak připravíme dva nezávislé drbd disky. Tato konfigurace nám pak dovolí používat na prvním serveru vlastní http server a na druhém serveru například databázový server. Rozložíme tak vhodně zátěž serveru a v případě poruchy na jednom serveru se služba migruje na druhý server.

Nástroje pro správu DRBD

Drbdadm je vysokorúrovňový nástroj, který vytváří vrstvu nad drbdsetup a drbdmeta. Jedná se o základní program k nastavení Drbd. Nastavené parametry si bere ze souboru `/etc/drbd.conf`.

Drbdsetup slouží k nastavení jaderného modulu Drbd. Jedná se o nízkoúrovňový program, který přímo mění parametry běžícího modulu.

Drbdmeta je určen k práci s metadaty. Dovoluje vytvořit, zálohovat nebo obnovit meta strukturu dat. Obvykle jej není potřeba používat.

Nastavení sítě

Replikace probíhá přes TCP/IP, proto je nejvýhodnější použít propojení přes ethernet. Je vhodné použít dedikovanou gigabitovou ethernet kartu, propojit kvalitní kaneláží a otestovat propustnost například pomocí nástroje `iperf`[2] abychom předešli možným problémům s výkonem. Protože replikace probíhá pouze mezi dvěma servery, je vhodné použít propojení kříženým kabelem a vyhneme se tak snížení propustnosti na routerech a přepínačích.

[2]http://dast.nlanr.net/Projects/Iperf/

Nesmíme také zapomenout povolit na firewalu buď porty, na kterých drbd komunikuje, nebo rovnou veškerou komunikaci na rozhraní, které slouží k propojení DRBD.

Nastavení zdrojů

Základní konfigurační soubor Drbd je `/etc/drbd.conf`. Je důležité, aby tento soubor byl identický na obou strojích v clusteru.

Nejjednodušší možné nastavení může vypadat takto:

```
global {
  usage-count yes;
}
common {
  protocol C;
}
resource drbd0 {
  on Alfa {
    device    /dev/drbd0;
    disk      /dev/sda3;
    address   10.1.1.1:7789;
    meta-disk internal;
  }
  on Omega {
    device    /dev/drbd0;
    disk      /dev/sda3;
    address   10.1.1.2:7789;
    meta-disk internal;
  }
}
resource drbd1 {
  on Alfa {
    device    /dev/drbd1;
    disk      /dev/sda4;
    address   10.1.1.1:7790;
    meta-disk internal;
  }
  on Omega {
    device    /dev/drbd1;
    disk      /dev/sda4;
    address   10.1.1.2:7790;
    meta-disk internal;
  }
}
```

V tomto nastavení jsme vytvořili nové virtuální blokové zařízeni `/dev/drbd1` nad oddílem `/dev/sda3` pevného disku. První server(Alfa) má ip adresu rozhraní pro synchronizaci 10.1.1.1 a druhý (Omega) 10.1.1.2.

Spuštění synchronizace

Spustíme drbd:
`/etc/init.d/drbd start`
Zkontrolujeme zda se zavedl modul drbd:

`lsmod | grep drbd`
Na obou serverech vytvoříme resource:
`drbdadm create-md drbd_disc`
Na primarním pak spustíme synchronizaci
`drbdadm --overwrite-data-of-peer primary all`
Pomocí `cat /proc/drbd` si můžeme zobrazit stav sychronizace a uvidíme jeho průbeh:

```
Version: 8.0.6 (api:86/proto:86)
SVN Revision: 2093 build by test@test, 2008-02-02 12:41:17
 0: cs:SyncSource st:Primary/Secondary ld:Consistent
    ns:122084 nr:0 dw:662016 dr:782689 al:165 bm:174 lo:19 pe:67 ua:82 ap:0
        [===>................] sync'ed: 19.2% (540188/662008)K
        finish: 0:01:45 speed: 5,080 (4,872) K/sec
```

Délka prvotní synchronizace bude závislá na velikosti disku a rychlosti vašeho hardware.

Na vytvořeném zařízení `/dev/drbd1` pak vytvoříme souborový systém např. takto: `mkfs.ext3 /dev/drbd1`

Ovládání a kontrola stavu

Stav modulu Drbd můžeme v reálném čase kontrolovat v souboru `/proc/drbd`, kde jsou veškeré informace o stavu replikace. Výstup může vypadat třeba takto:

```
# cat /proc/drbd
version: 8.0.6 (api:86/proto:86)
SVN Revision: 3048 build by test@test, 2008-02-02 01:18:03
0: cs:Connected st:Primary/Secondary ds:UpToDate/UpToDate C r---
ns:313 nr:5 dw:318 dr:1848 al:0 bm:62 lo:0 pe:0 ua:0 ap:0
resync: used:0/31 hits:0 misses:0 starving:0 dirty:0 changed:0
act_log: used:0/127 hits:313 misses:0 starving:0 dirty:0 changed:0
1: cs:Connected st:Secondary/Primary ds:UpToDate/UpToDate C r---
ns:0 nr:456 dw:456 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
resync: used:0/31 hits:0 misses:0 starving:0 dirty:0 changed:0
act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

V tomto případě výstup značí, že na serveru tomto serveru jsou nadefinované dva Drbd disky. První je v modu primary a druhý v modu secondary. Toto znamená, že první disk se synchronizuje směrem z tohoto serveru na druhý a další disk ve směru opačném.

5.4 Linux-HA

Instalace

Instalaci můžeme provést jednoduše z distribučních balíků pomocí `yum install heartbeat`, případně zkompilovat přímo ze zdrojových balíků stažených ze stránek projektu [3].

[3] http://www.linux-ha.org

zkratka	význam
cs	connection state
st	node state (local/remote)
ld	local data consistentency
ns	network send
nr	network receive
dw	disk write
dr	disk read
pe	pending (waiting for ack)
ua	unack'd (still need to send ack)
al	access log write count

Tab. 5.1: Význam zkratek ve výstupu `cat /proc/drbd`

Nastavení

Jak už bylo dříve uvedeno Linux-HA může fungovat ve dvou modech, ve starším který podporuje pouze dva uzly v clusteru a v novějším módu verze 2.x. Jelikož jsme při použití replikace přes Drbd omezeni pouze na dva uzly, je často výhodnější použít při kombinaci Drbd a Linux-HA právě starší způsob konfigurace pro jeho jednoduchost a snadnou iplementaci.

Základná nastavení se provádí v souboru `/etc/ha.cf`

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 10
udpport 694
udp     eth0
node    bart.example.com
node    meggy.example.com
```

Nastavení spouštěných služeb se provede editací souboru `/etc/ha.d/haresources`

```
bart.example.com \
        IPaddr::1.2.3.4/24/eth0 \
drbddisk::drbd0 \
        Filesystem::/dev/drbd0::/storage/web::ext3 \
        httpd \
        MailTo::nekdo@nekde.com
meggy.example.com \
        IPaddr::10.10.10.1/24/eth0 \
drbddisk::drbd1 \
        Filesystem::/dev/drbd1::/storage/db::ext3 \
        postgresql \
MailTo::nekdo@nekde.com
```

Tímto nastavením jsem zabezpečili, že na serveru Alfa se přiřadí ip 1.2.3.4, přepne drbd0 do stavu primary, připojí disk s http daty a spustí Apache http server. Na

serveru Omega se přiřadí lokální ip adresa pro přístum k DB z http serveru, drbd1 se nastaví na primary a připojí disk s DB daty. Nakonec se spustí vlastní PostgreSQL server. Volba `MailTo` zabezpečuje, že se při přepnutí služeb posílá informace o migraci na uvedený mail.

5.5 RedHat Cluster Suite

Protože se jedná o komplexní balík programů, je vhodné vybrat distribuci, která RedHat Cluster Suite standardně obsahuje. Jak už název napovídá, nejlepší podporu dosáhneme na distribuse RHEL, ale tam musíme počítat s velmi vysokými pořizovacímy náklady. Pokud se spokojíme s provozem bez oficiální podpory výrobce, tak je vhodné zvolit CentOS. RHCS obsahují i další rozšířené distibuce (Debian, Ubuntu a další).

5.5.1 LVS

Jak už bylo v dřívějších kapitolách uvedeno, balík RedHat Cluster Suite obsahuje k řešení rozložení zátěže nástroj LVS. RHCS k nastavení LVS obsahuje program **piranha**. Nastavení je pak možné pomocí webového rozhraní nebo manuálně přímo v souboru **/etc/sysconfig/ha/lvs.cf** což se ale nedoporučuje.

