TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: M 2612 – Elektrotechnika a informatika
Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

Webové aplikace s vysokou dostupností

High available web application

Diplomová práce

Autor: **Jan Tryzna**
Vedoucí práce: Ing. Jiří Hnídek
Konzultant: Ing. Ondřej Raška, MITON CZ, s.r.o.

V Jablonci nad Nisou 1. 4. 2007

Zadání diplomové práce

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum : 31. 8. 2007

Podpis:

Poděkování

Rád bych poděkoval vedoucímu diplomové práce Ing. Jiřímu Hnídkovi za korekci diplomové práce, firmě MITON CZ, s.r.o. a jejím zaměstnancům za poskytnutí prostředků a konzultací pro realizaci diplomové práce, rodině a přátelům za podporu.

Anotace

Teoretickou část diplomové práce tvoří přehled principů a metod pro realizaci systému s vysokou dostupností. Nejprve jsou vymezeny termíny vysoká dostupnost a cluster s vysokou dostupností. Dále je uveden popis algoritmů, které se využívají pro rozkládání zátěže mezi jednotlivé uzly clusteru spolu s charakteristikou metod přeposílání jednotlivých požadavků v systémech s vysokou dostupností. Tuto část uzavírá přehled možností pro správu a distribuci dat v clusterových systémech podporující běh webové aplikace.

Praktická část diplomové práce řeší návrh a realizaci systému s vysokou dostupností pro běh webové aplikace. Obsahuje popis použitých nástrojů a jejich konfigurací. Systém se skládá z clusteru, který úzce spolupracuje se sítí pro doručování obsahu. Cluster tvoří DNS, webové a databázové servery a nástroje zajišťující vysokou dostupnost. Pro realizaci DNS serveru byl použit nástroj BIND. Apache byl nasazen jako webový server a systém MySQL byl použit pro správu databáze. HAProxy je nástroj, který byl nasazen, aby zajišťoval rozklad zátěže a monitorování stavu jednotlivých uzlů. Konzistenci a dostupnost databáze zajišťuje MMM MySQL replication manager. Na serverech v síti pro doručování obsahu byl nasazen FTP server ProFTPD pro nahrávání dat a webový server Lighttpd pro jejich distribuci do webové aplikace.

Diplomová práce přináší komplexní řešení sytému s vysokou dostupností pro běh webové aplikace. Popsané nástroje jsou dostupné jako Open source software a šířeny pod obecně veřejnou licencí GNU.

Klíčová slova: Vysoká dostupnost, Cluster, Rozklad zátěže, Síť pro doručování obsahu

Annotation

The theoretic part of diploma project is made by list of principes and methods for implementation of high availability computers systems. There are definitions of terms high availability and high available cluster in the first instance continue with description of algorithms which are used for balancing load and request routing between each nodes of cluster. This part is closed by list of options for data administrating and distribution inside of systems which support functioning web application.

The practical part of diploma project deals with technical design and execution of high availability system made for functioning web application. Used tools and their configuration are including. System is making up by cluster which closely cooperates with content delivery network. Cluster makes DNS, webs&databases servers and tools which are provide high availability. BIND was use as tool provide DNS server. Apache was apply for web server and for database administration was chosen MySQL system. Load balancing and state monitoring of each node has provided by HAProxy. MMM MySQL replication manager manage accessibility and consistency of database. In content delivery network's servers is apply ProFTPD for upload dates and for distribution was chosen web server Lighttpd.

Diploma project brings complex solution of high available system for functioning web application. Described tools are available as open source software which are distribute under general public licence GNU.

Key words: High availability, Cluster, Load balancing, Contend delivery network

Obsah

Seznam obrázků

Úvod

Celosvětová síť internet v dnešní době představuje největší zdroj informací a zábavy. Nabízí prostor pro prezentaci všech stránek lidského konání. Pomocí různých webových aplikací je možné ovládat bankovní účty, obchodovat na burze, sledovat televizní přenosy nebo jen prezentovat fotografie či video a mnoho dalšího. S rostoucí přenosovou rychlostí roste i dostupnost informací a fyzická vzdálenost již nehraje téměř žádnou roli. Zvyšující se nároky webových aplikací mají za následek neustálý vývoj technologií pro udržení celého sytému v chodu. Nejen tato skutečnost mě vedla k výběru diplomové práce s názvem „Webové aplikace s vysokou dostupností“.

Termín vysoká dostupnost se stále častěji objevuje v požadavcích na realizaci webové aplikace. Z pohledu uživatele to znamená, že výpadek některého ze serverů neovlivní běh aplikace a vykonávání jednotlivých dotazů. Tento požadavek však v první řadě představuje hardwarovou i softwarovou redundanci částí systému. Pro vlastní zajištění vysoké dostupnosti je nutné použít nástroje, které umožňují monitorování jednotlivých uzlů, rozkládání zátěže a směrování požadavků na aktivní prvky systému.

Cílem diplomové práce je seznámit se s možnými způsoby rozkladu zátěže webových aplikací na jednotlivé servery a možnostmi zajištění vysoké dostupnosti celého systému. Následně návrhnout a realizovat řešení odolné proti výpadku jednotlivých uzlů clusteru a chování navrženého systému ověřit na konkrétní webové aplikaci.

1. Vysoká dostupnost aplikací

Pojem vysoká dostupnost, anglicky high availability (HA), v oblasti informačních technologií představuje schopnost systému a jeho komponent zajistit uživateli nepřetržitý přístup do dané aplikace. Případný výpadek, ať již plánovaný nebo neplánovaný, by tedy neměl uživatele žádným způsobem ovlivnit. Je zřejmé, že systémy vysoké dostupnosti se snaží eliminovat zejména výpadky neplánované a umožňují údržbu jednotlivých komponent bez přerušení běhu aplikace.

Plánovaný výpadek může být zapříčiněn údržbou systému, jako je aktualizace software, změna konfigurace nebo jiným dopředu naplánovaným zásahem, který vyžaduje restart systému. Neplánovaný výpadek bývá důsledek poruchy hardwarových nebo softwarových komponent. Mezi nejčastější příčiny patří porucha procesoru, operační paměti či síťové karty, přetížení sytému nebo chyba v síťovém připojení.

Dostupnost se obvykle vyjadřuje jako poměr celkové doby běhu aplikace bez výpadku ku pevnému časovému úseku. Jako pevný časový úsek je uvažován měsíc nebo rok. Výsledná hodnota se udává v procentech. Běžné hodnoty dostupnosti systémů s vysokou dostupností jsou:

99,9 % = doba výpadku 48,3 min/měsíc nebo 8,76 hod/rok,

99,99 % = doba výpadku 4,38 min/měsíc nebo 52,56 min/rok,

99,999 % = doba výpadku 0,44 min/měsíc nebo 5,26 min/rok.

Nástroje pro měření dostupnosti musí být nezávislé na systému vysoké dostupnosti, aby nebyli ovlivněny jeho případnými výpadky a měření musí probíhat nepřetržitě.

Řešení vysoké dostupnosti vyžaduje redundanci minimálně však duplicitu hardwarových komponent se shodnou konfigurací a datovým obsahem. Za běhu aplikace se provádí replikace dat a databáze. Na částech systému, které plní stejnou funkci je tímto způsobem udržován shodný datový obsah. Za těchto podmínek je pak možné rozložit zátěž na jednotlivé prvky systému a provádět paralelní obsluhu jednotlivých uživatelů. V případě výpadku jednoho ze serverů je služba, která byla právě zpracovávána převzata jiným strojem. Může to být jeden z aktivních nebo je dotaz přesměrován na některý z uzlů, který je pro tuto situaci připraven v pohotovostním režimu. Převzetí služby, anglicky failover, proběhne automaticky a bez přerušení běhu aplikace. Monitorovací zařízení nahlásí chybu obsluze, která po vyřešení problému uvede havarovaný server znovu do provozu nebo jej nahradí novým [4].

2. Cluster s vysokou dostupností

2.1 Výpočetní cluster

Výpočetní cluster je paralelní seskupení osobních počítačů nebo serverů, spojených sítí LAN, kteří spolu úzce spolupracují. Navenek působí jako jeden stroj. Clustery jsou sestavovány pro zvýšení výpočetního výkonu. Používání clusterů je většinou finančně výhodnější než využití jediného superpočítače. Výpočetní clustery se používají pro náročné vědecké výpočty, jako centrální jednotky rozsáhlých počítačových sítí, pro provoz řídícího systému jaderných elektráren vykreslování grafických animací, vytváření modelů pro předpověď počasí atd. Architektura výpočetních clusterů byla základem návrhu systémů pro hostování webových aplikací a služeb s tím spojených.

První zmínka o možnosti rozložení zátěže mezi více výpočetních jednotek se datuje přibližně na přelom padesátých a šedesátých let minulého století. Vývoj informačních technologií vedl k sestavení počítačové sítě ARCnet vyvinutou společností Datapoint v roce 1977. Tato síť umožňovala paralelní provádění jednotlivých výpočtů. Výsledkem projektu firmy DEC byl v roce 1984 produkt VAXcluster. Jednalo se o systém spolupracujících počítačů. Mezníkem clusterových systémů bylo uvedení operačního sytému PVM (Paralel Virtual Machine) v roce 1989. Tento software byl vyvinut ve spolupráci University of Tennessee, Oak Ridge National Laboratory a Emory University v USA, který byl založen na komunikaci již pomocí protokolu TCP/IP. Umožňoval vytvoření virtuálního superpočítače pro provádění náročných výpočtů. V roce 1995 byl ve společnosti NASA dokončen projekt Beowulf, jehož výsledkem bylo sestavení paralelního počítače – clusteru. Cluster se skládal z šestnácti počítačů využívajících operační systém Linux s procesory 486DX4 propojených upraveným vícekanálovým 10Mb ethernetem. Tím začal masivní vývoj a používání clusterových systému. Dnešní nejvýkonnější cluster BlueGene/L představuje systém, který obsahuje 65,536 uzlů s dvoujádrovými procesory spojených sítí s přenosovou rychlostí 1,024 Gb/s, což představuje výkon 360 teraFLOPS (floating-point operations per second – počet operací v plovoucí řádové čárce za sekundu) [6].

2.2 Využití clusterů pro webové aplikace

Cluster s vysokou dostupností, anglicky High-availability cluster, je používán pro zvýšení dostupnosti služeb a aplikací. Zajišťuje také stabilitu celého systému.

Teoreticky lze dosáhnout 100% dostupnosti, ale v praxi tato hodnota dosahuje nejvýše 99,999 % (viz 1).

Výhodou clusteringu je v první řadě cena. Spojením běžných serverových stanic je získán minimálně stejný výkon jako použití jednoho nákladného superpočítače. Další výhodou je snadná rozšiřitelnost o další stroje. Přidání nového uzlu vyžaduje pouze stejnou softwarovou konfiguraci jako mají ostatní počítače v clusteru. Po fyzickém zapojení je nutné stroj implementovat do systému a pak se již začíná podílet na poskytování služeb, které jsou na clusteru provozovány.

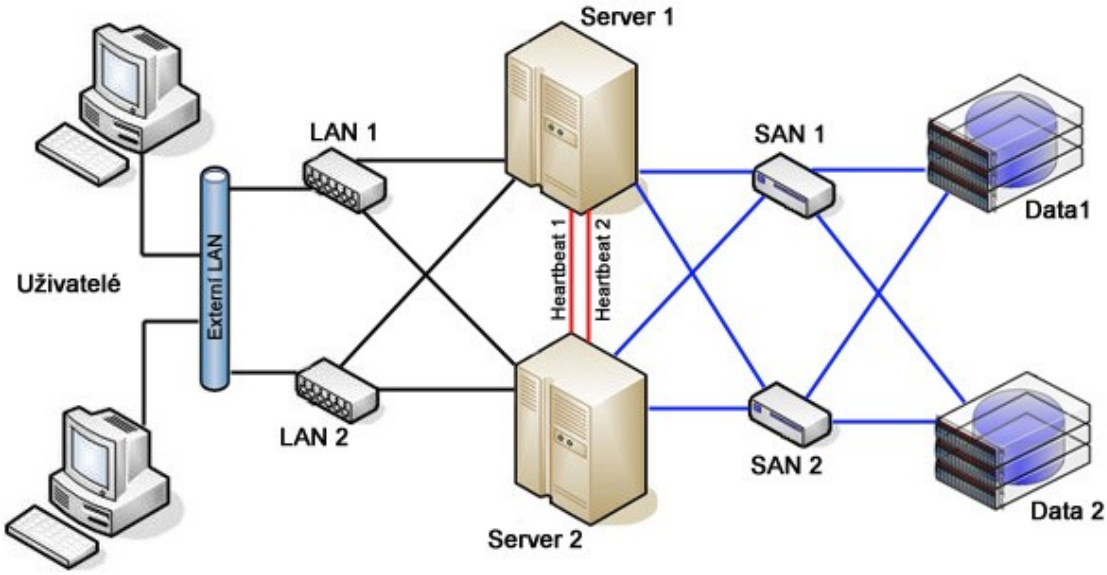
Systém používá nástroje pro monitorování stavu jednotlivých uzlů. Sledování stavu je realizováno formou zasílání zpráv protokolů aplikační nebo transportní vrstvy modelu OSI. V závislosti na počtu uzlů a vnitřní struktuře clusteru jsou dotazy na stav rozesílány dvěma způsoby. První způsob pracuje na principu takzvaných pulsů, kdy je zaslán signál z jedno uzlu na všechny ostatní a tím se ověřuje zda jsou stále aktivní. Signál putuje celým systémem z jednoho uzlu na druhý. Když selže monitorování pulsu u nějakého uzlu, služby prostředků clusteru provedou příslušnou akci. Pro tento princip se vžil název *heartbeat* stejně jako pro softwarový prostředek, který tento způsob monitorování zajišťuje. Druhý způsob vyžaduje zařazení řídícího uzlu do clusteru, na kterém běží monitorovací nástroj. Sledování stavu formou zasílání zpráv je realizováno s každým uzlem zvlášť. Tento princip monitorování využívá například nástroj HAProxy (viz 4.1.2). V situaci, kdy je detekován výpadek některého z uzlů jsou služby, které vykonával převedeny na jiný aktivní stroj. Výhodou je, že proces převzetí služby, anglicky failover, nevyžaduje žádný zásah obsluhy.