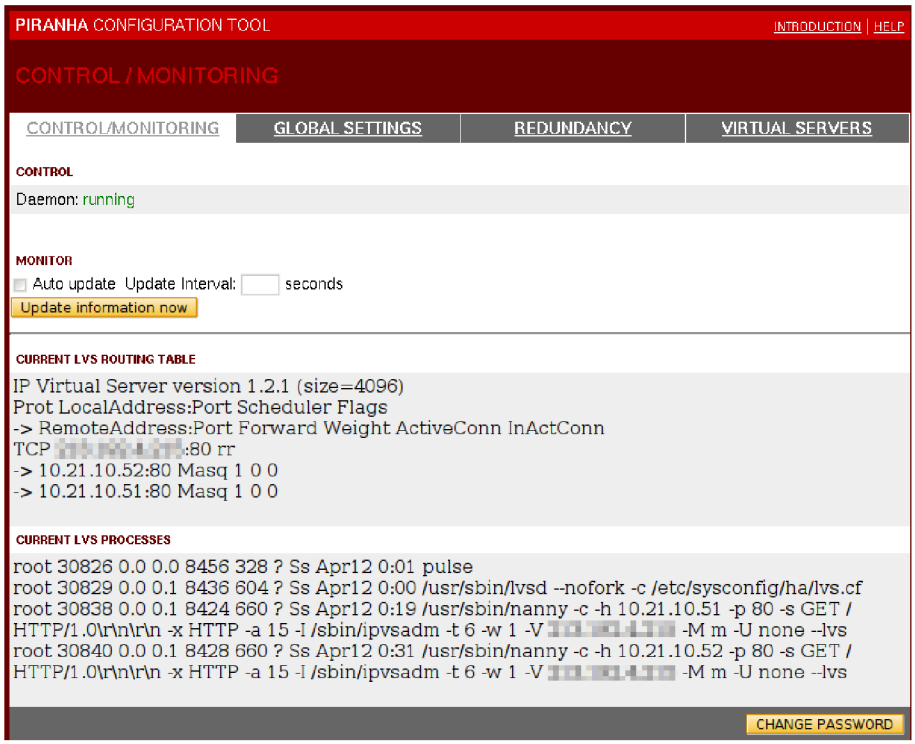
Pro testování jsem použil server Dell PowerEdge 1950 s CPU Intel Xeon E5335 (8x1,995GHz) a 8GB RAM, na kterém běží VmWare ESX. Na tomto serveru se nainstalovali virtuální servery s CentOS 5. Mezi jednotlivými VM byla vytvořená virtuální síť pro interní komunikaci v rámci klusteru.

Instalaci provedeme pomocí **yum install piranha**. Pak spustíme službu pomocí **/etc/init.d/piranha-gui start**. Nastavíme heslo pro uživatele *piranha* příkazem **/usr/sbin/piranha-passwd**. K webovému rozhraní přistoupíme na adrese http://localhost:3636

Po přihlášení se zobrazí základní informace o běžícím LVS balanceru a jeho parametry. Základní nastavení LVS provedeme v přes nabídku **Global settings**, kde nastavíme veřejnou ip adresu LVS serveru. Typ sítě nastavíme na hodnotu NAT. Hodnotu **NAT Router IP** nastavíme na adresu z lokálního rozsahu a nastavíme na virtuální rozhraní **eth1:1** spřažené s rozhraním připojeným na lokální virtuální síť mezi nody clusteru .

Na záložce **Virtual Servers** vytváříme jednotlivé balancované služby. Nastavení viz obrázek 5.5. Hodnota **Virtual IP Address** je veřejná ip adresa, na které chceme, aby byla služba dostupná.

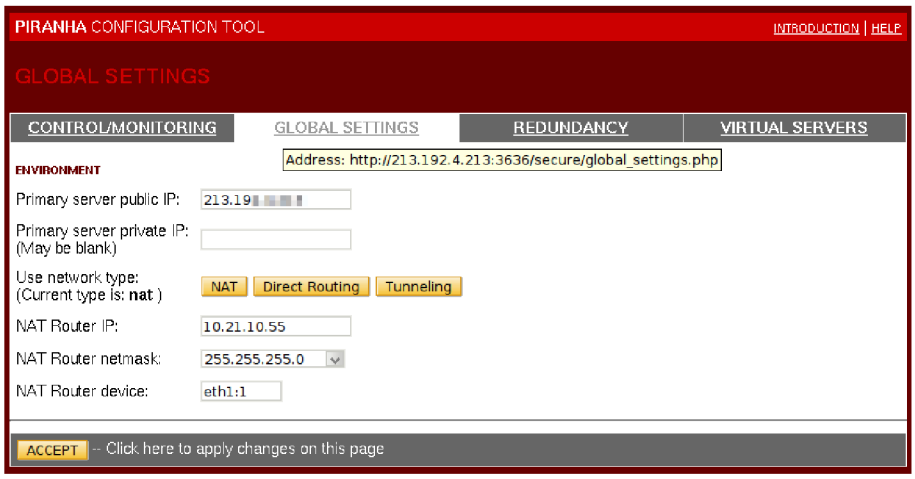
Na závěr je potřeba mastavit ip adresy reálných serverů, kam se jednotlivé požadavky budou směřovat (obr. 5.6).



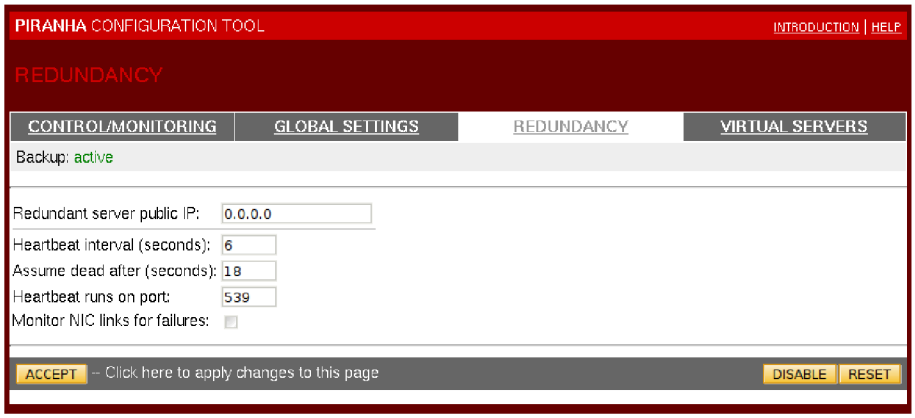
Obr. 5.1: Piranha - základní obrazovka

Výsledné nastavení pomocí GUI se tak uložilo do **/etc/sysconfig/ha/lvs.cf** takto:

```
serial_no = 82
primary = 213.19.x.x
service = lvs
backup_active = 0
backup = 0.0.0.0
heartbeat = 1
heartbeat_port = 539
keepalive = 6
deadtime = 18
network = nat
nat_router = 10.21.10.55 eth1:1
nat_nmask = 255.255.255.0
debug_level = NONE
virtual http {
     active = 1
     address = 213.19.x.x eth0:1
     vip_nmask = 255.255.255.0
     port = 80
     send = "GET / HTTP/1.0\r\n\r\n"
     expect = "HTTP"
     use_regex = 0
     load_monitor = none
     scheduler = rr
     protocol = tcp
     timeout = 6
     reentry = 15
```



Obr. 5.2: Piranha - globální nastavení

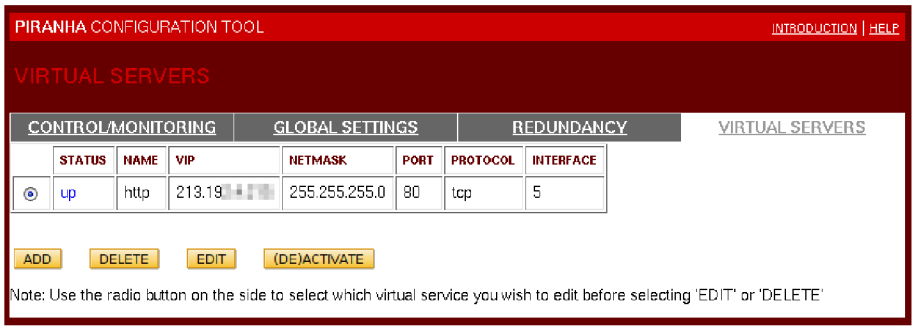


Obr. 5.3: Piranha - nastavení redundance

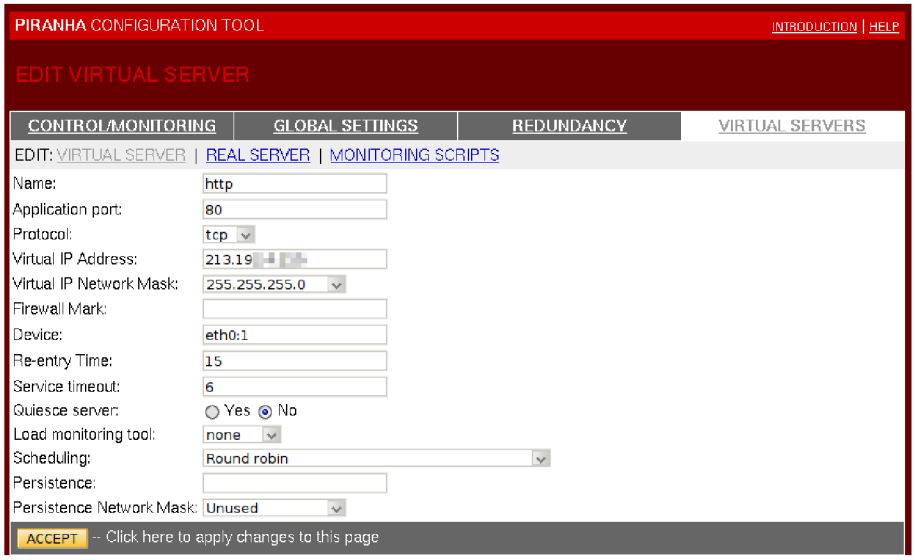
```
    quiesce_server = 0
    server nod1 {
        address = 10.21.10.51
        active = 1
        weight = 1
    }
    server nod2 {
        address = 10.21.10.52
        active = 1
        weight = 1
    }
}
```

Výkonostní test

Pro otestování funkčnosti a výhodnosti použití LVS na vytížených serverech jsem použil testovací program **ab**, který je součástí HTTP serveru Apache. Abych na serverech vytvořil nějaké vytížení při zpracovávání http požadavků a test se tak



Obr. 5.4: Piranha - výpis balancovaných služeb

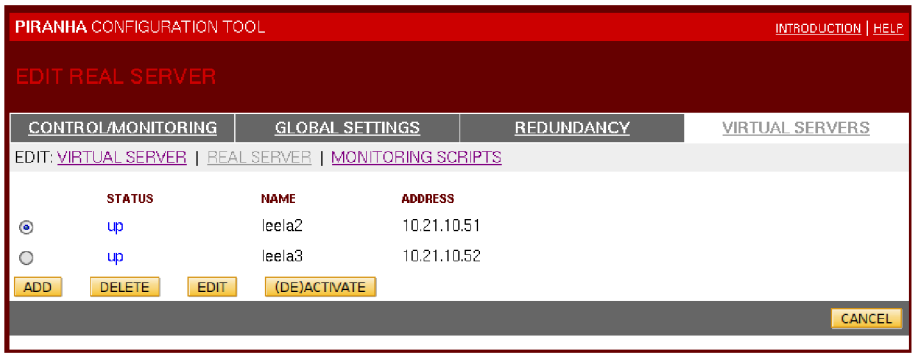


Obr. 5.5: Piranha - nastavení balancované služby

podobal reálným podmínkám, tak se jako cílové url použil PHP script původně určený pro otestování výkonu PHP (http://www.free-webhosts.com/php-benchmark-script.php) ketrý uměle vyvářel zátěž serveru.

Výsledek testu kdy byly aktivní oba servery:

```
Server Software:        Apache/2.2.3
Server Hostname:        213.192.4.215
Server Port:            80

Document Path:          /bench.php
Document Length:        932 bytes

Concurrency Level:      10
Time taken for tests:   19.104836 seconds
Complete requests:      1000
Failed requests:        24
   (Connect: 0, Length: 24, Exceptions: 0)
Write errors:           0
Total transferred:      1124144 bytes
HTML transferred:       932144 bytes
```



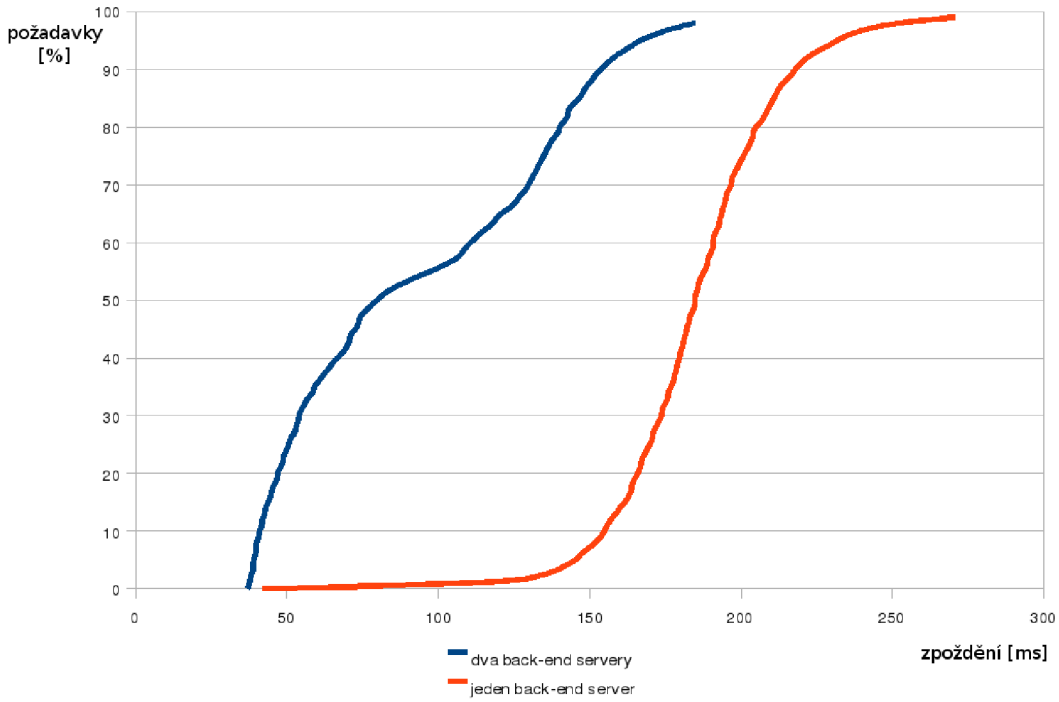
Obr. 5.6: Piranha - nastavení ip adres reálných back-end serverů

```
Requests per second:    52.34 [#/sec] (mean)
Time per request:       191.048 [ms] (mean)
Time per request:       19.105 [ms] (mean, across all concurrent requests)
Transfer rate:          57.42 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        1    4  94.8      1    2999
Processing:    40  185  30.4    184     527
Waiting:       39  184  29.1    183     456
Total:         42  189  99.7    185    3190

Percentage of the requests served within a certain time (ms)
  50%    185
  66%    194
  75%    201
  80%    205
  90%    218
  95%    232
  98%    251
  99%    271
 100%   3190 (longest request)
```

Poté se jeden z back-end serverů vypnul a výsledky vypadaly takto:

```
Server Software:        Apache/2.2.3
Server Hostname:        213.192.4.215
Server Port:            80

Document Path:          /bench.php
Document Length:        932 bytes

Concurrency Level:      10
Time taken for tests:   9.722484 seconds
Complete requests:      1000
Failed requests:        515
   (Connect: 0, Length: 515, Exceptions: 0)
Write errors:           0
Total transferred:      1124623 bytes
HTML transferred:       932623 bytes
Requests per second:    102.85 [#/sec] (mean)
Time per request:       97.225 [ms] (mean)
```

```
Time per request:       9.722 [ms] (mean, across all concurrent requests)
Transfer rate:          112.93 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        1    1   0.6      1       6
Processing:    36   94  68.3     79    1639
Waiting:       36   94  68.2     78    1639
Total:         37   96  68.3     80    1640

Percentage of the requests served within a certain time (ms)
  50%     80
  66%    124
  75%    135
  80%    140
  90%    154
  95%    167
  98%    185
  99%    217
 100%   1640 (longest request)
```