Pro provoz webových aplikací se implementují clustery s různou vnitřní strukturou. Jednotlivé uzly mohou být využity pro provoz jediné nebo i více služeb zároveň. Cluster je možné sestavit pro potřeby jedné nebo více aplikací, které na něm budou provozovány v závislosti na objemu dat, množství uživatelských přístupů a v neposlední řadě na finančních možnostech provozovatele [8], [23].

2.3 Struktura clusterů

Cluster je sestaven z jednotlivých serverů spojených sítí LAN. Jednotlivé stroje nazýváme uzly clusteru. Vysoká dostupnost je zajištěna přesouváním dotazů havarovaných uzlů na aktivní a rozkladem zátěže mezi jednotlivé stroje. V případě výpadku některého uzlu je v pohotovostním režimu připraven jiný okamžitě převzít jeho

služby. V tomto případě hovoříme o redundanci hardware. Neobsahuje-li cluster žádné redundantní stroje jsou služby havarovaného serveru rovnoměrně rozloženy mezi ostatní uzly clusteru. Redundance není podmínkou pro zajištění vysoké dostupnosti. Je ale zřejmé, že minimální konfigurace clusteru s vysokou dostupností musí obsahovat alespoň dva stroje (viz. Obr 2-1, [20]).



Obr. 2-1 – Cluster s dvěma uzly

V praxi však tyto systémy obsahují větší množství serverů. Cluster lze vnitřně rozdělit na několik částí. Na vstupních uzlech je provozována služba DNS, kde je realizován překlad doménových jmen na IP adresy jednotlivých strojů, rozklad zátěže na jednotlivé aplikační servery (viz. kap. 3.1) a může zde běžet služba pro částečné cacheování obsahu stránek. Na dalších strojích jsou provozovány služby jako web server, databáze, mail server, FTP server případně Samba server atd. Pro maximální využití výkonu jednotlivých strojů je možné provozovat více služeb na jednom serveru. V závislosti na náročnosti a typu aplikace může být oddělen úložný prostor pro data od aplikačních serverů. Přenos dat mezi servery a úložným prostorem je pak realizován pomocí vysokorychlostní sítě (viz 3.3.2) nebo sítě internet (viz 3.3.3). Uspořádání uzlů v clusteru lze rozdělit do následujících modelů:

Active/Active – tento model neobsahuje žádný nadbytečný uzel, zátěž je rovnoměrně rozložena na všechny servery. V případě výpadku jsou služby

havarovaného uzlu opět rovnoměrně rozděleny mezi zbylé stroje.

Active/Passive – je varianta rozložení, kde ke každému aktivnímu uzlu clusteru je právě jeden redundantní v pohotovostním režimu, který je připraven převzít služby při výpadku. Jedná se o plnou redundanci.

N+1 – poskytuje pouze jeden uzel v pohotovostním režimu pro N aktivních. Redundantní uzel musí být univerzálně nakonfigurován, aby mohl převzít službu jakéhokoliv uzlu z daného clusteru.

N+M – Cluster typu, kde N uzlů je aktivních a M v režimu stand-by. Vhodný je pro hostování více druhů aplikací nebo služeb. M redundantních uzlů, tedy více než jeden, zajistí vyšší dostupnost aplikací při několikanásobných výpadcích.

N-to-1 – obsahuje pouze jeden redundantní uzel, který je připraven převzít služby havarovaného stroje pouze dočasně. Ve chvíli, kdy je původní server uveden znovu do provozu přebírá své služby zpět a redundantní uzel se vrací do pohotovostního režimu.

N-to-N – Jedná se o kombinaci *Active/Active* a *N+M* clusterů. Tento typ neobsahuje žádné redundantní uzly. Selže-li nějaký z uzlů jsou akce, které právě obsluhoval rozděleny mezi zbylé stroje v clusteru.

Každý uzel v clusteru je jednoznačně určen privátní IP adresou. Předávání nebo přiřazování jakékoliv dotazu je směrováno právě pomocí IP adres aplikačních serverů. Směrování probíhá na vstupních zařízeních clusteru, kterými zpravidla bývají DNS server nebo router. Na vstupních uzlech se provádí rozklad zátěže (anglicky load balancing) a monitorování stavu jednotlivých uzlů [5].

2.4 Předávání služby – failover

Proces předání služby, anglicky failover, je vyvolán na základě chybového hlášení monitorovacího zařízení. Monitorovací zařízení detekuje výpadek některého ze serverů a automaticky převede službu, kterou poskytoval na aktivní server. IP adresa havarovaného serveru je automaticky vyřazena z cyklického seznamu algoritmu pro rozdělování zátěže. Nově přicházející dotazy jsou vykonávány na aktivních uzlech clusteru. Havarovaný server je opět zařazen do cyklického seznamu automaticky po obnovení jeho chodu.

3. Metody a nástroje pro realizaci vysoké dostupnosti

3.1 Load balancing

Load balancing nebo-li rozklad zátěže, je metoda, která umožňuje provoz jedné služby na více serverech najednou. S rostoucím počtem podílejících se serverů ovšem roste pravděpodobnost výpadku. Load balancing sám o sobě nezaručuje vysokou dostupnost, ale pouze rozloží zátěž a tím urychlí běh celé aplikace. Pro zajištění vysoké dostupnosti je nutné použít další nástroje, které zajistí převzetí služeb havarovaného serveru. Při splnění této podmínky je rozklad zátěže mezi více strojů efektivní. Realizace load balancing clusteru předpokládá na vstupu stroj, který bude řídit rozdělování zátěže na jednotlivé servery (viz. 4.1.2). Pro rozdělování dotazů na jednotlivé servery se používají algoritmy popsané v následujících kapitolách 3.1.1 - 3.1.5. Mezi nejpoužívanější patří Round robin, Weighted round robin, Least connection, Weighted least connection [10], [11], [12].

3.1.1 Round robin

Round robin je metoda, která jednoduchým způsobem řeší rozklad zátěže. Přicházející dotazy od uživatelů jsou tímto algoritmem přeposílány ke zpracování mezi jednotlivé stroje clusteru. Například webová aplikace má jedno doménové jméno a je provozována na více serverech najednou. Každý ze serverů má svou IP adresu. Round robin pracuje s cyklickým seznamem IP adres, anglicky hash table. Jednotlivé dotazy postupně posílá konkrétnímu stroji ke zpracování. Příchozí dotaz je poslán serveru, který je první v tabulce IP adres a po odeslání dotazu je tato IP adresa je zařazena na konec tabulky. Tímto způsobem se cyklicky střídají všechny severy v práci.

Roud robin je využívám pro load balancing v rozsáhlých sítích, pro přenos dat v reálném čase nebo pro rozložení zátěže mezi servery, které mají vůči sobě různou geografickou polohu. Nevýhodou tohoto systému může být nerovnoměrné rozložení zátěže v případě rozdílné náročnosti jednotlivých dotazů nebo různého výkonu jednotlivých uzlů. Každý dotaz zabírá nějaký procesorový čas. Může nastat situace, kdy jeden ze serverů řeší náročnější úlohu, tedy potřebuje více času, a ve frontě má další dotazy. Další stroj čeká na přiřazení dotazu, protože všechny své úlohy již vyřešil a je tak nevyužitý. Proto je vhodné tento způsob rozkladu zátěže použít v systémech s větším počtem strojů, kde má každý ze serveru dostatek času k odeslání odpovědi než

je znovu dotazován. V tomto případě je vhodné do clusteru zařadit stroje se stejným výpočetním výkonem [10], [11], [12].

3.1.2 Weighted round robin

Algoritmus weighted round robin byl navržen pro efektivnější rozkládání zátěže mezi uzly clusteru s různým výpočetním výkonem. Každému serveru je přiřazena váha, celočíselná hodnota označována W, která charakterizuje jeho výpočetní kapacitu. Stroji s větší váhou, tak bude přiřazeno více dotazů ke zpracování než serveru s váhou nižší. Uzlům se stejnou hodnotou W, je přidělováno stejné množství dotazů. Například mějme tři servery A, B a C s různou kapacitou $W_A = 4$, $W_B = 3$ a $W_C = 2$. Sekvence přiřazování dotazů na jednotlivé servery bude vypadat takto: AABABCABC a bude se periodicky opakovat. Roud robin (viz 3.1.1) lze označit jako speciální případ weighted roud robin, kde všechny uzly clusteru mají stejnou váhu.

Směrování dotazů pomocí algoritmu weighted round robin je výkonnější než roudn robin, protože řeší problém různého výpočetního výkonu jednotlivých strojů. Nicméně nevýhodou stále zůstává možná nerovnoměrnost rozložení zátěže v případě rozdílné náročnosti jednotlivých dotazů [10], [11], [12].

3.1.3 Least connection

Směrovací algoritmus least connection distribuuje dotazy mezi servery na základě počtu spojení na jednotlivých strojích. Příchozí dotaz bude předán uzlu s nejmenším počtem právě vykonávaných dotazů. Jedná se o dynamický algoritmus, protože je nutné stále monitorovat počet připojení na jednotlivých serverech. Tento algoritmus zajistí rovnoměrný rozklad zátěže na výkonnostně podobných serverech.

Nevýhodou je, že každý síťový protokol má definovanou dobu, po kterou čeká na další data. Až po uplynutí tohoto časového intervalu se spojení ukončí. V tomto algoritmu je i neaktivní spojení, již zpracovaný dotaz, počítán do celkového počtu vykonávaných dotazů na daném uzlu. Je tedy zřejmé, že v případě většího rozdílu ve výkonu mezi jednotlivými stroji, bude silnější server méně vytížený než slabší [10],[11],[12].

3.1.4 Weighted least connection

Weighted least connection je nadstavbou algoritmu least connection, kde je každému uzlu clusteru přiřazena váha podle jeho výpočetního výkonu. Váha vyjadřuje

kolik procent z celkového počtu spojení bude vyřizovat. Výchozí hodnota je jedna.

Algoritmus pracuje podle následujících vztahů [10]:

Předpokládejme *n* serverů, kde každý server *i* má váhu W_i $(i = 1,...,n)$ a počet spojení C_i $(i = 1,...,n)$. *ALL_CONNECTIONS* je součet všech spojení C_i $(i = 1,...,n)$. Následující spojení bude přesměrováno na server *j* když,

$$\frac{C_j}{ALL\_CONNECTIONS \cdot W_j} = \min\left[\frac{C_i}{ALL\_CONNECTIONS \cdot W_i}\right]$$

ALL_CONNECTIONS je konstanta a je možné ji vynechat, pak výsledný vztah bude

$$\frac{C_j}{W_j} = \min\left[\frac{C_i}{W_i}\right]$$

Tento algoritmus je nejčastěji používán v praxi, protože nejlépe řeší rovnoměrný rozklad zátěže mezi jednotlivé uzly clusteru. Hodnota *C* může vyjadřovat nejen počet spojení, ale také například zatížení systému. Nevýhodou stále zůstává započítání neaktivních spojení do celkového počtu spojení pro daný server [10], [11], [12].

3.1.5 Další metody

Locality based least connection je algoritmus pro rozklad zátěže mezi servery s různou geografickou polohou, které jsou v síti jednoznačně určeny IP adresou. Nejčastěji se používá pro rozklad zátěže cacheovacích clusterů. Datové pakety jsou posílány na IP adresu aktivního serveru. Je-li daný server přetížený, již zpracovává váhou určený počet spojení, potom je spojení přesměrováno na další server, který ještě může nové spojení přijmout a zpracovat. Jádrem tohoto postupu je směrovací algoritmus Weighted least connection (viz 3.1.4).

Locality based least connection s replikací je také algoritmus pro IP load balancing serverů s různou geografickou polohou. Nejčastěji je opět nasazován pro rozklad zátěže cacheovacích clusterů. Rozdíl od předchozího algoritmu je v tom, že load balancer si uchovává cestu k serverům, které mohou obsloužit daný dotaz a vytváří si tak sadu serverů. Dotaz je pak směrován na server s nejmenším počtem spojení. V případě, že všechny servery z této sady jsou plně vytíženy je vybrán zbylý uzel z clusteru s nejmenším počtem spojení a je přidán do této skupiny. Není-li sada serverů připravena za pevně definovaný čas obsloužit dotaz, pak je nejvíce zatížený stroj odebrán ze skupiny, aby nedošlo k nadměrné replikaci daného spojení.

Destination hashing je směrovací algoritmus, který staticky přiřazuje spojení jednotlivým serverům pomocí cyklického seznamu cílových IP adres na základě geografického umístění.

Source hashing je směrovací algoritmus, který staticky přiřazuje spojení jednotlivým serverům pomocí cyklického seznamu zdrojových IP adres.

Shortest expected delay je směrovací algoritmus, který přiřazuje spojení na základě nejkratší předpokládané odezvy. Předpokládaná prodleva je doba vykonání jednoho dotazu na daný server. Lze vyjádřit vztahem $(C_i + 1) / U_i$, kde C_i je počet spojení a U_i je stabilní rychlost obsluhy dotazů daného serveru.