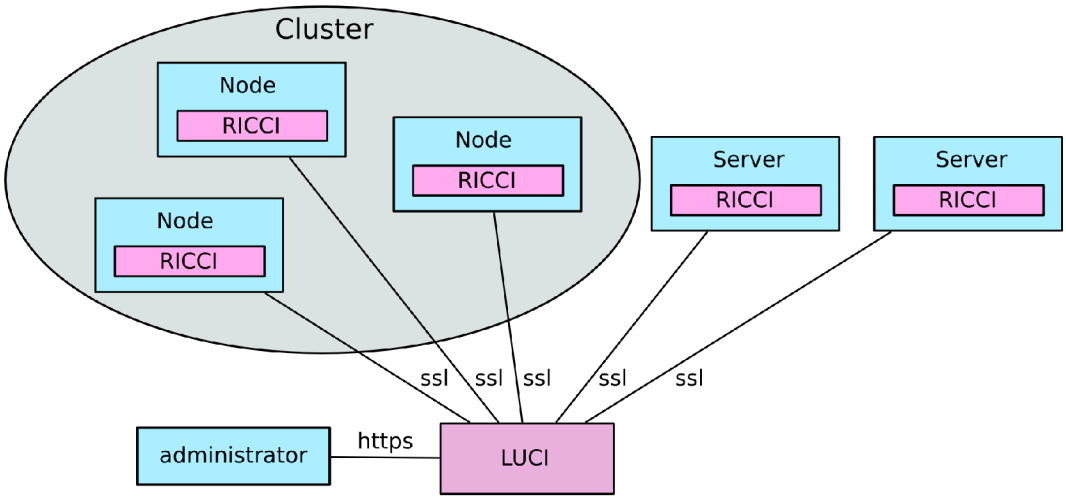
Obr. 5.7: LVS - Graf rozložení doby odezvy serveru

V případě současného běhu obou serverů jsme dosáhli přenosové rychlosti 112.93 kB/s a průměrná doba obsloužení jednoho požadavku byla 96 ms. V případě běhu pouze jednoho serveru byly hodnoty 57kB/s a 187 ms. Z výsledků lze tak jasně vidět, že zátěž se v případě běhu obou serverů za LVS rovnoměrně rozložila a dosáhli jsme přibližně dvojnásobného výkonu.

5.5.2 High Availabilty v RHCS

Pro zajištění vysoké dostupnosti nám Redhat Cluster Suite nabízí více nástrojů. Ve verzi 5 byla snaha tyto nástroje centralzovat na jedno místo a zpřehlednit a usnadnit tak správu HA clusteru. Vznikl tak nástroj Conga, který má za cíl hlavně sjednotit správu diskového prostoru napříč celým clusterem. Sjednocuje tak např. nástroje red-hat-cluster, system-config-cluster, deploy-tool, system-config-lvm a umožňuje tato nastavení provádět z jednoho centrálního webového rozhraní.

Architektura Conga se skládá ze dvou démonů. Jeden (Luci) tvoří centrální bod, na který se administrátor připojuje pomocí webového prohlížeče a z démonů (Ricci) běžících na jednotlivých uzlech, které provádějí jednotlivá cílová nastavení.



Obr. 5.8: Conga - architektura systému

Inicializaci serveru Luci provedeme příkazem `luci_admin init` a službu spustíme `service luci start`. K rozhraní se nalogujeme na adrese https://localhost:8084. Samotné nastavení není triviální přesto jsou však kroky intuitivní. Kompletní popis rozhraní překračuje rozsah této práce a doporučuji postupovat podle [11].

5.5.3 GFS

5.6 Apache - mod_proxy_balancer

Mod_proxy_balancer je modul do Apache který rozšiřuje možnosti modulu mod_proxy o možnost dynamického rozložení zátěze mezi více Apache serverů. Samotný modul mod_proxy je určen k vytvoření několika drůhů http proxy serveru. V rámci optimalizace a zvyšování výkonu je u vysoce zatěžovaných serverů tento modul vhodné použít k vytvoření tzv. *Reverse Proxy*.

Oproti klasické proxy, která vytváří novou vrstvu mezi aplikací a uživatelem co nejblíže koncovému uživateli je reverzní proxy na straně serveru s HTTP serverem.

Při vyřizování požadavků na vysoce zatěžovaných serverech vykazuje velké zpoždění už jen přečtení HTTP hlavičky a pokud všechny požadavky vyřizuje Apache se všemi moduly vzniká tak velké zatížení a používá se tak zbytečně velké množství operační paměti, protože každý požadavek představuje vlastní vlákno.

Jako reverzní proxy se pak použije menší a efektivnější program, který načte požadavek od uživatele a po načtení jej pošle na aplikační server. V praxi je tak vhodné jako proxy použít třeba Apache s nastaveným modelem *worker* a vypneme co možná nejvíce přídavných modulů abychom omezily velikost programu v paměti nebo jiný jednoduchý http server (LightHTTP, Nginx). Požadavky z proxy pak mohou být posílány na localhost, kde na jiném portu běží Apache se standartním paměťovým modelem a všemy moduly(PHP, Perl,...). Pomocí mod_rewrite je vhodné nastavit, aby se požadavky na statické objekty (obrázky) vyřizovaly přímo na proxy a požadavky na generovaný obsah (php, perl, python,...) posílal na backend server. Ještě výhodnější varianta je pokud proxy a backend http server budeme provozovat na vlastních fyzických serverech a tím vytvoříme jednoduchý cluster.

Jednoduchou reverse proxy nastavíme takto:

```
ProxyRequests Off
ProxyPass / http://localhost:8080/
ProxyPassReverse / http://localhost:8080/
```

5.6.1 Konfigurace LB

Nejprve musíme zabezpečit načtení potřebných modulů v httpd.conf pomocí directivy LoadModule. Moduly musí být obsaženy v distribučním balíku Apache nebo musí být správně zkompilovány při manuálně kompilovaném Apache.

```
LoadModule proxy_module modules mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
```

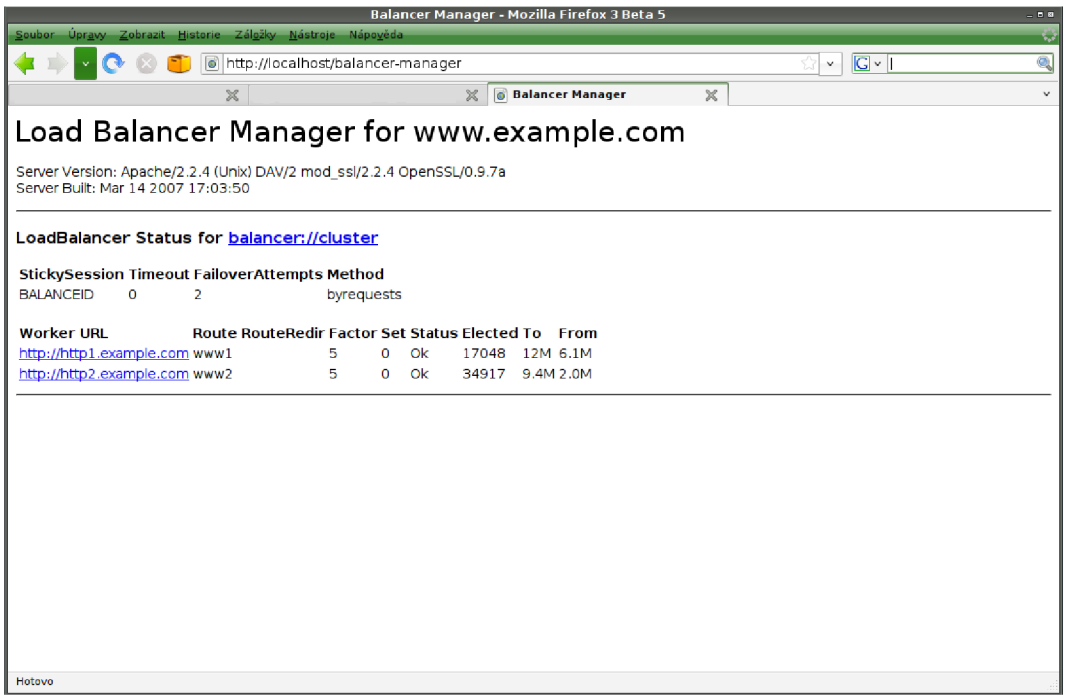
Základní konfigurace mod_proxy_balancer pro testovací virtuální doménu nastavíme takto:

```
NameVirtualHost *
<VirtualHost *>
        ServerName www.example.com
        ServerAlias example.com
        DocumentRoot /var/www/
        ProxyRequests Off

        <Proxy *>
          Order deny,allow
          Allow from all
        </Proxy>

        ProxyPass /balancer-manager ! # zajistí, že se nebude provádět LB u cesty /balancer-manager
        ProxyPass / balancer://cluster/ stickysession=BALANCEID nofailover=On
```

```
        ProxyPassReverse / http://http1.example.com/
        ProxyPassReverse / http://http2.example.com/

        <Proxy balancer://cluster>
          BalancerMember http://http1.example.com  lbfactor=1 route=http1
          BalancerMember http://http2.example.com  lbfactor=1 route=http2
          ProxySet lbmethod=byrequests
        </Proxy>

        <Location /balancer-manager>
          SetHandler balancer-manager
          Order deny,allow
          Allow from localhost
        </Location>
</VirtualHost>
```

Tímto jsem vytvořili LB pro virtuální doménu example.com. U této domény se zátěž bude rozkládat mezi servery http1.example.com a http2.example.com s metodou rovnoměrného rozložení podle počtu požadavků. V případě rokládání zátěže na základě velikosti prošlých dat se použije `lbmethod=bytraffic`. Na tuto volbu pak navazuje direktiva `lbfactor`, která nastavuje poměr rozkládání mezi jednotlivé servery.

Modul obsahuje i jednoduché GUI pro kontrolu a změnu parametrů LB za chodu. Toto rozhraní pak najdeme na adrese http://example.com/balancer_manager. Protože je na této adrese možné měnit důležité parametry, je nutné tuto cestu zabezpečit heslem, nebo povolit jen pro určité adresy klientů. V ukázkové konfiguraci je povolen přístup jen z localhost.



Obr. 5.9: Zobrazení stavu mod_proxy_balancer pomocí balancer-manager

5.6.2 Zachování session

Pokud chceme zabezpečit, aby za určitých okolností byly požadavky směrovány na určité back-end servery využijeme k tomu direktivu `stickysession`. Při zapnutí této volby dochází při čtení požadavku směrování podle obsahu cookie se jménem nastaveným v `stickysession`. Generování této cookie a její změny můžeme provádět například v aplikaci (Java, php,...), nebo přímo na serveru pomocí `mod_rewrite`.

V případě, že chceme použít `mod_rewrite`, tak na jednotlivé back-end servery přidáme tuto konfiguraci.

```
RewriteEngine On
RewriteRule .* - [CO=BALANCEID:balancer.http1:.example.com]
```

Na druhý server:

```
RewriteEngine On
RewriteRule .* - [CO=BALANCEID:balancer.http2:.example.com]
```

Tímto přidáme cookie **BALANCEID** s obsahem **balancer.http1:.example.com**. Na jednotlivých serverech se tak přidá řetězec s jeho názvem. Toto zajistí, že se požadavky od jednoho uživatele budou směrovat stále na jeden server po dobu uložení cookie v prohlížeči.

5.7 CoralCDN

5.7.1 Testování výkonu

Pro test se využilo již vytvořeného testovacího provozu projektu CoralCDN. Testování výkonu se provedlo tak, že se z jednotlivých světových lokací testoval přístup na server umístěný v Praze. Použilo se testovacího programu ab. Test se spouštěl s těmito parametry:

```
ab -e export_coral -n 500 -c 5 http://example.com.nyud.net/index.html
```

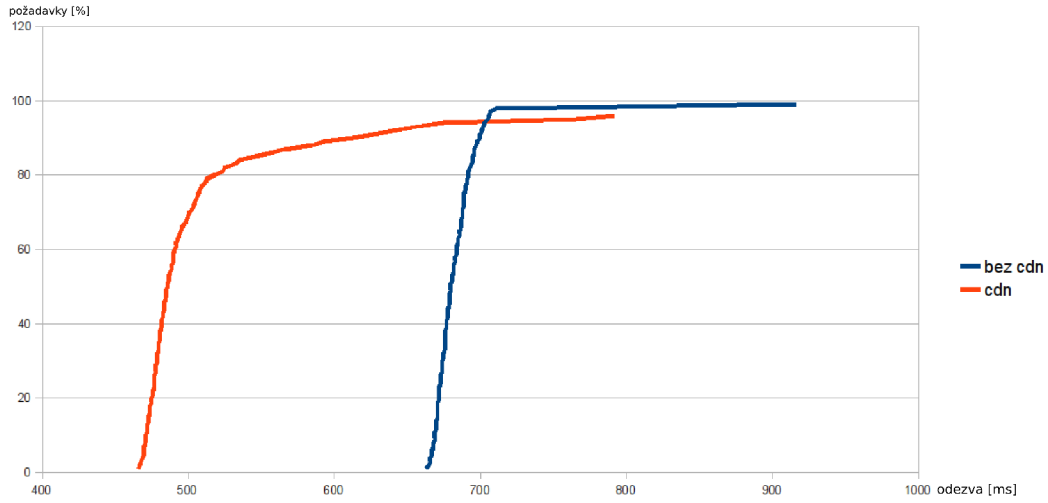
Cílová stránka měla velikost 13kB a test se spustil vždy napřed na přímé url a potom na url směrující na cdn. Po hodině se test opakoval. Tento postup se opakoval desetkrát a výsledek se zprůměroval. Na výsledných grafech je vidět, že zpoždění při použití CDN se pohybovalo v řádově menších hodnotách, jelikož se požadavek přesměroval na geograficky bližší server. U grafu přístupu z uzlu v Moskvě se zrychlení neprojevilo, ale spíše zhoršilo. Toto je pravděpodobně způsobeno tím, že v okolí Moskvy nebyl vhodný server, na který by se požadavek přesměroval, nebo byl server přetížený, což se v síti PlanetLab stává velmi často.

Výsledky měření přenosové rychlosti korespondují s měřením zpoždění. Při použití CDN došlo ke zvýšení rychlosti. Jen u měření z Moskvy opět došlo ke zhoršení.

Tyto výsledky ukázaly, že CDN přesměrovává na bližší servery a tudíz zvýší výkon. Dá se tedy i předpokládat, že CDN bude chránit před přetížením hlavního serveru špičkových zátěžových rázech a rozloží zátěž při tzv. SlashDot efektu.

lokace	rychlost s cdn [kB/s]	rychlost bez cdn [kB/s]
Moskva, RU	109	308
Sao Paulo, BR	119	93
Austin, TE	144	389
Tokio, JP	389	144

Tab. 5.2: Porovnání přenosových rychlostí



Obr. 5.10: Odezva na serveru v Brazilii

5.8 Globule

Cílem testová Globule bylo ověřit možnosti použití této CDN pro globální poskytování obsahu s využitím HTTP serveru Apache. Z důvodu ověření funkce globální CDN jsem pro testování zvolil prostředí PlanetLab.

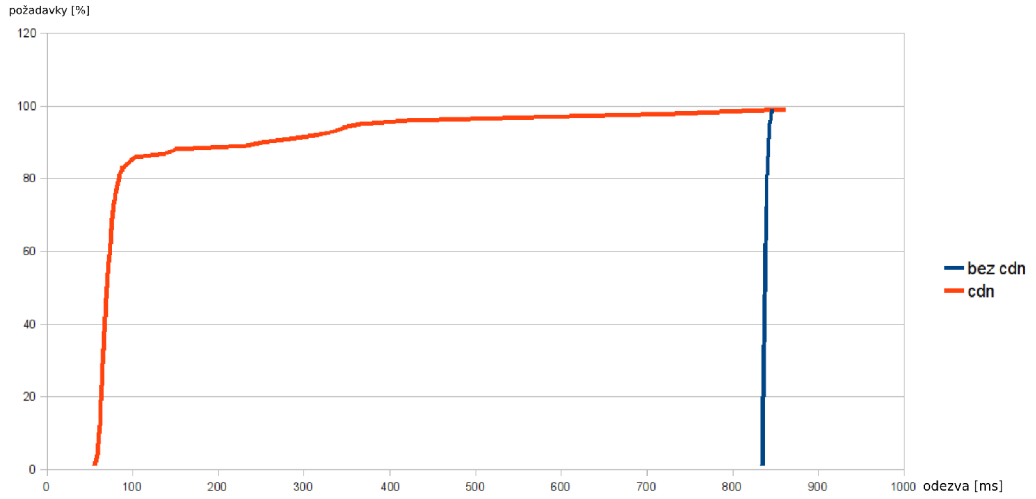
Pro test jsem vytvořil topologii uvedenou na obr. 5.15. Na serveru umístěném v Praze byl hlavní (master) server, který obsahuje aktuální data a na dalších lokalitách běžely podpůrné (slave) servery, které mají za úkol rychlejší distribuci dat po světě.

Jelikož v sítích globálního rozměru má největší vliv na výkonnost RTT[4], tak jsem je před testy pro představu změřil. Uvedené hodnoty RTT v tabulce 5.3 jsou měřeny ze serveru v Praze na jednotlivé světové servery.