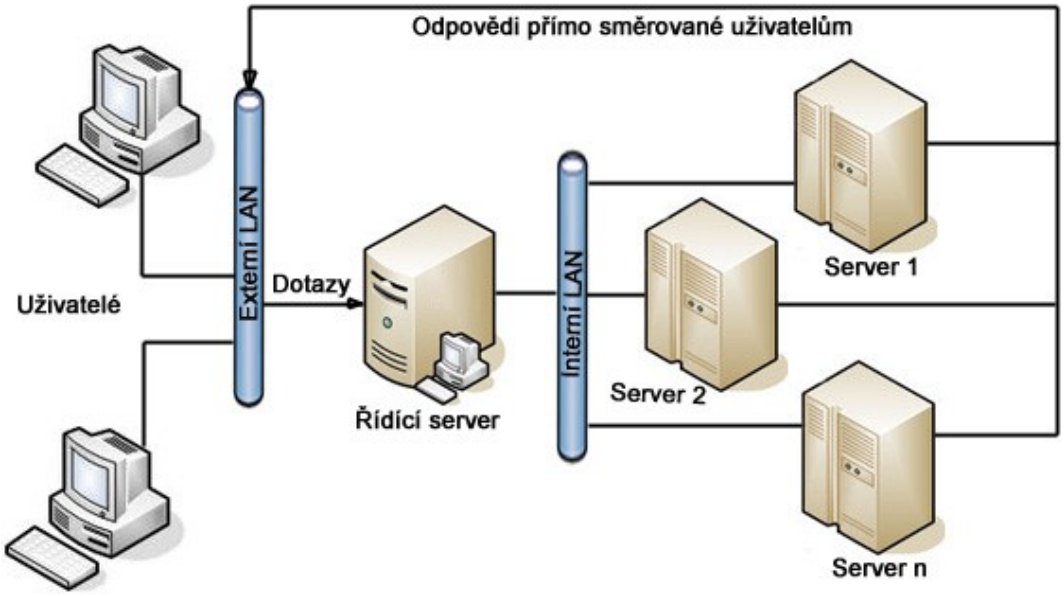
Never queue je směrovací algoritmus, který rozesílá dotazy dvěma způsoby. První možnost počítá s přítomností redundantního uzlu v pohotovostním režimu. Dotaz je odeslán v případě, že není k dispozici žádný výkonnější z aktivních strojů. Druhý způsob směrování dotazu je použit v případě, že v systému není přítomen žádný nadbytečný uzel a dotaz je poslán na server s nejmenší předpokládanou odezvou [10], [11], [12].

3.2 Load balancing − přeposílání požadavků

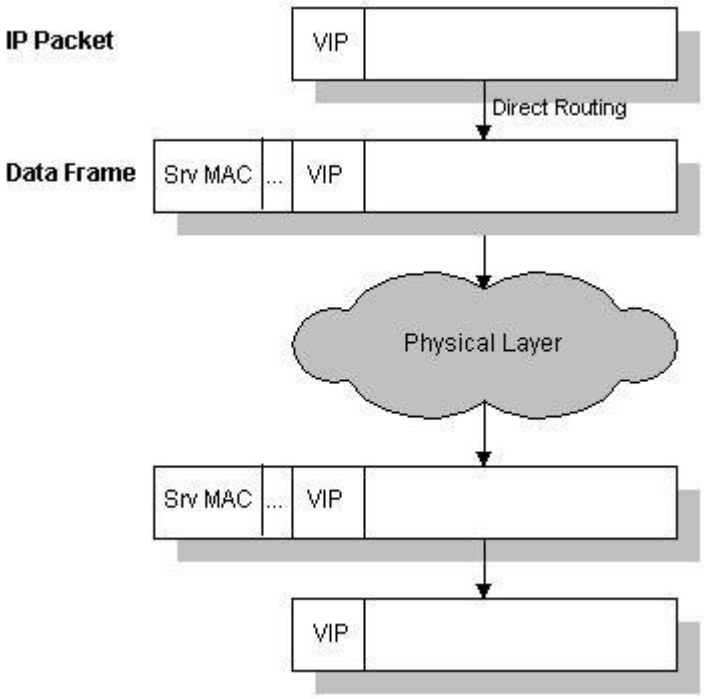
Přeposílání požadavků řeší jakým způsobem budou datagramy doručeny od řídícího stroje k pracovním uzlům clusteru a od nich k uživateli. Nejčastěji se používají metody popsané v následujících kapitolách 3.2.1 – 3.2.3.

3.2.1 Direct routing

Direct routing nebo-li přímé směrování je metoda, která přeposílá datagramy na cílový server pomocí virtuální IP adresy a MAC adresy cílového serveru. Řídící stroj a pracovní servery proto musejí být zapojeny ve stejné síti (viz. Obr. 3-1,[20]). Sdílejí jednu virtuální IP adresu, která slouží pro přijetí datagramu s dotazem. Load balancer a jednotlivé servery mají nadefinované takzvané non-arp alias rozhraní, které je definováno virtuální IP adresou. Dotaz je směrován na vybraný aplikační server pomocí MAC adresy na úrovni fyzické vrstvy modelu OSI (viz Obr 3-2, [12]). Cílový server přijme datagram z rozhraní a odpověď je pak odeslána přímo uživateli. Tento způsob směrování má minimální procesní náročnost, proto se používá v systémech s vysokou frekvencí dotazů [12].



Obr. 3-1 – Direct routing

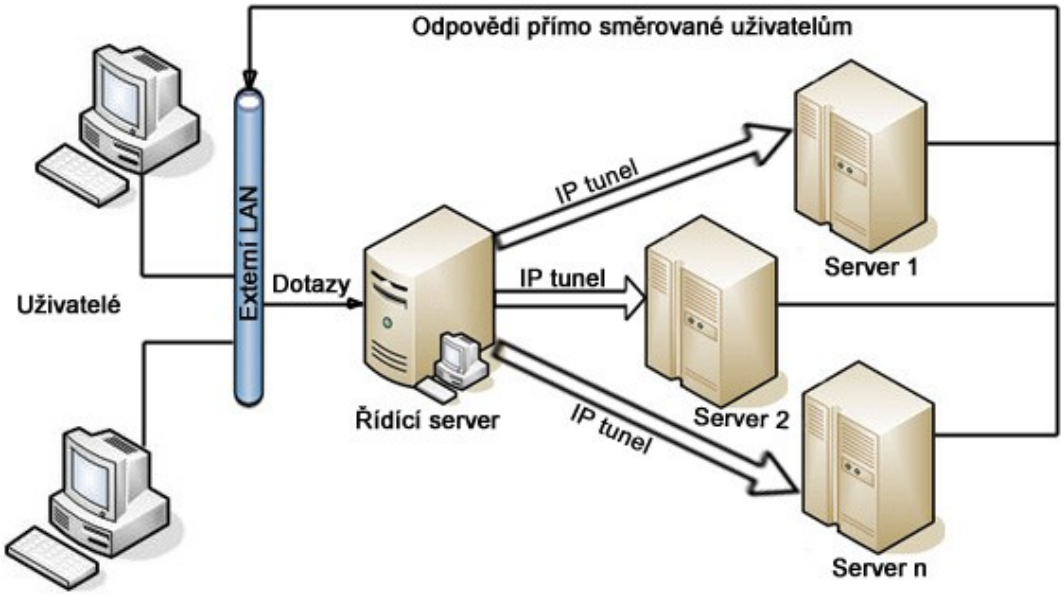


Obr. 3-2 – Direct routing – předání dotazu

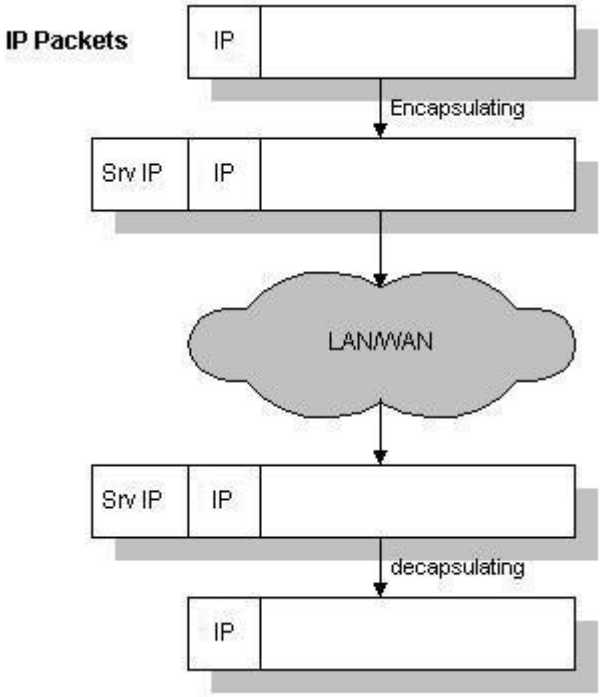
3.2.2 Tunneling

Tunneling pracuje na stejném principu jako direct routing (viz 3.2.1) s tím rozdílem, že datagramy jsou na řídícím serveru znovu zapouzdřeny protokolem IP a je tak vytvořen takzvaný IPIP tunel (viz. Obr. 3.3, [20]). Datagramy jsou směrovány přímo na IP adresu pracovního stroje na úrovni síťové vrstvy modelu OSI (viz Obr. 3-4, [12]).

Pracovní servery nemusí být na stejné síti a mohou mít různou geografickou polohu, čímž se zvýší odolnost celého systému proti výpadku [10], [12].



Obr. 3-3 – Tunneling

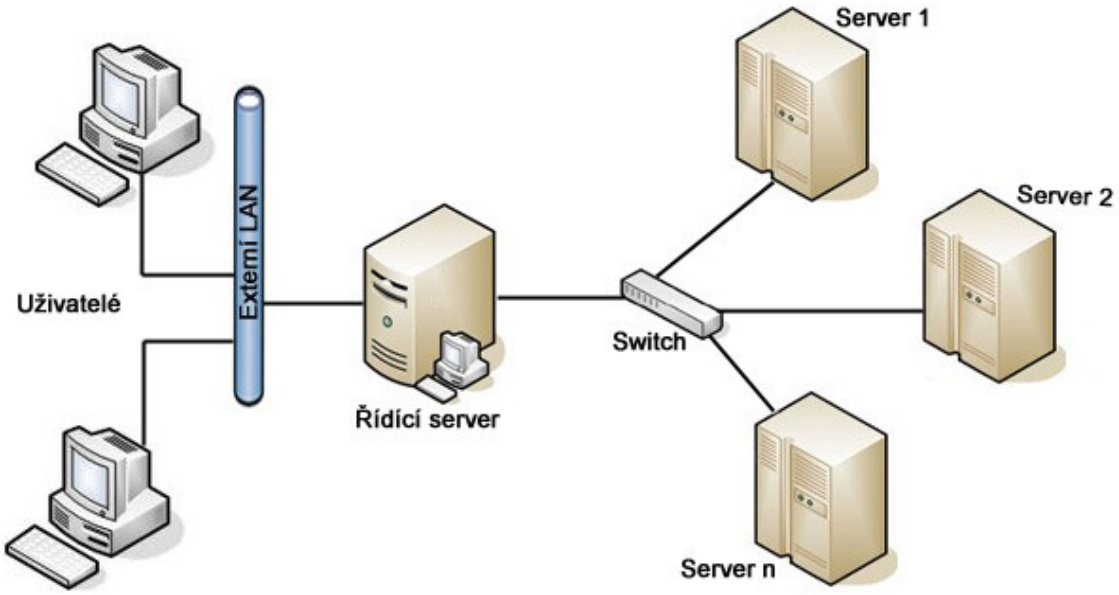


Obr. 3-4 – Tuneling – předání dotazu

3.2.3 Network address translation (NAT)

Network address translation nebo-li překlad adres je další možný způsob přeposílání požadavků. Řídící server přeloží IP adresu v datagramu na adresu jednoho

z pracovních strojů, který byl vybrán algoritmem pro rozklad zátěže. Servery v clusteru mají IP adresy, kterými jsou jednoznačně určeny ve vnitřní síti. Datagram s odpovědí je pracovním strojem předán zpět řídícímu serveru, kde je opět přeložena cílová adresa na IP adresu uživatele. Řídícím serverem je odpověď odeslána uživateli. Celá služba je tak dostupná pouze na jediné IP adrese. Tato metoda je procesně nejnáročnější ze všech uvedených možností [10], [12]. Konkrétní zapojení (viz. Obr 3-5, [20]).



Obr. 3-5 – Network address translation (NAT)

3.3 Správa dat v systémech s vysokou dostupností

V systémech s vysokou dostupností je nutné zajistit stále aktuální a shodný datový obsah na uzlech, kde je provozována stejná služba. Jedná se zejména o databázi, data aplikací a uživatelů. Replikace dat probíhá paralelně s vykonáváním dotazů jednotlivých služeb. Pro zaručení konzistence a integrity dat se používají systémy a postupy popsané v následujících kapitolách.

3.3.1 Network file system

Network file system nebo-li síťový souborový systém je obecně služba, která je provozována přes všechny souborové servery. Zajišťuje sdílení a distribuci dat v počítačové síti. Umožňuje základní operace se soubory – vytváření, mazání, čtení a zapisování do souboru. Pro nasazení v clusterech s vysokou dostupností musí systém také zajišťovat replikaci a aktualizaci dat na všechny uzly v reálném čase. V závislosti

na konfiguraci clusteru, objemu přenášených dat a potřebách aplikace mohou být data uložena buď přímo na aplikačních serverech, odděleně na vyhrazených souborových serverech nebo na oddělených úložných zařízeních. Výhodnější je data skladovat na serverech odděleně od pracovních uzlů clusteru. Zátěž na celý cluster je pak rozložena rovnoměrněji a vykonávání jednotlivých dotazů je rychlejší. Datová úložiště jsou k pracovním serverům připojena pomocí vysokorychlostní sítě (viz. 3.3.2) nebo sítě internet (viz 3.3.3). Základní hardwarové části, které souborový server spravuje jsou disková pole, kde jsou data uložena (viz 3.3.4). Souborový sytém, spravující jednotlivá úložná zařízení a datové servery, které jsou spojené sítí označujeme jako distribuovaný souborový systém.

V systémech s vysokou dostupností, kde se pro realizaci rozkladu zátěže využívá redundance nebo duplicita hardwarových i softwarových komponent je nejdůležitější funkcí použitého síťového souborového systému efektivní replikace a aktualizace dat.

V *distribuovaných souborových systémech* probíhá aktivní replikace dat v reálném čase. Data jsou s každou změnou distribuována sítí ve formě datových bloků na jednotlivá úložná zařízení nebo souborové servery tak, aby byl jejich obsah stále aktuální a shodný. Takto postavený systém skladování dat umožňuje vysokou dostupnost dat pro aplikační servery i v případě výpadků některého ze souborových serverů.

Replikaci databáze zajišťuje systém řízení báze dat nebo-li databázový systém jako je například MySQL (viz 4.2.2). Soustava databázových serverů obsahuje řídící stroj nebo-li *master* a jeden nebo více podřízených počítačů, které označujeme jako *slave*. *Master* udržuje originál databáze a kopie distribuuje sítí na podřízené stroje. Jen server v roli *master* má umožněno čtení i zápis do databáze. Každý *slave* po přijetí aktualizace odešle řídícímu počítači zprávu o průběhu přenosu dat. V případě chyby jsou data odeslána znovu. Servery v roli *slave* mají umožněno pouze číst z databáze. Tímto způsobem je zaručena konsistence databáze. Nastane-li havárie řídícího serveru může některý z podřízených strojů převzít jeho služby a funkci *mastera* nebo je v systému zařazeno více řídících počítačů, což je označováno jako multi-master systém (viz 4.2.3). Tak je zajištěna vysoká dostupnost celého databázového clusteru [13], [14], [15].