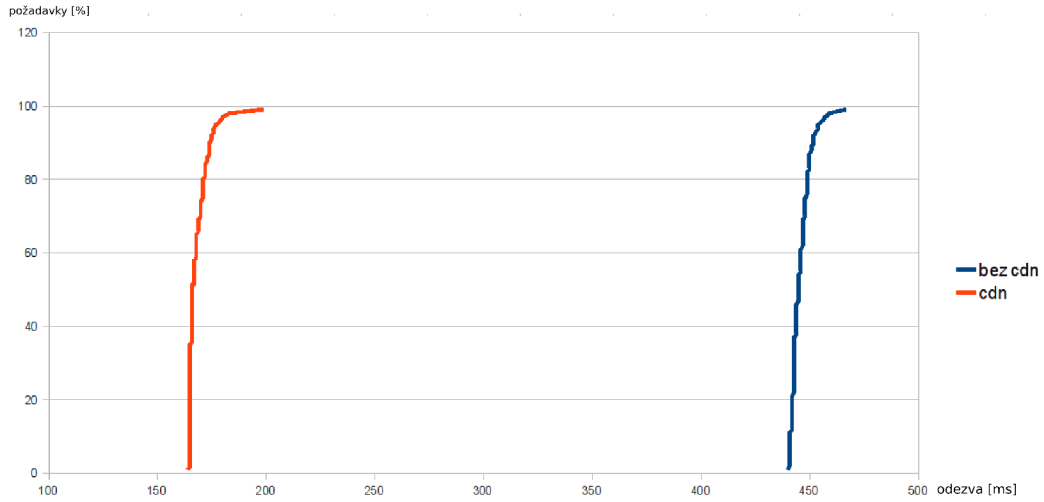
5.8.1 Instalace

Jak už bylo dříve uvedeno Globule je modul k serveru Apache. Instalace je možná jen jako dokompilování modulu, nebo je na webových stránkách projektu ke stažení

[4]RoundTripTime



Obr. 5.11: Odezva ze serveru v Tokiu



Obr. 5.12: Odezva ze serveru v Austinu

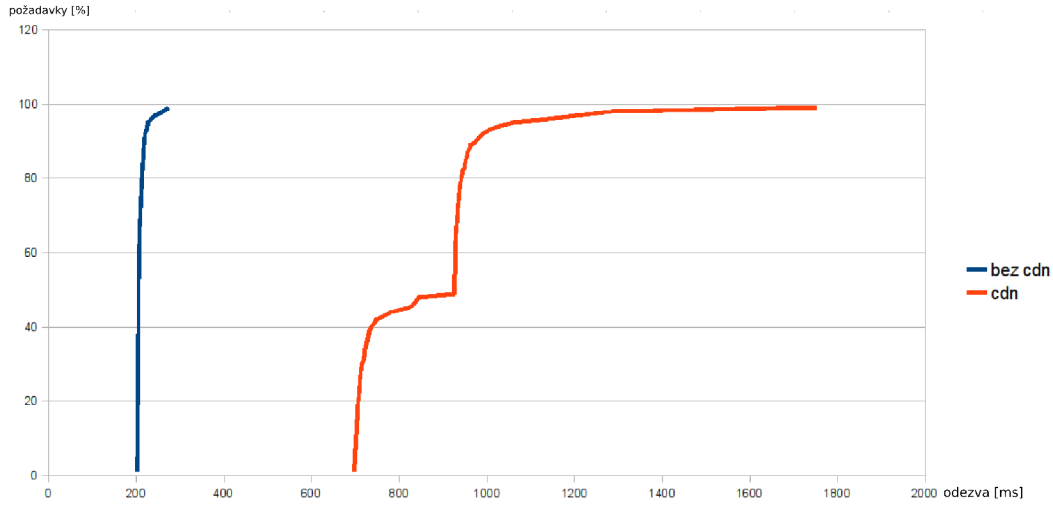
zdrojový balík obsahující Apache, Globule a další podpůrné programy a nástroje. Pro otestování jsem z důvodu potencionální větší odladěnosti zvolil tuto variantu.

Kompilaci jsem z důvodu nutnosti instalace více balíčků a nezatěžování uzlu v PlanetLabu provedl na dedikovaném stroji s čistou instalací Fedora 4. Tato distribuce byla zvolena z toho důvodu, že je použita i na uzlech PlanetLabu.

Při kompilaci se ukázalo, že je potřeba doinstalovat některé balíky, na kterých má Globule závislosti. Před kompilací je tedy potřeba doinstalovat toto:

```
yum install whic gd libpng-devel dialog expat-devel \
  openssl-devel strace pcre-devel flex \
  gdb db4-devel libxml2-devel libtool gdbm-devel \
  gcc-c++ perl-DBD-MySQL  gd-devel xorg-x11-libs  libjpeg libpng
```

Samotné stažení balíku a spuštění kompilace:



Obr. 5.13: Odezva ze serveru v Moskvě

lokace	sever	RTT [ms]
Moskva, RU	pl2.grid.kiae.ru	80
Sao Paulo, BR	planetlab2.larc.usp.br	240
Berkeley, CA	planetlab11.millennium.berkeley.edu	180
Sydney, AU	planetlab1.it.uts.edu.au	522
Austin, TE	planetlab3.csres.utexas.edu	144
Tokio, JP	planetlab2.iii.u-tokyo.ac.jp	288

Tab. 5.3: Doba odezvy serverů ve světě z master serveru(Praha)

```
wget 'http://www.globule.org/download/installer.sh'
  chmod u+x installer.sh
  sh ./installer.sh --keep-build
```

Standartně se Globule nainstaluje do adresáře `/usr/local/globule`, který pak stačilo přenést na jednotlivé servery v PlantetLabu.

```
rsync -e "ssh -i /home/cesnet_vutbr1/.ssh/pl_key" \
  -av /usr/local/globule \
  cesnet_vutbr1@planetlab11.millennium.berkeley.edu:/usr/local
```

Na jednotlivých serverech bylo ještě potřeba doinstalovat balíky kvůli závislostem.

```
yum install xorg-x11-libs  libjpeg libpng
```

5.8.2 Konfigurace

Nastavení Apache serveru se provádí standarně v souboru httpd.conf, který se v našem případě nachází v `/usr/local/globule/etc`. Protože port 80 na uzlech PlanetLabu je již využitý pro interní účely, je potřeba změnit port Apache serveru na jiný.

Obr. 5.14: Globule testovací topologie

```
Listen 8888
```

Před testem se pro přehlednost konfigurace nastavily alias záznamy na DNS serveru testovací domény na jednotlivé CDN servery:

```
...
cdn IN A~195.113.161.83
texas.cdn IN A~128.83.122.181
berkeley.cdn IN A~169.229.50.13
brazil.cdn IN A~143.107.111.195
...
```

Pro test jsem vytvořil na master serveru nový virtualhost.

```
<VirtualHost *>
ServerName cdn.mikulka.info:8888
DocumentRoot /usr/local/globule/www
    <Location "/">
    GlobuleReplicate on
    GlobuleManagedBy -
    GlobuleRedirectionMode HTTP
#   GlobuleDefaultRedirectPolicy static
    GlobuleDefaultRedirectPolicy RR
#   GlobuleDefaultRedirectPolicy AS
    GlobuleDefaultReplicationPolicy Invalidate
    GlobuleReplicaIs http://texas.cdn.testdomena.cz:8888/  heslo
    GlobuleReplicaIs http://brazil.cdn.testdomena.cz:8888/  heslo
    GlobuleReplicaIs http://tokio.cdn.testdomena.cz:8888/  heslo
    GlobuleReplicaIs http://berkeley.cdn.testdomena.cz:8888/  heslo
    GlobuleReplicaIs http://sydney.cdn.testdomena.cz:8888/  heslo
    </Location>
</VirtualHost>
```

Direktiva GlobuleReplicaIs deklaruje jeden nebo více serverů, na které se bude provádět replikace. Na konci se nastaví heslo, které musí být zhodné na obou serverech, tzv sdílené tajemství pro ověření autentičnosti serveru.

Uvedené adresy musí korespondovat s nastavením na slave serverech včetně správného nastavení VirtualHostu a portu.

Volba *GlobuleRedirectionMode* nastavuje způsob přesměrování na slave servery. Volby jsou

- OFF
- HTTP aktivuje přesměrování pomocí HTTP redirektu (302)
- DNS zapne interní DNS server pomocí kterého se bude provádět redirect, vyžaduje port 53
- BOTH aktivuje HTTP i DNS přesměrování

Nastavení způsobu rozhodování o přesměrování provádí GlobuleDefaultRedirectPolicy

- RR zapne Round-Robin přesměrování, rovnoměrné přesměrování mezi jednotlivými servery bez optimalizace
- AS je metoda kdy se pro přesměrování použijí informace o autonomních oblastech routovacího protokolu BGP. Vyžaduje nastavení voleb *GlobuleBGPDataFile* a *GlobuleBGPReloadAfter*
- static vždy vrátí první server v seznamu, pouze pro testování

Pro spuštění přesměrování pomocí AS je potřeba nejprve připravit soubor s informacemi o routování ze stránek routeviews.org a nastavit direktivu `GlobuleBGPDataFile /etc/oix-full-snapshot-latest.dat`

```
cd /usr/local/globule
wget ftp://ftp.routeviews.org/oix-route-views/2008.04/oix-full-snapshot-latest.dat.bz2
bzip2 -d oix-full-snapshot-latest.dat.bz2
```

Direktiva *GlobuleDefaultReplicationPolicy* definuje způsob replikace. Toto nastavení se týká pouze nových souborů. Pro starší soubory je typ replikace vybírán automaticky na základě přístupových logů. Použil jsem volbu *Invalidate*. Tato volba zabezpečí, že pokud se soubor na primárním serveru změní, tak se na slave serveru označí pro synchronizaci a při vyžádání klientem se synchronizuje z master serveru.

Na slave serverech pak konfigurace vypadá nasledovně.

```
<VirtualHost *>
    ServerName texas.cdn.mikulka.info:8888
    DocumentRoot /usr/local/globule/www/
    <Location "/">
      GlobuleReplicaFor http://cdn.mikulka.info:8888/  heslo
    </Location>
</VirtualHost>
```

Adresář DocumentRoot musí existovat a můsí mít práva pro zápis uživatelem pod kterým běží Apache.

Bylo vysledováno, že pokud změníme konfiguraci je pro správnou synchronizaci výhodné smazat adresář **.htglobule** v **DocumentRoot** adresáři. Pokud se to neprovede, v některých specifických případech dojde k nesprávné synchronizaci na slave servery.

5.8.3 Výsledky testování

Původní záměr bylo otestovat schopnost práce Globule jako globální CDN. Po instalaci a výše uvedené konfiguraci se prováděly serie testů, které ukázaly nekteré nedostatky Globule. Pro správnou funkci globální CDN je jedna z nejdůležitější správná funkce přesměrování na vhodné cílové servery. V některých materiálech o projektu Globule se uvádí, že obsahuje metody pro dynamické rozhodování na základě RTT a dalších parametrů. V praxi se ukázalo, že Globule tyto funkce zatím nemá funkční.

První testy probíhaly se zapnutou volbou `GlobuleDefaultRedirectPolicy RR`, která zapne přesměrování založené na metodě Round-Robin. Při této konfiguraci Globule správně přesměrovával cyklicky mezi všema nastavenými servery. Pokud měl některý ze serverů poruchu Globule to pomocí pravidelného testování zjistil a deaktivoval přesměrování na vadný server. Tato funkcionalita umožnuje dosáhnou určité úrovně vysoké dostupnosti i při poruchách na serverech. Metoda Round-Robin je sice vhodná na jednoduché rozložení zátěže, ale pro globální CDN je nedostačující. Round-Robin totiž způsobí že požadavky jsou směrovány i na velmi vzdálené servery a to vede k markantnímu snížení výkonu provádění těchto požadavků hlavně kvůli velkým zpožděním na dlouhých linkách. S velkým RTT totiž vznikájí výkonostní problémy nejen na úrovni HTTP protokolu, ale i na samotném TCP. TCP používá potvrzovací mechanizmus a systém oken který při velkých RTT vykazuje snížení rychlosti a je potřeba pro rychlé přenosy nasadit některé z metod pro zvýšení výkonu na linkách s velkým RTT. K tomu slouží například používání oken větších než standatních 64kB nebo novější metody kontroly zahlcení HYBLA, BIC, CUBIC, a další.

Pro optimalizaci směrování na bližší servery obsahuje Globule v konfiguraci volbu `GlobuleDefaultRedirectPolicy AS`. Při této volbě se rozhodování má řídit pomocí dodaných routovacích tabulek z routeviews.org. Rozsáhlé testování napříč sítí PlanetLab ale ukázalo, že tato volba není funkční. Přesměrovávací agent v Globule stále podle všeho přesměrovával náhodně. Pokusy o nalezení chyby i za použití logu a debug výstupu se nezdařily. I přes nefunkčnost pokročilejších metod přesměrování je tu stále možnost přesunout logiku rozhodování na vyšší vrstvu. Rozhodování tak může řídit třeba samotný PHP script na základě IP adresy klienta tak, že bude

měnit url generované stránky na adresy určitých backend-serverů.

Výsledky testů tedy ukázaly, že Globule se ve stávající verzi nehodí na globální nasazení jako CDN. Tento fakt ale nic nemění na věci, že Globule se stále dá velmi dobře použít jako CDN na méně rozlehlých sítích, protože má kvalitně vyřešené metody synchronizace souborů mezi servery. Globule je stále velmi dobře použitelná jako alternativa klasického řešení rozložení zátěže. Globule taky řeší zabezpečení vysoké dostupnosti díky integrovaným pravidelným kontrolám běhu back-end serverů.

5.8.4 Apache JMeter

Pro testy výkonu byl původně vybrán nástroj jMeter. Tento Open-source testovací nástroj vznikl jako součást projektu jakarta od Apache. Program je napsán v jazyce java. Podporuje textový i grafický mód. Jeden z důvodů proč byl vybrán je podpora distribuovaného testování. Je totiž možné z jednoho hlavního uzlu s testerem jMeter spouštět testy na vzdálených serverech s jMeterem spuštěným s režimu server, přičemž se výsledky agregují do hlavního uzlu. Tato možnost dovoluje vytvářet velkou zátěž na testovaný server překračující možnosti testování z jednoho bodu.

JMeter obsahuje široké možnosti nastavení pro testy webových aplikací. Pomocí JMeteru vytvoříte pro aplikaci specifický test, který simuluje reálnou zátěž během provozu. Můžete specifikovat kolik klientů současně bude posílat požadavky, na jaké soubory, HTTP hlavičky, cookies.

Konfigurace se provádí přímo pomocí syntaxe XML, nebo se použije grafické rozhraní. Pro náš test se vložily tyto elementy:

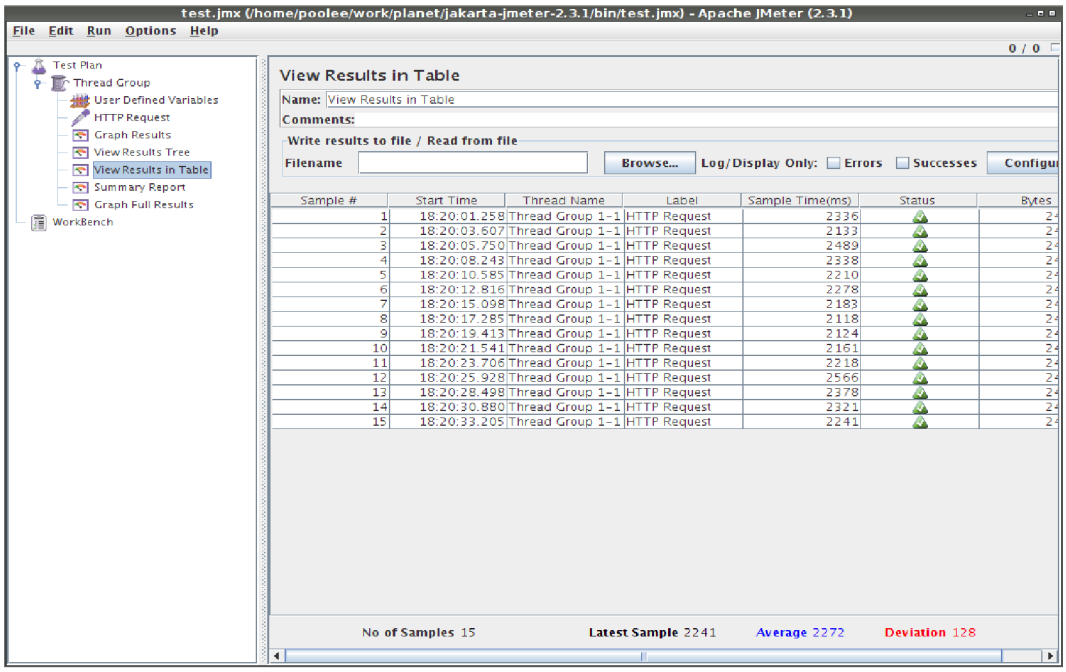
- ThreadGroup - určuje počet vláken a počet opakování testu
- HTTP Request - nastavuje parametry HTTP dotazu (url, port,...)
- View Result in Table - v tabulce zobrazí časy zpracování jednotlivých dotazů
- Summary report - zobrazí celkové výsledky, průměrné zpoždění

5.8.5 Problémy s planetLabem

Jelikož uzly PlanetLabu běží na platformě Linux-VServer tak to přináší oproti běhu na nativním fyzickém stroji určité problémy. Při delším běhu Apache s modulem Globule v náhodných intervalech přestal program odpovídat. V logu se objevovali záznamy o vyčerpání systémových zámků.

```
[emerg] Resources problem#1: absolute minimum number of system locks
```

Dočasně se problém odstraní smazáním semaforů systému takto:



Obr. 5.15: Rozhraní programu jMeter

```
# zjistí semafory uživatele apache
ipcs -s | grep apache

# smaže semafory uživatele apache
ipcs -s | grep apache | perl -e 'while (<STDIN>) { @a=split(/\s+/); print `ipcrm sem $a[1]`}'
```

Další problémy jsem zaznamenal se samotným výkonem uzlů PlanetLabu. Jelikož jeden uzel sdílí více uživatelů, je tak výkon rozdělěn mezi ně a při testech výkonu jsme nemohli zaručit stabilní výsledky. Některé servery byly dokonce tak zatíženy, že se musely pro test použít jiné uzly. Load serveru na některých uzlech byl přes 20, což je neakceptovatelné.