3.3.2 Storage area network (SAN)

SAN je vysokorychlostní síť nezávislá na lokální síti, speciálně zaměřená na propojení úložných zařízení jako jsou disková pole nebo páskové jednotky. SAN je oddělena od lokální sítě a funguje jako sekundární pouze za účelem správy a distribuce dat. Zajišťuje snížení zátěže na hlavní síť, protože veškeré objemné přesuny dat se odehrávají mezi systémy zapojenými v SAN. Zátěž je rozložena mezi více serverů s různou kapacitou úložného prostoru. SAN je optimalizovaná na sběr, ukládání a výběr bloků dat.

Většina sítí, které jsou využívány výhradně k propojování serverů se systémy pro ukládání a správu dat používá pro vzájemnou komunikaci rychlého rozhraní SCSI. Vlastní přenos dat je realizován technologií Fibre Channel, která umožní přenosové rychlosti 1,2,4 a 8 Gbit/s. Fibre Channel je technologie, která využívá k přenosu dat optických vláken. Používají se i jiné síťové protokoly pro zapouzdření a přenos dat z diskového rozhraní jako například :

iSCSI protokol, který mapuje SCSI pomocí TCP/IP protokolu.

HyperSCSI síťový protokol mapující SCSI skrz ethernet.

ATA over Ethernet protokol mapující ATA přes ethernet.

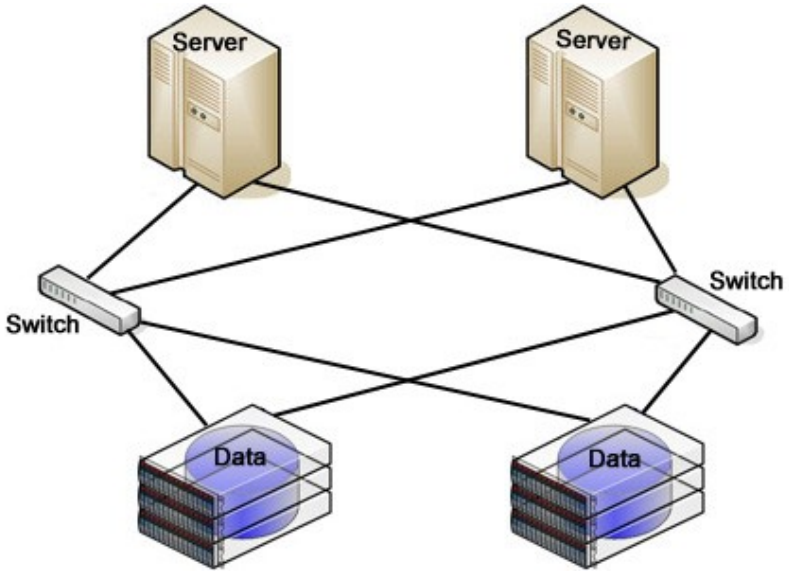
Tyto technologie však dosahují maximální přenosové rychlosti 10 Mbit/s jsou tedy výrazně pomalejší. Kombinace rychlých SCSI disků připojených do sítě pomocí Fibre Channel zaručí v současné době nejrychlejší přenos dat na pracovní servery, ale je zároveň finančně nejnáročnější.

SAN je snadno rozšiřitelná o další diskové jednotky, které mohou být hned po naistalování využívány všemi servery. Dále umožňuje spouštění operačního sytému jednotlivých strojů přímo z této sítě, což podstatně usnadní případnou výměnu havarovaného serveru. SAN zajišťuje replikaci dat, která jsou využívána aplikačními servery, na stroje do oddělené záložní sítě. Sekundární úložné zařízení je propojeno s hlavním systémem shodným typem vysokorychlostní sítě. Tím může být již zmiňovaná technologie Fibre channel. Řešení havárie celého sytému je pak efektivnější a je stále zaručena vysoká dostupnost dat aplikací. Novější verze SAN systémů podporují služby jako je klonování nebo snapshotting jednotlivých diskových jednotek v reálném čase pro účely zálohování, obnovy dat po havárií nebo duplikaci systému.

Ovladače diskových jednotek v sítích SAN jsou navrženy pro rychlý přístup a

práci s bloky dat. V těchto sítích je možné kombinovat různé typy disků. Typ SATA zapojený do sítě pomocí Fibre Channel, jehož nevýhody jsou menší výkon a větší pravděpodobnost selhání. Výhodou je však větší kapacita a nižší cena v porovnání s disky typu SCSI. Tohoto se využívá k paralelnímu přístupu k datům. Data ke kterým se často přistupuje jsou uložena na rychlejších SCSI discích a na mediích typu SATA jsou skladována data, která jsou méně často využívána. Jednotlivá media jsou v této síti jednoznačně určena adresou, která je označována jako Logical unit number (LUN).

Vnitřní struktura SAN sítí pro práci s daty se nazývá *switched fabric* (viz Obr. 3-6, [20]). Takovéto uspořádání je navrženo pro rychlou a bezpečnou práci s daty. V této struktuře je každý stroj přímo propojen s ostatními pomocí přepínačů (switch) [13], [14], [15].



Obr. 3-6 – Switched fabric

3.3.3 Content delivery network (CND)

CDN (Content Delivery Network) je síť pro doručování obsahu. Jde o soubor serverů a síťových prvků, které spolupracují přes síť internet. Zajišťuje efektivní doručování velkých datových celků, jako například video, fotografie nebo hudební soubory koncovým uživatelům. V tomto systému jsou jednotlivé souborové servery nebo celé systémy pro ukládání dat rozmístěny na geograficky různých místech (viz Obr. 3-7, [20]). Uživatel je při konkrétním dotazu na specifický obsah přesměrován na geograficky nejbližší souborový server a díky tomu jsou mu data rychleji dostupná.

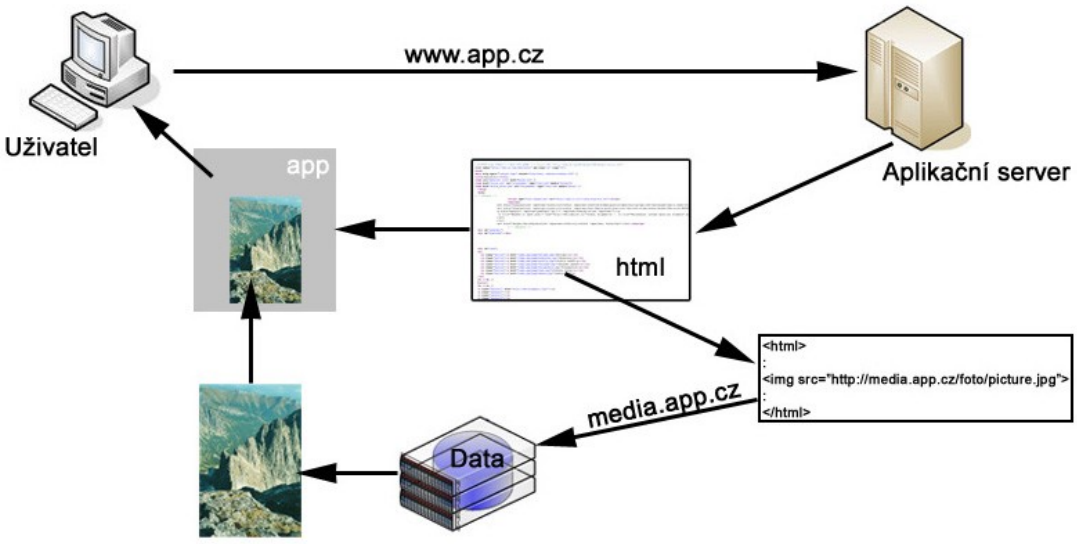
Zátěž na aplikační servery je nižší a zatížení je rovnoměrněji rozděleno na všechny aktivní prvky celého systému. Celý CDN systém musí kromě doručování obsahu webových aplikací také zajistit efektivní replikaci dat na jednotlivé souborové servery tak, aby byla data stále aktuální a shodná (viz 4.3).

Obr. 3-7 – Content delivery network (CND)

Síť internet pracuje na principu spolupráce koncových zařízení, tedy pracovních serverů a počítačů koncových uživatelů, která si mezi sebou posílají datové pakety. CDN rozšiřuje tuto vlastnost o různé druhy aplikací využívajících metody pro optimalizaci doručování větších datových bloků koncovému uživateli. Mezi tyto postupy patří cacheování statického obsahu webových stránek (viz 3.3.5), rozklad zátěže na jednotlivé servery (viz 3.1), směrování dotazů (viz 3.2) a služby pro doručování obsahu. Pomocí těchto prostředků se celý systém snaží zajistit uživateli co nejrychlejší přísun dat. To znamená, že uživatel je směrován na síťově nejbližší server, který je schopen nejefektivněji zpracovat jeho dotazy a doručit datový obsah webové stránky. Je tedy zřejmé, že klient je paralelně obsluhován více stroji a tím je efektivně rozkládána zátěž na celý systém, kde je provozována webová aplikace.

Zpracování dotazu a zobrazení obsahu webové stránky probíhá následujícím

způsobem. (viz Obr. 3-8, [20]) Uživatel s konkrétní IP adresou pošle prostřednictvím webového klienta požadavek na zobrazení webové stránky nebo její části. DNS server přesměruje požadavek na vybraný aplikační server, kde je daná aplikace hostována. Aplikační server zpracuje dotaz, vygeneruje HTML kód s odkazy na specifický obsah, který je umístěný na souborových serverech v síti CDN. DNS server podle IP adresy uživatele vybere jemu síťově nejbližší datový server, ze kterého bude specifický obsah doručen [13], [14], [15].



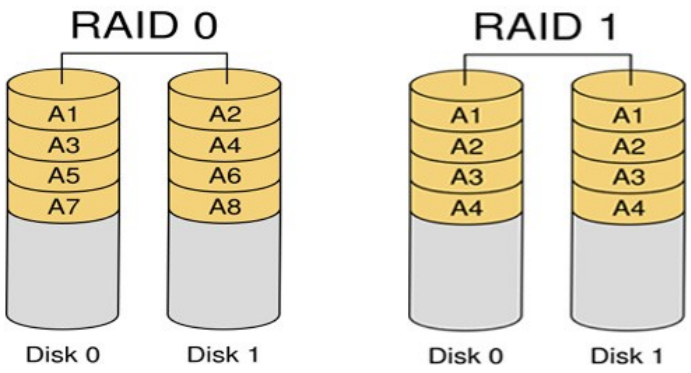
Obr. 3-8 – Doručení obsahu pomocí CDN

3.3.4 RAID

RAID (Redundant Array of Independent Disks) nebo-li vícenásobné diskové pole nezávislých disků je typ diskových řadičů, které zabezpečují koordinovanou práci dvou nebo více fyzických diskových jednotek. Data jsou na discích uložena podle různých logických uspořádání tak, aby selhání jednoho disku nezpůsobilo havárií celého pole. Zvyšuje se rychlost přístupu k datům odolnost vůči chybám, konsistence a integrita uložených dat. Diskové pole zařazené v počítačovém systému se prezentuje jako jeden logický celek. RAID lze implementovat na hardwarové a softwarové úrovni. Hardwarový RAID je sestaven z jednotlivých disků připojených ke speciálnímu řadiči, pomocí kterého operační systém detekuje celé pole jako jeden logický celek. Softwarový RAID nastavením operačního systému prezentuje uživateli jednotlivé disky jako jeden celek. Existuje celkem šest základních typů zapojení diskového pole:

RAID 0 (viz Obr. 3-9, [20]) vyžaduje minimálně dvě diskové jednotky a data jsou na disky ukládány dvěma způsoby − zřetězením a prokládáním. Zřetězení spočívá v ukládání dat nejdříve na první disk a po zaplnění se ukládá na další. Nevýhodou je, že po selhání jednoho disku jsou data již neobnovitelná. Prokládání nebo-li striping ukládá data na disky proloženě. To znamená, že soubor je rozdělen na menší části, které jsou střídavě vkládány na jednotlivé disky. V případě havárie jednoho disku jsou data neobnovitelná, ale tento způsob umožňuje rychlejší čtení, které probíhá z více disků najednou.

RAID 1 (viz Obr. 3-9, [20]) předpokládá zařazení dvou a více disků. Data jsou současně zaznamenávána na dva disky najednou. Tento postup je nazýván zrcadlení nebo-li mirroring. V případě výpadku jednoho disku jsou k dispozici data na druhém. Jedná se o nejjednodušší, ale poměrně efektivní ochranu dat.



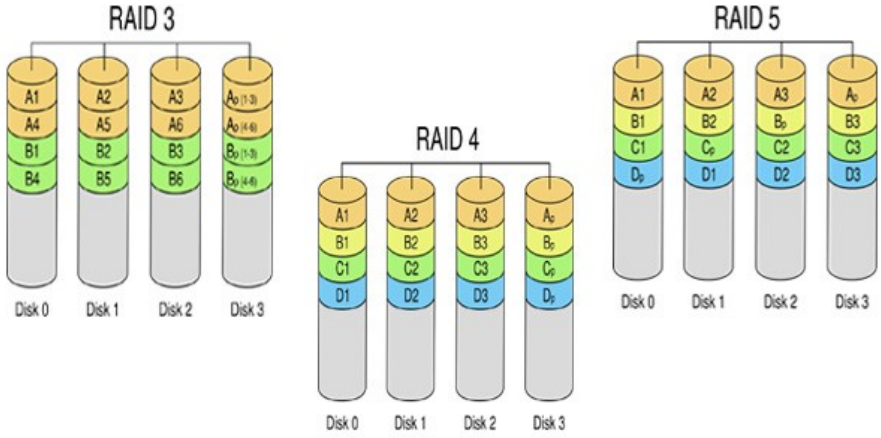
Obr. 3-9 – RAID 0 a RAID 1

RAID 0-1 je kombinací RAID 0 a RAID 1, kde je nutné použít alespoň čtyři diskové jednotky. Data jsou prokládaně (stripováním) ukládána na dva disky a jejich kopie shodným způsobem na další dva disky. Tím získáme dvě logické jednotky s redundantním obsahem. Výhodou tohoto způsobu je, že zátěž rozkládáme mezi více disků. Duplicita obsahu poskytuje jednoduchou obnovu dat po havárií.

RAID 1-0 je opět kombinací RAID 0 a RAID 1. Tato struktura vyžaduje opět zařazení alespoň čtyř diskových jednotek. Nejprve jsou stejná data ukládána na první dva disky A, B a pak na druhé dva C, D, znovu prokládáním. Získáme tak dva logické disky AB a CD na nichž jsou data uložena *stripovaně*. Výhody jsou podobné jako

u varianty RAID 0-1 s tím rozdílem, že tento způsob je odolnější vůči výpadku a obnova dat po chybě některého z disků je rychlejší.

RAID 3 (viz obr. 3-10, [20]) používá N + 1 stejných disků minimálně však 2 + 1. Data jsou ukládána na N disků po bitech a na zbylý disk je uložen exkluzivní součin, takzvaná parita. Při výpadku paritního disku jsou data zachována. V případě havárie libovolného jiného disku je možno z ostatních disků spolu s paritním diskem chybějící data obnovit. Výhodou je, že pro zajištění integrity dat je třeba jen jednoho redundantního disku. Nevýhodou je že, na paritní disk jsou data vkládána při každém zápisu na ostatní disky, čímž je nejvíce vytížen.



Obr. 3-10 – RAID 3, RAID 4 a RAID 5

RAID 4 (viz Obr. 3-10, [20]) vyžaduje použití N + 1 diskových jednotek. Data jsou na disky prokládána po blocích, stejně jako parita na paritní disk. Tento systém přináší stejné výhody a nevýhody jako je tomu v případě RAID 3.

RAID 5 (viz Obr. 3-10, [20]) vyžaduje zařazení alespoň tří pevných disků. Tento systém řeší problém s přetíženým paritním diskem tak, že parita je střídavě ukládána na jeden z disků. Tím je umožněna obnova dat při výpadku některého z úložných medií a není nutné použít jeden vyhrazený disk pro paritu. S výhodou lze využít paralelního přístupu k diskům pro vykonávání dotazů.

RAID 6 předpokládá použití minimálně čtyř diskových jednotek. Tento systém je podobný jako RAID 5 s tím rozdílem, že ukládá dvě sady paritní informace. Umožňuje tak výpadek dvou disků najednou s tím, že data jsou stále obnovitelná. Rychlost čtení je srovnatelná s RAID 5, ale zápis je pomalejší vlivem výpočtu dvou sad

paritních informací [18].

3.3.5 Cacheování obsahu webových stránek

Cacheování je metoda, která na základě požadavku na konkrétní data dokáže vzít jejich kopií, na místě s lepší dostupností pro daného uživatele. Nejčastěji cacheovaná data jsou obrázky, textové dokumenty a případně i jiné soubory, které jsou opakovaně zobrazovány v dané aplikaci. Do cache je možné uložit i část nebo celou webovou stránku v podobě HTML kódu. Tento postup zkracuje dobu vykonávání dotazu a snižuje zatížení aplikačních serverů tím, že jsou data načítána z jiného místa v síti internet. Dále redukuje zatížení sítě a počet spojení s aplikačním serverem tak, že nacacheovaná data jsou poskytována i dalším uživatelům. Je tedy zřejmé, že objekty jsou do cache uloženy prvním klientem, který si je vyžádal a každý další dotaz od různých uživatelů je realizován právě odtud. Obecně je cacheování obsahu využíváno pro zvýšení dostupnosti celé aplikace.

Tento princip cacheování obsahu je prováděn tak, že cache server čte příchozí dotazy a porovnává je se svým obsahem. V případě, že může uživateli vyhovět posílá odpověď přímo klientovi, bez přesměrování dotazu na aplikační server. Svůj obsah vytváří ukládáním částí nebo celých odpovědí aplikačního serveru. Aplikace, která je provozována na aplikačním serveru musí být napsána tak, aby bylo cache zařízení jednoznačně dáno jaká data má ukládat a jaká je jejich doba platnosti. Hlavička HTML kódu obsahuje informace o platnosti konkrétního objektu a podle toho cache zařízení rozpozná zda je aktuální a platný. V případně vypršení některého z těchto údajů požádá zařízení aplikační server o aktualizaci, buď s dotazem od uživatele nebo samostatně.

Webové cache je možné realizovat na různých úrovních na síti mezi pracovním serverem a klientem. Jedná se o cache implantované ve webových prohlížečích, proxy cache a reverzní proxy cache.

Cache webového prohlížeče využívá část pevného disku klientského počítače. Do této rezervované části jsou ukládány objekty, které byli uživateli již jednou doručeny a jsou určené ke cacheování. Prohlížeč hlídá jejich platnost a při opakování stejné akce posílá na aplikační server dotaz tak, aby byl zpět doručen jen zbývající obsah stránky. Tento způsob cacheování je vlastností webového prohlížeče a o použití rozhoduje uživatel nikoliv provozovatel webové aplikace.

Proxi cache pracuje na stejném principu jako cache prohlížečů s tím rozdílem,

že obsluhuje mnohem větší množství uživatelů. Je umístěna na proxy serveru, který plní funkci prostředníka mezi uživatelem a aplikačním serverem. Dotazy přicházející od uživatelů může obsloužit sám bez navázání spojení s aplikačním serverem. Proxy cache je typ sdílené cache, která efektivně snižuje dobu vykonání dotazu a zatížení sítě, tím že ukládá populární obsah webových stránek od velkého množství uživatelů.

Reverzní poroxy cache nebo-li gateway cache je realizována pomocí proxy serveru, který je umístěn přímo za stroji na kterých je provozována daná aplikace. Dotazy od uživatelů tedy nejprve přicházejí na proxy server, ten je na základě obsahu a platnosti objektů v cache buď celé, nebo z části přepošle na aplikační server. Takováto struktura se navenek prezentuje jako jeden celek. Zajišťuje aplikaci lepší dostupnost, větší stabilitu a efektivnější zpracovávání dotazů. Příkladem využití reverzní proxy cache je v sítích pro doručování obsahu (viz 3.3.3), kde je uložený obsah distribuován celou sítí internet.

Cache umožňuje zprostředkování dalších služeb jako jsou autentifikace uživatele nebo filtování obsahu webových stránek. Paralelně zařazené cacheovací systémy spolu komunikují speciálně navrženým HTCP protokolem, který umožňuje přeposílání dotazů, správu a vytváření webové cache [21].

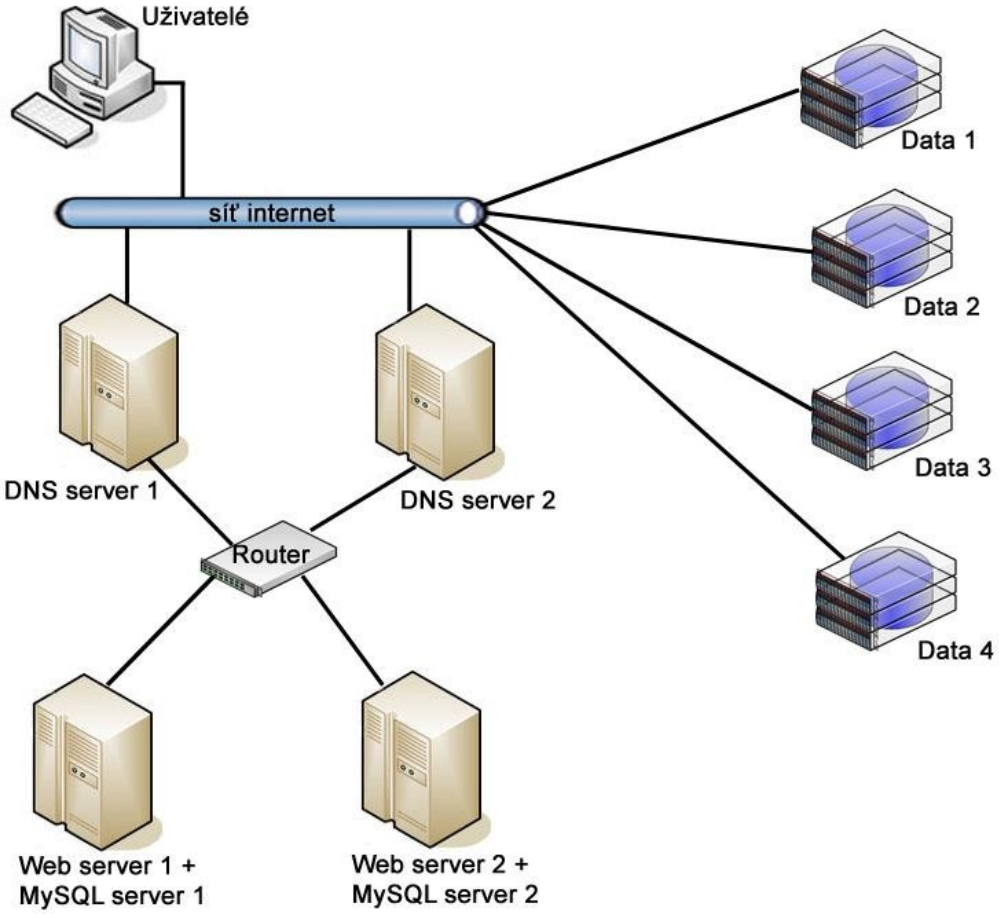
4. Řešení systému s vysokou dostupností (Praktická část)

Úkolem praktické části diplomové práce bylo navrhnout cluster s vysokou dostupností tak, aby splňoval hardwarové, softwarové a kapacitní požadavky webové aplikace. Systém musí být odolný vůči výpadkům jednotlivých uzlů a zajistit dostupnost minimálně 99.5%. Případná havárie některého ze strojů by neměla ovlivnit běh celé aplikace. Webová aplikace počítá s návštěvností 20 000 unikátních přístupů denně, což by mělo představovat minimálně 60 000 zobrazení. Cluster by měl paralelně obsluhovat 2500 návštěvníků a spravovat 2000 uživatelských účtů s datovým prostorem 100MB.

Webová aplikace představuje portál pro hudební kapely. Každý registrovaný uživatel, hudební kapela, bude mít možnost vystavit základní informace, fotografie, část tvorby (hudební soubory, video soubory, texty písní). Portál obsahuje různé statistiky o návštěvnosti a hodnocení jednotlivých profilů a jejich částí. Návštěvníci portálu mohou procházet různým způsobem žebříčky kapel, vyhledávat podle kritérií, pouštět si video a audio soubory, prohlížet profily kapel. O jejich pohybu po portálu se zapisují statistiky. Přehrávání skladeb se eviduje a vytváří podklady pro hodnocení. Z popisu aplikace je zřejmé, že nejvíce náročné operace pro webové a hlavně datové servery je streamování videa a hudby a jejich paralelní distribuce návštěvníkům. Výpočet a údržba stále aktuálních statistik je náročná zejména na databázi.

Vytvořený sytém s vysokou dostupností lze rozdělit na dvě vzájemně oddělené části (viz. Obr 4-1, [20]). Hlavní část tvoří cluster sestavený z primárního a sekundárního doménového serveru, routeru a dvou aplikačních serverů. Na všech strojích byl nasazen operační systém Linux v distribuci CentOS. Úloha DNS (domain name server) serveru (viz. 4.1) spočívá v přidělování požadavků jednotlivým serverům a rozkládání zátěže. Nástroj HAProxy, umístěný na stejném stroji, zajišťuje monitoring stavu webových serverů a rozklad zátěže (viz 4.1.2). Úkolem routeru je směrování jednotlivých dotazů přímo na aplikační servery. Nástroj MMM MySQL replication manager, umístěný na routeru, zajišťuje monitoring databázových serverů (viz 4.2.3). Oba aplikační servery (viz 4.2) se skládají z web serveru (viz 4.2.1) a databázového serveru (viz 4.2.2). Replikačními rutinami je zajištěn jejich shodný obsah. Web server a databáze spolu úzce spolupracují při zpracování jednotlivých dotazů. Na aplikačním serveru je uložen kód celé aplikace a statický obsah webových stránek. Databáze obsahuje veškeré informace o uživatelích, data statistik a vlastní aplikace. Druhou

izolovanou částí jsou datové servery (viz 4.3). Datové servery plní funkci odděleného úložiště pro správu uživatelských dat. Jedná se o hudební soubory a video soubory, což je dynamický obsah webu. Všechny stroje mají stejný datový obsah. Tato struktura umožňuje doručovat, dynamický obsah stránek v závislosti na geografické poloze uživatele. Cluster a datové servery spolu velmi úzce spolupracují při zobrazování celého obsahu webové stránky.



Obr. 4-1 – Struktura systému s vysokou dostupností

4.1 DNS server a monitoring stavu jednotlivých uzlů clusteru

DNS servery (domain name server) tvoří vstupní část clusteru. Zajišťuje obousměrný překlad doménových jmen na IP adresy, určuje cílový stroj a přeposílá jednotlivé dotazy. Na této úrovní byl realizován také rozklad zátěže mezi jednotlivé aplikační servery a jejich sledování. Informace o stavu jednotlivých strojů je klíčová pro

zajištění vysoké dostupnosti celého systému.

4.1.1 DNS server (domain name server) – BIND

Cluster obsahuje dva DNS servery, které spravují jednotlivé domény. Jeden je nastaven jako primární a druhý sekundární. Primární server je ten, na němž data vznikají. Pokud je třeba provést v doméně změnu, musí se editovat data na primárním serveru. Sekundární server je automatickou kopií primárního. Průběžně si aktualizuje data a slouží jako záloha pro případ výpadku primárního serveru.