Potíže se ukázaly také při měření pomocí jMeter. Aplikace je napsána v Javě tudíž je poměrně náročná na velikost obsazené RAM a na virtuálním serveru PlanetLabu často padala. Kvůli náročnosti a nestabilitě jMeteru na uzlech PlanetLabu jsem nakonec výsledky jMeteru neuvedl pro jejich nejednoznačnost a použil jen nástroj *ab*.

6 ZÁVĚR

Dokument se zabývá možnostmi implementace serverového prostředí s důrazem na řešení vysoké dostupnosti poskytovaných služeb (high-avalability) a rozložení serverové záteže (load-balancing) s využitím volně dostupných open-source nástrojů v prostředí operačního systému GNU/Linux.

Rozebírá použití různých typů clusterových technologií k dosažení vyšší dostupnosti a vyššího výkonu. Pro realizaci High Avalability clusteru popisuje použití projektu Linux-HA.

Pro dosažení vyššího výkonu byly navrhnuty metody rozložení zátěže pomocí několika postupů, popsány úskalí jejich nasazení do ostrého provozu a oblasti použití jednotlivých typů LB. Konkrétní výběr systému pro rozložení zátěže záleží hlavně na správném výběru pro konkrétní účel použití. Například pro řešení LB u HTTP serveru, kde je potřeba mít plnou kontrolu nad tím, jakým způsobem budeme data rozdělovat doporučuji modul mod_proxy_balancer do serveru Apache.

Jeden z hlavních problémů použití více aktivních uzlů clusteru je režim nakládání a přístupu ke sdíleným datům. Byly navrhnuty možnosti řešení tohoto problému jak po stránce hardware, tak po stránce replikačních software a sdílených souborových systémů. Jako řešení pro malé HA clustery se ukázal vhodný projekt DRBD. Pro kompexnější řešení je vhodné použít např. SAN se sdíleným souborovým systémem. Pro toto prostředí jsou vhodnými kandidáty souborové systémy GFS a OCFS2.

V kapitole věnující se virtualizaci byly nastoleny možnosti kombinace systémů s vysokou dostupností s virtualizačními technologiemi a ukázalo se, že toto spojení má perspektivu a své výhody. Virtualizace bylo použito i při dalším testování v kombinaci VmWare ESX + Redhat Cluster Suite.

Alternativou ke klasickému rozdělování zátěže jsou Content Delivery Networks. CDN je systém serverů geograficky rozložených po internetu, které spolu spolupracují pro zajištění rychlého doručení dat klientovi. Přestože jsou CDN doménou převážně komerčních společností, bylo úkolem otestovat implementaci CDN pomocí open source software a jejich vhodnost pro použití jako globální CDN.

Většina popsaných software a systémů bylo ověřeno a byla popsána jejich instalace a konfigurace. Ověřily se tak funkce DRBD pro replikaci dat, LVS a modul pro Apache server mod_proxy_balancer pro rozložení zátěže. LVS jsme použili v rámci balíku programů Redhat Cluster Suite, který obsahuje i další podpůrné nástroje a představuje komplexní řešení pro HA a LB clustery. Výkonostní testy prokázaly učinnost řešení HTTP serveru s rozložením zátěže mezi dva HW servery.

Byly implementovány CDN založených na projektu Globule a CoralCDN. Pro testování těchto systémů se použilo vývojové sítě PlanetLab, ke které má přístup i VUT v Brně. Sestavila se síť složená ze serverů rozložených geograficky po světě pro

ověření funkce v globálním měřítku. Testy ukázaly některé nedostatky těchto systémú a ukázaly, že nejsou přímými konkurenty zavedených komerčních sítí. Globule ve stávajícím stavu se nepříliš hodí pro geograficky rozlehlé sítě z důvodu špatně fungující optimalizace přesměrování požadavků. U CoralCDN výkonostní testy prokázaly přínos sítí pro doručování obsahu.

LITERATURA

[1] Espen Braastad, *Management of high availability services using virtualization.* Oslo University College, Department of Informatics, 2006

[2] Werner Fischer, *Implementation of a Disaster Resilient Linux Cluster with Storage Subsystem Based Data Replication.* Oslo University College, Department of Informatics, 2004

[3] Chokchai Leangsuksun, Tong Liu, Yudan Liu, Stephen L. Scott, Richard Libby, and Ibrahim Haddad, *Highly Reliable Linux HPC Clusters.* Computer Science Department, Louisiana Tech University, Enterprise Platforms Group, Dell Corp., Oak Ridge National Laboratory, Intel Corporation, Ericsson Research, 2004

[4] Sven Ingebrigt Ulland, *High-Level Load Balancing for Web Services.* UNIVERSITY OF OSLO Department of Informatics, 2006

[5] Yee Jiun Song, Venugopalan Ramasubramanian, Emin Gun Sirer, *Optimal Resource Utilization in Content Distribution Networks.* Dept. of Computer Science, Cornell University, Ithaca, NY 14853

[6] Berry van Halderen, Guillaume Pierre, *Globule User Manual* [online]. Online manuál k programu Globule. Dostupné z URL: <http://www.globule.org/docs/doc.pdf>.

[7] Lars Ellenberg, *Data Redundancy By DRBD* [online]. Dostupné z URL: <http://wiki.linux-ha.org/DataRedundancyByDrbd>.

[8] *Red Hat Global File System* [online].
Dostupné z URL: <http://www.redhat.com/gfs/>.

[9] *Oracle Cluster File System (OCFS2) User's Guide* [online].
Dostupné z URL: <http://oss.oracle.com/projects/ocfs2/>.

[10] Christoph Mitasch, Werner Fischer, *High availability clustering of virtual machines possibilities and pitfalls.* Paper for the talk at the 12th Linuxtag, May 3rd-6th, Wiesbaden/Germany 2006

[11] Florian Haas, Philipp Reisner, Lars Ellenberg *The DRBD User's Guide.*
Dostupné z URL: <http://www.drbd.org/users-guide/index.html>.

[12] *Configuring and Managing a Red Hat Cluster.*
Dostupné z URL: <http://www.redhat.com/docs/manuals/csgfs/browse/>.

[13] Guillaume Pierre, Maarten van Steen *Globule: A Collaborative Content Delivery Network.*
Dostupné z URL: <http://www.globule.org/publi/GCCDN_commag2006.html>.
IEEE Communications Magazine, 2006

[14] Akamai Technologies *Fast Internet Content Delivery with FreeFlow.* 2000

[15] The Apache Software Foundation *Apache HTTP Server Version 2.2 Documentation.*
Dostupné z URL: <http://httpd.apache.org/docs/2.2/>. 2000

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

HA High Availability - vysoká dostupnost

VM Virtual Machine - virtuální počítač

NFS NFS - Network File System

FLOPS floating-point operations per second

MASTERNOD hlavní uzel, dohlíží na ostatní uzly v clusteru

RESOURCE služba o kterou spouštíme v HA

STONITH Shoot The Other Node In The Head

LB Load Balancer

GFS Global File System

DNS Domain Name System

STONITH Shoot The Other Node In The Head

MTBF Mean Time Before Fail

MTBR mean Time Before Repair

PDU Power Distribution Unit

NIC Network Interface Card

HPC High Performance Computing

URI Uniform Resource Identifier

RHCS RedHat Cluster Suite

HTTP Hyper Text Transfer Protocol

CDN Content Delivery Network

RTT Round Trip Time

BGP Border Gateway Protocol

NAT network Address Translation