DNS server obsahuje databázi síťových informací, podle kterých distribuuje dotazy mezi příslušné uzly clusteru nebo externí zařízení. Zajišťuje zasílání odpovědí zpět koncovým uživatelům. Pro realizaci doménového serveru byl použit softwarový nástroj BIND verze 9.2.4-24 s rozšiřujícím balíčkem geoIP. Balíček geoIP rozšiřuje možnosti serveru o schopnost určit geografickou polohu koncového uživatele na základě jeho IP adresy. Této vlastnosti se využívá při výběru datového serveru zařazeného do vnitřní sítě pro doručování obsahu CDN (viz. 3.3.3). Uživatelskému počítači jsou přeposílána požadovaná data ze serveru, který je určen pro obsluhu dotazů z této oblasti nebo-li zóny. BIND obsahuje kromě hlavního konfiguračního souboru ještě takzvané zónové soubory (viz Příloha A).

Hlavní konfigurační soubor obsahuje definice jednotlivých pohledů na jedinou zónu. Byly nadefinovány dva pohledy odkazující na dva různé zónové soubory. Jeden pro uživatele jejichž dotaz přichází z IP adres užívaných v České republice a druhý pro všechny ostatní. Definice pohledů umožňuje doručovat různý datový obsah pod jednou doménou pomocí jediného DNS serveru. Tím bylo dosaženo rozložení zátěže mezi jednotlivé datové servery na základě geografické polohy uživatele a možnost automatické distribuce dvou rozdílných mutací webové aplikace.

Definice pohledu pro uživatele z České republiky:

```
view cz {
        match-clients { !82.208.49.146; country_CZ; };

        zone "miton.eu" IN {
                type master;
                file "pri/miton.eu";
                notify yes;
        };
};
```

V bloku view byl definován pohled *cz*. V parametrech klíčového slova *match-*

clients je seznam všech IP adres, které mohou přistupovat k danému pohledu. Parametr *country_CZ* v sobě skrývá databázi českých IP adres. Adresa 82.208.49.146 je pro tento pohled zakázána (viz níže). V bloku zone byla definována zóna *miton.eu*. Za klíčovým slovem type je hodnota master. Tím bylo nastaveno, že pro zónu *miton.eu* je tento počítač primárním name serverem. Za *file* byl uveden zónový soubor pro danou zónu *miton.eu*. Parametr *notify* nastavený na *yes* umožňuje zasílání informací o změně zóny. Definice druhého pohledu se od prvního liší v názvu, položce *match-clients* a rozdílném zónovém souboru v bloku zone. Tento pohled byl nazván *all*. Položka *match-client* obsahuje parametr *any*. Tento parametr umožňuje přístup všem uživatelům krom těch, kteří jsou ve výčtu předchozího pohledu. Zónový soubor pro tento pohled se jmenuje *miton.eu.other*.

Zónový soubor se skládá z jednotlivých zdrojových záznamů (anglicky resource rekord - RR) obsahující dílčí informace o jednotlivých doménách. Jsou to doménové jméno, životnost v sekundách, třída rodiny protokolů, typ záznamu a parametry vztahující se k typu záznamu.

Zónový soubor *miton.eu* zajišťující obsluhu pro uživatele z České republiky:

```
localhost 1D IN A 127.0.0.1
miton.eu. 1D IN A 82.100.58.131
* 1D IN A 82.100.58.131
@ 1D IN SOA ns.miton.cz. hostmaster.miton.cz. (
                        2007022115 ; serial
                        8H ; refresh
                        2H ; retry
                        1W ; expire
                        1D ; default_ttl
                        )
@ 1D IN MX 5 mx.miton.cz.
@ 1D IN MX 10 mx.mitoncz.com.
@ 1D IN NS barbucha.miton.cz.
@ 1D IN NS ns.mitoncz.com.
cdn 10M IN A 87.236.198.195
        IN A 88.86.107.35
        IN A 82.100.58.155
cdn1 1H IN CNAME amalka.miton.cz.
cdn2 1H IN CNAME skubanek.miton.cz.
cdn3 1H IN CNAME rumcajs.miton.cz.
cdn4 1H IN CNAME machal.miton.cz.
```

Konfigurační soubor obsahuje definici domény *localhost* s životností záznamu jeden den a IP adresou *127.0.0.1*. Tato doména byla nastavena jen pro interní účely při správě serveru. Další doméně *miton.eu* byla nastavena také životnost záznamu jeden den a byla

ji přiřazena IP adresa *82.100.58.131*. Definice uvozená * umožňuje převzít doménové jméno z následujících nastavení, které jsou uvozeny @ . Vzhledem k tomu, že této definici byla přiřazena stejná IP adresa jako doméně *miton.eu*, tak všechny následující definice jsou doplňujícími nastaveními této domény. Toto nastavení bylo provedeno, aby bylo možné použít aliasů serverů podřízených doméně *miton.cz*. Řádek definice s parametrem *SOA* nastavuje parametry primárního serveru a jedná se o zahajující záznam zónového souboru. Obsahuje jméno autoritativního serveru *ns.miton.cz* a email na správce *hostmaster.miton.cz*, pro zasílání chyb. Pravidla zápisu vyžadují místo @ použít tečku. Dalším parametrem je sériové číslo záznamu, které je nutné při každé úpravě nahradit jiným. Podle sériového čísla pozná sekundární server, že došlo ke změně a zahájí synchronizaci údajů. Parametr *refresh* určuje jak často se má sekundární server dotazovat na novou verzi definice zóny. Byl nastaven na osm hodin. Hodnota *retry* ukládá sekundárnímu serveru po jakých intervalech má opakovat své pokusy, pokud se mu nedaří kontaktovat primární server. Pro tento stav byl nastaven čas dvě hodiny. Položka *expire* určuje čas po kterém označí sekundární server své záznamy za neaktuální, pokud se nedaří kontaktovat primární server. Tato doba byla nastavena na jeden týden. Implicitní doba platnosti záznamů *default_TTL* byla nastavena na jeden den. Typ záznamu uvozený *MX* oznamuje adresu a prioritu serveru, nižší číslo vyšší priorita, pro příjem elektronické pošty pro danou doménu. Byly definovány dva servery pro příjem elektronické pošty. Interní *mx.miton.cz* s vyšší prioritou a externí *mx.mitoncz.com*. Následují dva záznamy s typu NS obsahující jméno serveru, který o doméně poskytuje autoritativní informace. Byly nastaveny tyto dva servery: interní *barbucha.miton.cz* a externí *ns.mitoncz.com*. Další položkou je definice domény *cdn*. Platnost záznamu byla nastavena na deset minut. K doméně *cdn* je možné přistupovat ze třech různých IP adres *87.236.198.195*, *88.86.107.35* a *82.100.58.155*. Doména *cdn* byla definována jako rozhraní pro vstup do sítě pro doručování obsahu (viz 4.3). Dále jsou definice doménových jmen jednotlivých datových serverů zařazených v síti CDN. Podle částí aliasů je zřejmé, že se jedná o servery interní sítě. Jednotlivé domény pro tuto zónu byly nazvány *cdn1*, *cdn2*, *cdn3*, *cdn4*. Typ záznamu *CNAME* umožnil místo IP adres serverů použít jejich alias. Doba platnosti záznamu byla nastavena na jednu hodinu.

Zónové soubory pro oba pohledy se liší pouze v jedné definici. Jedná se

o definici doménového jména *cdn*. V zónovém souboru *miton.eu.others* byla tomuto doménovému jménu přiřazena pouze jedna IP adresa *82.208.49.149*. Jedná se o IP adresu, která byla vyloučena z výčtu adres pro přístup do pohledu *cz*. Doba platnosti záznamu byla nastavena na dvě minuty.

4.1.2 Rozklad zátěže, monitoring – HAProxy

Na stejném stroji, kde byl zprovozněn DNS server byl nainstalován reversní TCP/HTTP proxy server HAProxy verze 1.1.23. HAProxy zajišťuje sledování stavu jednotlivých serverů a v případě výpadku provede přesunutí služby na druhý aktivní stroj. Rozděluje zátěž (viz 3.1) mezi jednotlivé webové servery. Umožňuje sledování stavu jednotlivých serverů přes webové rozhraní (viz Obr 4-2). Běží na pozadí operačního systému jako takzvaný *deamon*.

HAProxy

Statistics Report for pid 29836

> General process information

pid = 29836 (nbproc = 1)
uptime = 10d 4h47m08s
system limits : memmax = unlimited ; ulimit-n = 16272
maxsock = 16272
maxconn = 8128 (current conns = 5)

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
backup UP
backup UP, going down
backup DOWN, going up
not checked

> Proxy instance wwwclusterssl : 0 conns (maxconn=8000), 0 queued (0 unassigned), 9922 total conns

Server					Queue		Sessions				Errors				
Name	Weight	Status	Act.	Bck.	Curr.	Max.	Curr.	Max.	Limit	Cumul.	Conn.	Resp.	Sec.	Check	Down
liberty1	8	UP	Y	-	0	0	0	2	-	4392	0	0	0	1014	13
liberty2	10	UP	Y	-	0	0	0	2	-	5478	0	0	0	1041	13
Dispatcher	-	UP	-	-	0	0	0	3	8000	52	3	0	0	-	-
Total	-	**UP**	**2**	**0**	**0**	**0**	**0**	**3**	**8000**	**9922**	3	0	0	2055	26

> Proxy instance wwwcluster : 5 conns (maxconn=8000), 0 queued (0 unassigned), 1644727 total conns

Server					Queue		Sessions				Errors				
Name	Weight	Status	Act.	Bck.	Curr.	Max.	Curr.	Max.	Limit	Cumul.	Conn.	Resp.	Sec.	Check	Down
liberty1	8	UP	Y	-	0	0	2	40	-	729773	6	53	0	580	2
liberty2	10	UP	Y	-	0	0	2	52	-	912190	8	57	0	628	3
Dispatcher	-	UP	-	-	0	0	1	93	8000	2764	452	0	0	-	-
Total	-	**UP**	**2**	**0**	**0**	**0**	**5**	**93**	**8000**	**1644727**	466	110	0	1208	5

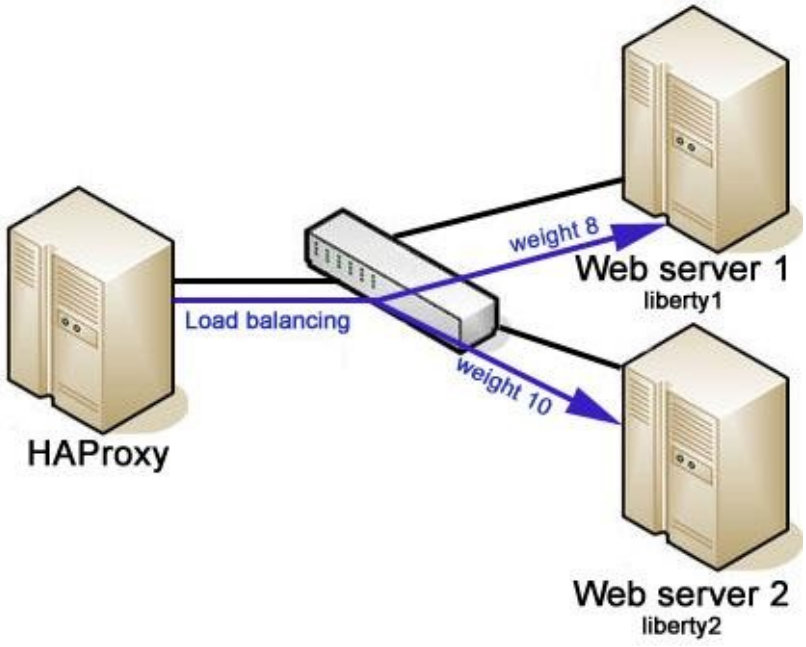
Obr. 4-2 – HAProxy – sledování stavu jednotlivých serverů

Sledování stavu jednotlivých strojů probíhá na stejných spojeních jako jsou posílány požadavky uživatelů a následné odpovědi serverů. Pro potřeby webové aplikace byla vytvořena dvě různá spojení jedno veřejné a druhé zabezpečené protokolem SSL. Protokol SSL (secure sockets layer) poskytuje zabezpečení

komunikace šifrováním přenášených dat a autentizací komunikujících stran. K monitorování využívá protokolu TCP/HTTP. Kontrolní dotazy na stav zasílá webovému serveru na jeho IP adresu a stejným způsobem, je-li vše v pořádku, dostává odpověď zpět. HAProxy samostatně, bez zázahu obsluhy, sbírá aktuální informace o jednotlivých serverech, které vyhodnocuje. V případě detekce chyby automaticky převede služby z havarovaného uzlu na aktivní a interaktivně zobrazí informace o situaci přes webové rozhraní. Tato akce trvá nejdéle dvacet sekund. HAProxy poskytuje administrátorovi následující informace (viz obr 4-2). V poli General process information jsou to identifikační číslo proxy serveru, doba běhu clusteru bez výpadku, nastavení systémových limitů pro komunikaci a maximální počet paralelních spojení. V tabulkách jsou uvedeny informace o konfiguraci, aktuálním stavu a zatížení jednotlivých komunikačních kanálů. Položka server obsahuje názvy serverů zařazených v clusteru, váhu pro rozdělování zátěže, stav serveru a informaci zda se jedná o záložní nebo aktivní stroj. Část *Queue* udává počty aktuálně zpracovávaných dotazů a statistický údaj o maximálním množství požadavků. Sekce Sessions informuje o aktuálním a nejvyšším počtu otevřených sezení, limitující a souhrnné množství všech session. Poslední sloupec Errors poskytuje informace o počtu neúspěšných spojení a neodeslaných odpovědí, udává počet dlouhodobých výpadků, neúspěšných kontrol a celkové množství havárií. Podle zbarvení řádku lze okamžitě registrovat stav daného serveru. Poslední řádek tabulky udává součtové hodnoty jednotlivých sloupců.

Nástroj HAProxy zajišťuje rozklad zátěže (anglicky load balancing viz 3.1) mezi jednotlivé webové servery (viz Obr. 4-3, [20]). Byl použit algoritmus weighted roud robin (viz 3.1.2). Tento postup rozděluje příchozí požadavky na základě vah přidělených jednotlivým serverům. Zátěž je rozdělována mezi dva webové servery *liberty1.miton.cz* a *libery2.miton.cz*. Prvnímu byla přidělena váha 8 a druhému 10. Celkově jsou požadavky rozdělovány v poměru 4:5. Reálně jsou však požadavky přidělovány podle následující sekvence *AB AB AB AB AB AB AB AB BB*…. Písmeno *A* představuje požadavek přidělený serveru *libert1* a písmeno *B* dotaz na server *liberty2*. Směrování jednotlivých datagramů je realizováno podle pravidel metody NAT (*network address translation*, viz 3.2.3). Servery jsou zapojeny v jedné síti a proxy server předává přijaté požadavky jménem původního klienta. Jedná se o princip transparentní proxy. Datagramy s odpovědí jsou webovými servery přeposílány zpět na proxy server, který je

směruje a přeposílá koncovým uživatelům. Webové servery tak tvoří autonomní systém což splňuje požadavky clusteru.



Obr. 4-3 – HAProxy – Load balancing, Weighted round robin

Konfigurační soubor HAProxy (viz Příloha B) obsahuje definici globálních parametrů, které ovlivňuje chování celého procesu. Implicitní parametry upřesňují globální nastavení a kriteria pro komunikaci s webovými servery. Základní nastavení je v konfiguračním souboru uvozeno hlavičkou *global*.

```
global
    log 127.0.0.1 local0
    maxconn 8128 #4096
    chroot /usr/share/haproxy
    uid 99
    gid 99
    daemon
    #debug
    #quiet
```

Tato položka obsahuje parametr *log*, který je definován IP adresou *127.0.0.1* a název zařízení *local0* pro hlášení průběhu jednotlivých operací. Hodnota *8128* uvozená *maxconn* je limitem pro počet spojení na jednu operaci. Parametr *chroot* nastavuje přístupová práva ke kořenovému adresáři */usr/share/haproxy* podle identifikátorů uživatele *uid 99* a skupiny *gid 99*. Položka *deamon* nastavuje proxy server, aby běžel jako demon na pozadí operačního systému.

Oblast *defaults* obsahuje nastavení pro rozsah a druh informací ukládaných hlášení a parametry pro sledování jednotlivých spojení definovaných v části *listen*.

```
defaults
    log global
    mode http
    option httplog
    option dontlognull
    option httpclose
    option forwardfor
    retries 3
    redispatch
    maxconn 8000 #4000
    contimeout 5000
    clitimeout 50000
    srvtimeout 50000
```

Hlášení jednotlivých spojení budou zaznamenávány na stejné místo jako hlášení celého systému *log global*. Položka *mode* nastavuje zdroj informací pro monitorování. Data budou získávána z hlaviček odpovědí webového serveru http header, proto parametr *http*. Nastavení *httplog* definuje rozsah informací pro hlášení. Hodnota *dontlognull* zajišťuje neukládání prázdných hlášení. *Httpclose* zahrnuje do hlášení uzavření spojení. Položka *forwardfor* přidává do hlášení IP adresu klienta. Následující parametry definují chování proxy serveru vzhledem k přijímání a odesílání jednotlivých požadavků. Parametr *retries 3* určuje počet pokusů o opětovné navázaní spojení. *Redispatch* zajišťuje, aby při přebíraní služby byly převedeny i příslušné záznamy cookies. *Maxconn 8000* je limitní hodnota pro paralelní počet spojení proxy serveru pro jednu sledovanou instanci. *Contimeout 5000* určuje dobu jednoho pokusu pro navázání spojení v milisekundách (ms). Parametr *Clitomeout 50000* (ms) udává dobu po kterou proxy server čeká na přijetí nebo odeslání dat od klienta. *Svrtimeout 50000* (ms) definuje dobu po kterou proxy server na přijetí nebo odeslání dat o serveru.

Sekce *listen* definuje instance jednotlivých sledovaných služeb.

```
listen wwwcluster 82.100.58.144:80
    mode http
    cookie SERVERID rewrite
    balance roundrobin
    server liberty1 82.100.58.213:80 cookie cookieliberty1
weight 8 check inter 2000 rise 2 fall 5
   server liberty2 82.100.58.214:80 cookie cookieliberty2 weight
10 check inter 2000 rise 2 fall 5
    stats enable
    stats uri /stats
    stats auth admin:natalka
```

Klíčové slovo *listen* uvozuje název instance *wwwcluster*, IP adresu a port *82.100.58.144:80* na které proxy server poslouchá. *Mode http* definuje způsob komunikace mezi proxy serverem a jednotlivými webovými servery. Informace jsou předávány ve formátu HTML na úrovni aplikační vrstvy. *Cookie SERVERID rewrite* nastavuje způsob práce s jednotlivými záznamy cookies v případě havárie jednoho z webových serverů. Parametr *rewrite* této položky umožňuje přesunout záznam cookie i s celým dotazem na aktivní server. HTTP cookies umožňují serveru uchovat si informace o stavu spojení na počítači uživatele. *Balance roud robin* definuje algoritmus pro rozkládání zátěže mezi jednotlivé servery. Klíčové slovo *server* uvozuje definici jednotlivých strojů, které proxy server monitoruje. Následuje identifikační jméno serveru *liberty1,* jeho IP adresa *82.100.58.213* a port *80,* na které server přijímá a odesílá požadavky. *Cookie* obsahuje proměnnou *cookielibrty1* pro zápis nebo čtení záznamů cookie. *Weight 8* udává váhu, se kterou jsou na server přidělovány jednotlivé požadavky. Další parametry jsou používány pro monitorování stavu serveru. Klíčové slovo *check* povoluje sledování serveru. *Inter 2000* určuje interval mezi jednotlivými testy v milisekundách. Parametr *fall 5* udává počet neúspěšných testů před zahájením procesu přenesení služeb poskytovaných sledovaným strojem na server aktivní. Hodnota *rise 2* definuje počet úspěšných testů před plným nasazení stroje zpět do provozu. Zasílání kontrolních zpráv je realizováno na úrovni aplikační vrstvy OSI modelu. Proxy a webový server si mezi sebou zasílají cookies. Za definicí sledovaných webových serverů následuje nastavení statistik pro zobrazení informací přes webové rozhraní. Definice jsou uvozeny klíčovým slovem *stats*. Hodnota *enable* povoluje tuto službu. *Uri/stats* definuje webovou adresu pro přístup k těmto informacím. Statistiky jsou přístupné na *http://82.100.58.144/stats*. Poslední parametry udávají přístupové jméno a heslo. Byla nadefinována také instance pro zabezpečený přístup *wwwslusterssl*. Tato instance se liší pouze v následujících parametrech.

```
mode tcp
option ssl-hello-chk
```

Parametr *mode tcp* zajišťuje komunikaci na úrovni transportní vrstvy pomocí protokolu TCP. Komunikace probíhá na stejných IP adresách, ale na zabezpečeném portu *443*. Port 443 je zabezpečen vrstvou SSL. SSL (secure socket layer) je protokol, resp. vrstva vložená mezi vrstvu transportní a aplikační, která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran. Nastavení *option ssl-hallo-chk* umožňuje

monitorování stavu zabezpečeného spojení na úrovni transportní vrstvy OSI modelu. Proxy server odešle webovému serveru zprávu SSL CLIENT HELLO a je-li vše v pořádku obdrží od webového serveru odpověď HELLO.

4.2 Webový, databázový server a replikace databáze

Hlavní část clusteru tvoří webové a databázové servery. Webový server zajišťuje běh aplikace. Společně s databází zpracovává dotazy od uživatelů a odpovědi přeposílá zpět. Na webové servery byl uložen kompletní kód aplikace a data potřebná k zobrazení statického obsahu stránek. Ostatní data jsou uložena na datových serverech v síti CDN (viz 4.3). Databázový server obsahuje záznamy potřebné pro běh aplikace a uživatelská data.

4.2.1 Webový server – Apache

Cluster obsahuje dva stroje na než byl nainstalován webový server Apache verze 2.2.3. Apache je softwarový webový server s veřejně dostupným kódem. Webový server zajišťuje běh aplikace. Byla provedena naprosto shodná konfigurace, protože na obou serverech je provozována stejná webová aplikace. Každý ze serverů obsahuje všechny části kódu aplikace a grafické soubory, které dávají vzhled jednotlivým částem aplikace. Změny nebo doplňky některých z těchto částí jsou nahrávány na oba servery současně pomocí nástroje SVN. Nástroj SVN je systém pro správu zdrojových kódů. Změny jsou nejprve nataženy do repositáře a pak najednou distribuovány na oba stroje tak, aby se dostávali na servery ve stejných kopiích.

Servery byly pojmenovány *liberty1.miton.cz* a *libery2.miton.cz* a byly shodně nakonfigurovány. Konfigurační soubor (viz. Příloha C) je rozdělen na tři části.

Část *Global Enviroment* obsahuje hlavní nastavení serveru. Je zde definována cesta do kořenového adresáře, kam byly umístěny soubory webové aplikace. Dále to jsou časové a kvantitativní parametry, které určují chování serveru při zpracovávání jednotlivých dotazů. Je zde definováno číslo portu, na kterém server poslouchá. Je jím standardní port 80. Sekce obsahuje také definici uživatele a skupiny, které byla přidělena přístupová práva pro správu serveru. Největší část této sekce zabírá seznam takzvaných modulů, které server při spuštění načítá. Moduly jsou podprogramy, které doplňují jádro Apache, zajišťují a rozšiřují jeho funkcionalitu pro požadavky webové aplikace.

Main server configuration je další sekcí konfiguračního souboru. Tato část

obsahuje název serveru, nastavení přístupových práv do jednotlivých adresářů webového serveru a doplňující konfigurace modulů. Jsou zde uvedeny definice jednotlivých datových typů pro modul mod_mime_magic. Modul umožní serveru rozeznávat jednotlivé datové formáty. Dále jsou definovány světové jazyky, což serveru umožňuje poslat uživateli datový obsah v jazyce, kterému bude rozumět. Sekce obsahuje nastavení, které umožňuje sledovat stav serveru z definovaných IP adres.

Virtual Hosts je poslední sekce, která obsahuje definici jednotlivých virtuálních serverů. Konfigurací virtuálních serverů je možné provozovat více domén pod jednou IP adresou na jednom webovém serveru. Pomocí direktivy *include* byly jednotlivé definice (viz Příloha C) přesunuty do externího adresáře na serveru. Byly definovány celkem tři virtuální servery *liberty1.miton.cz*, *stage.cz* a *stage.cz* zabezpečený protokolem SSL (viz výše). Virtuální server *liberty1.miton.cz* zobrazí nastavení serveru přes webové rozhraní, což je užitečné při ladění webové aplikace. Definice virtuálního serveru *stage.cz* obsahuje výčet aliasů a nastavení přístupových práv ke kořenovému adresáři webové aplikace. Direktiva *php_admin_value open_basedir* umožňuje serveru zpracovávat PHP kód, ve kterém je aplikace napsána. Zabezpečený virtuální server *stage.cz* přijímá dotazy na portu *443* na rozdíl od nezabezpečené verze. Konfigurace navíc obsahuje spouštěcí direktivu *SSLEngine ON*, která spouští komunikaci po zabezpečeném kanálu. Dále obsahuje cestu k certifikátu a klíči SSL.

4.2.2 Databáze – MySQL

Databázové servery MySQL verze 5.0.22 byly nainstalovány na stejné stroje jako webové servery. Databáze používá tabulky typu InnoDB. Na každém serveru je nutné udržet shodná data, aby nebyla narušena konsistence databáze. Struktura databázového systému umožňuje paralelní čtení, zápis však musí být unikátní v rámci obou databází. Informace o novém záznamu je zapsána do souboru *mysqld.log* databázového serveru *db1* a na server *db2* je odeslán příkaz na synchronizaci dat. Server *db2* zpracuje požadavek a na základě záznamu v soboru *mysqld.log* serveru *db1* zahájí synchronizaci dat. Výsledkem tohoto procesu je opětovná shoda obou databází. Konfigurací bylo zajištěno, aby proces replikace dat fungoval obousměrně. Databáze pracuje buď v režimu *master* nebo *slave*. Režim *master* umožňuje čtení i zápis. Režim *slave* povoluje pouze čtení. Oba servery byly nastaveny tak, aby mohly zastávat obě funkce. V úloze *master* však může pracovat pouze jeden z nich, což zaručí konsistenci

dat. V případě výpadku jednoho ze strojů je role *master* převedena na druhý (viz 4.2.3).

Konfigurační soubor (viz. Příloha D) obou databázových serverů obsahuje definici režimu *master* i *slave*. Toto nastavení vyžadovalo následující konfiguraci sekce *[mysqld]* konfiguračního souboru.

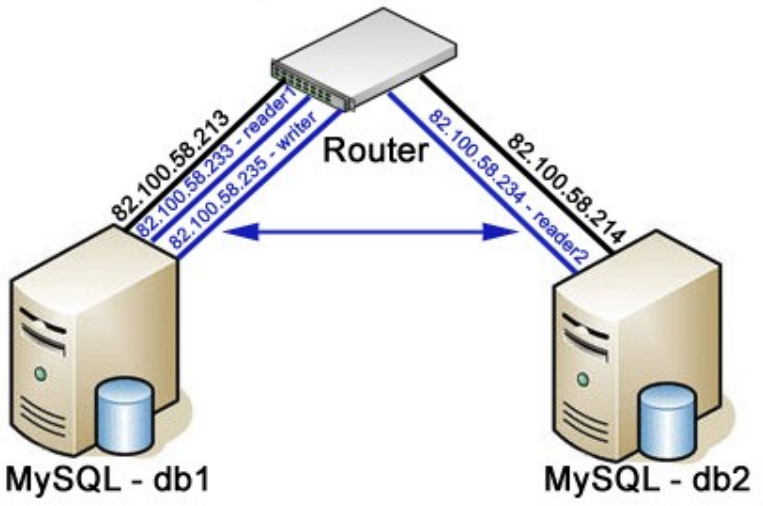
```
[mysqld]
server-id=1
log_bin
replicate-do-db=tmobile_liberty

master-host=82.100.58.214
master-user=replication
master-password=tojeheslo
```

Server byl jednoznačně identifikován parametrem *server-id=1*. Parametrem log_bin bylo zapnuto zaznamenávání změn databáze do souboru *mysqld.log*. Příkaz *replicate_do_db=tmobile_liberty* vyvolá replikaci dat do databáze *tmobile_liberty* na serveru *db2*. Tím byla nastavena role *master*. Definice režimu *slave* začíná parametrem *master-host=82.100.58.214*, který udává IP adresu serveru *db2*. Záznamy *master-user=replication* a *master-password=tojeheslo* je definován uživatel serveru *db1*, který má práva číst záznamy ze souboru *mysld.conf* na jejichž základě provede synchronizaci dat. Stejný uživatel je definován na serveru *db2* s tím rozdílem, že parametru *master-host* byla nastavena IP adresa serveru *db1 82.100.58.213*. Server *db2* byl jednoznačně identifikován parametrem *server-id=2*. Adresářová struktura obou serverů je shodná.

4.2.3 MMM – Master-Master replication manager

MMM (MySQL Master-Master Replication Manager) je soubor skriptů napsaných v interpretovaném skriptovacím jazyce Perl. Zajišťuje monitorování a zprávu replikace dat mezi dvěma databázovými uzly clusteru, které mohou pracovat v režimu *master*. Umožňuje automatické přenesení služeb v případě výpadku některého z uzlů. Obsahuje části kódu zajišťující zálohu a synchronizaci dat po obnovení běhu havarovaného serveru. Synchronizace databáze je realizována na základě záznamů v log souboru. Pro správu dvou databázových uzlů využívá MMM celkem pěti IP adres (viz Obr. 4-4, [20]). Pro monitorování stavu jednotlivých serverů byly použity dvě permanentní IP adresy pro každý z uzlů. Dále byly nastaveny tři virtuální IP adresy, dvě umožňující čtení z databáze pro režim *slave* a jedna zajišťující zápis do databáze pro režim *master*. Tyto IP adresy byly vytvořeny jako alias připojení k jednotlivým uzlům. V závislosti na dostupnosti jednotlivých uzlů je možné virtuální IP adresy dynamicky

přidělovat oběma databázovým serverům. Software pracuje na úrovni transportní vrstvy modelu OSI.



Obr. 4-4 – MMM – MySQL Master-Master replication manager

MMM pracuje ve dvou režimech. První režim umožňuje monitorování jednotlivých uzlů přes spojení na permanentních IP adresách. V případě výpadku některého z uzlů zajistí přeposílání dotazů na běžící server, přičemž IP adresa spojení zůstává stejná. Po obnově havarovaného uzlu proběhne synchronizace databáze a je mu přidělena IP adresa pro čtení z databáze. Druhý režim umožňuje komunikaci s monitorovacím zařízením. Konfigurace obsahuje informace nezbytné pro realizaci synchronizace obsahu databáze mezi jednotlivými uzly. Tento sytém umožňuje rozklad zátěže mezi jednotlivé uzly.

Monitorovací část byla umístěna na router, tedy zařízení, které je přímo spojeno s oběma databázovými servery a je součástí clusteru. Běží na pozadí operačního systému jako takzvaný *deamon*. MMM instalované na routeru obsahuje konfigurační soubor (viz Příloha E) *mmm_mon.conf*. Konfigurace obsahuje definici cest do souborů, kam jsou zaznamenávány informace o změnách v databázi a sytému. Dále obsahuje informace o jednotlivých strojích a nastavení parametrů.

```
# MMMD command socket tcp-port
agent_port 9989
monitor_ip 82.100.58.254
```

Agent_port 9989 definuje komunikační port pro monitorování jednotlivých uzlů z IP

adresy routeru *monitor_ip 82.100.58.254*.

```
# Cluster hosts addresses and access param
host db1
    ip 82.100.58.213
    port 3306
    user rep_monitor
    password RepMonitor
    mode master
    peer db2

host db2
    ip 82.100.58.214
    port 3306
    user rep_monitor
    password RepMonitor
    mode master
    peer db1
```

Položky uvozené klíčovým slovem *host* uvozují definice uzlů *db1* a *db2*. Obsahuje informace o jejich permanentních IP adresách a portech, na kterých probíhá monitorování a synchronizace databáze. *User rep_monitor* a *password RepMonitor* je definice uživatele, který má nastavena práva ke komunikaci s databází monitorovaných uzlů. *Mode master* nastavuje serverům režim práce. Poslední položka *peer db2/db1* identifikuje server pro synchronizaci databáze po havárii.

```
# Define roles
active_master_role writer

# Mysql Reader role
role reader
    mode balanced
    servers db1, db2
    ip 82.100.58.233, 82.100.58.234

# Mysql Writer role
role writer
    mode exclusive
    servers db1, db2
    ip 82.100.58.235
```

Další částí, kterou konfigurační soubor obsahuje je definice jednotlivých rolí přidělená virtuálním IP adresám. Tuto část uvozuje direktiva *aktive_master_role writer*, která zajišťuje přenesení práv pro zápis do databáze na běžící server v případě, že dojde k havárii jednoho z uzlů. Úloha *reader* byla definována pro čtení z databáze pro dvě virtuální IP adresy *82.100.58.233* a *82.100.58.234*. *Mode balanced* umožňuje paralelní přeposílání požadavků na obě IP adresy. Položka *servers db1, db2* definuje servery, kterým je možné přidělovat roli *reader*. Úloha *writer* byla nastavena pro zápis do databáze. Povoluje zápis jen z jediné IP adresy *82.100.58.235*, aby nebyla narušena

konzistence databáze. Tato role může být přidělena pouze jednomu ze serverů *db1* nebo *db2*, což definuje parametr *mode exklusive*. Následující části konfiguračního souboru definují časové parametry pro kontrolu stavu jednotlivých serverů.

Druhá část, takzvaný *agent*, byla nainstalována na jednotlivé uzly. Běží na pozadí operačního systému jako takzvaný *deamon*. Zajišťuje komunikaci s monitorovacím zařízením a zahajuje synchronizaci databáze po havárii. Konfigurace je umístěna v souboru *mmm_agent.conf* (viz Příloha E). Nastavení jednotlivých serverů obsahuje jednak informace o umístění souborů, kam se zaznamenávají změny v databázi a chybová hlášení a jednak číslo portu, na kterém komunikace s monitorovacím zařízením probíhá.

Server *db1*:

```
# Define current server id
this db1
mode master

# For masters
peer db2
```

Server *db2*:

```
# Define current server id
this db2
mode master

# For masters
peer db1
```

Konfigurační soubor *agenta* obsahuje identifikační záznam *this db1 / this db2* charakterizující daný server. Položka *peer db2 / peer db1* definuje zdrojový uzel pro replikaci databáze. Dále jsou uvedeny přístupová nastavení obou serverů. Jedná se o IP adresu, komunikační port, jméno a heslo uživatele pro přístup do databáze.

```
# Cluster hosts addresses and access params
host db1
    ip 82.100.58.213
    port 3306
    user rep_monitor
    password RepMonitor

host db2
    ip 82.100.58.214
    port 3306
    user rep_monitor
    password RepMonitor
```

4.3 Datové servery a CDN

Datové servery byly umístěny mimo cluster a dotváří systém s vysokou dostupností pro běh webové aplikace. Jednotlivé stroje byly zařazeny do sítě pro doručování obsahu CDN (Contend delivery network viz 3.3.3). CDN obsahuje celkem čtyři servery se stejným datovým obsahem, což umožňuje rozložit zátěž mezi jednotlivé stroje a distribuovat větší datové celky v závislosti na geografické poloze koncového uživatele. Výběr vhodného datového serveru pro doručení žádaných dat zajistí DNS server podle IP adresy koncového uživatele (viz 4.1.1). DNS server pracuje s cyklickým seznamem datových serverů. Podaří-li se navázat spojení s vybraným datovým serverem za daný časový úsek začne vlastní přenos dat. V opačném případě je dotazován další server ze seznamu. Monitoring stavu jednotlivých serverů je zajištěn nástrojem Nagios. V případě výpadku zašle administrátorovi zprávu o havarii. Administrátor se pak musí postarat znovu uvedení serveru do běhu nebo jej v případě závažnější chyby vyřadit z cyklického seznamu DNS serveru.

Průběh nahrávání dat na jednotlivé datové servery probíhá ve dvou fázích. V první fázi jsou data z webového rozhraní uložena na primární datový server. Přenos dat probíhá na spojení zabezpečeném protokolem SSL. V druhé fázi následuje proces replikace dat na zbylé stroje vnitřní sítě CDN. Na jednotlivé datové servery byl nainstalován FTP server ProFTPD a vytvořeny uživatelské účty zabezpečené jménem a heslem. Synchronizace dat je realizována pomocí skriptu napsaném v jazyce Perl (viz Příloha F), který pro přenos dat využívá právě protokolu FTP. Skript pracuje s frontou záznamů typu FIFO, takzvanou *rourou*, která je vytvořena v log souboru FTP serveru *proftpd-log.fifo*. Každý záznam obsahuje potřebné informace o změně datového obsahu na primárním serveru. Jedná se o typ provedené operace a informace o cestě k souboru s nímž bylo manipulováno. Skript při prvním spuštění naváže spojení na IP adresy datových serverů. Toto spojení uchovává formou takzvaných vláken, anglicky thread, po celou dobu běhu programu. Následuje přihlášení na účty FTP serverů. Po úspěšné autentizaci začíná přenos dat. Pomocí vláken je sledován stav jednotlivých spojení. V případě výpadku je přenos dat přerušen a odesláno chybové hlášení. Skript po prvním spuštění běží na pozadí operačního systému jako deamon a spojení na jednotlivé servery udržuje stále aktivní. Každá změna v datovém obsahu na primárním datovém serveru automaticky vyvolá replikaci dat na zbylé stroje v síti CDN. Program umožňuje

vytvářet adresáře, vkládat a mazat soubory.

Každý server obsahuje diskové pole složené ze čtyř diskových jednotek o kapacitě 250 GB. Jedná se o disky typu SATA zapojené v architektuře RAID 5 (viz 3.3.4). Jednotlivé disky jsou připojeny ke speciálnímu řadiči, který diskové pole prezentuje operačnímu systému jako jeden celek. Jedná se hardwarový RAID. Celková kapacita diskového pole je 1 TB.

Za účelem přehrávání hudby a videa přímo ve webové aplikaci byl na datové servery nainstalován webový server Lighttpd. Na těchto serverech jsou umístěny PHP skripty, které zajišťují přehrávání video a hudebních souborů přímo ze serveru. Přístup k aplikacím byl umožněn pomocí protokolu FastCGI. Spouštění skriptů je přístupné pomocí URL adresy s cestou do souborů na serveru. V konfiguračním souboru Lighttpd (viz příloha G) bylo nutné načíst modul *mod_fastcgi*. Následující definice povoluje použití modulu.

```
#### fastcgi module

fastcgi.server = ( ".php" =>
                   ( "localhost" =>
                    (
                      "socket" => "/tmp/php- fastcgi.socket",
                      "bin-path" => "/usr/bin/php-cgi"
                    )
                   )
                  )
```

4.4 Testování

Na vytvořeném systému s vysokou dostupností byl nasazen hudební portál stage.cz v testovacím provozu. Tato webová aplikace spravuje uživatelské účty, zajišťuje přehrávání hudby a videa přímo z datových serverů. Od spuštění byl běh aplikace několikrát na krátkou dobu přerušen z důvodu ladění, údržby sytému a doinstalování nových částí. Celková dostupnost však zatím neklesla pod požadovaných 99,5 %.

Testování funkčnosti komponent pro zajištění vysoké dostupnosti proběhlo úspěšně v rámci jejich implementace do systému. HAProxy, která zajišťuje rozklad zátěže, monitorování a předávání služeb v případě havárie některého ze serverů, reaguje na výpadek do deseti sekund. Havarovaný server se za stejný čas začíná opět podílet na zpracovávání požadavků. MMM replication manager, který zajišťuje monitorování a předávání služeb databázových serverů má reakční dobu kolem pěti sekund. Pouze

obnova činnosti je o něco delší, protože nejprve probíhá replikace databáze. Při testování tento proces nepřesáhl dobu třiceti sekund. Těchto vlastností je s výhodou využíváno při aktualizaci jednotlivých komponent serverů.

Závěr

Výsledkem diplomové práce nazvané „Webové aplikace s vysokou dostupností“ je návrh a realizace systému s vysokou dostupností, na kterém je provozována webová aplikace.

Na základě získaných poznatků byl vytvořen návrh systému s vysokou dostupností pro požadavky webové aplikace. Navržený a později vytvořený systém lze rozdělit na dvě úzce spolupracující části. První část představuje cluster, který obsahuje webové a databázové servery. Druhou část tvoří datové servery zapojené ve speciální síti uzpůsobené pro doručování obsahu do webové aplikace. Při vlastní realizaci bylo použitím vhodných nástrojů dosaženo odolnosti systému vůči výpadku jednotlivých uzlů.

Na realizovaném systému s vysokou dostupností je provozována webová aplikace dostupná na doméně www.stage.cz. Jedná se o projekt, který představuje hudební portál. Uživatelům nabízí kromě běžných funkcí komunitních webů i přehrávání videa a hudby. V době dokončení diplomové práce bylo již vytvořeno přes 500 uživatelských účtů a systém zpracovává v průměru 200 000 dotazů denně.

Diplomová práce přináší komplexní řešení pro provoz webové aplikace vyžadující vysokou dostupnost a toleranci vůči výpadkům jednotlivých serverů.

Použitá literatura a zdroje

[1] Evi Nemeth, Gart Snyder, Trent R. Hein, LINUX Kompletní příručka administrátora, Brno, 2004, ISBN 80-722-6919
[2] Pužmanová Rita, Šmrha Pavel, Propojování sítí s TCP/IP, České Budějovice, 1999, ISBN 80-7232-080-7
[3] Kolektiv autorů, Linux dokumentační projekt, BRNO, 2003, ISBN 80-7226-761-2
[4] High availability [online] URL: <http://en.wikipedia.org/wiki/High-availability>
[5] High-availability cluster [online] URL: <http://en.wikipedia.org/wiki/High-availability_cluster>
[6] Computer cluster [online] URL: <http://en.wikipedia.org/wiki/Computer_cluster>
[7] Lewis Phill, A High-Availability Cluster for Linux, [online] URL: <http://www.linuxjournal.com/article/3247>
[8] Monitorování pulsu (heartbeat), [online] URL: <http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=/rzaig/rzaigconceptsheartbeatmonitor.htm>
[9] Tarreau Willy, Making applications scalable with Load Balancing, [online] URL: <http://1wt.eu/articles/2006_lb/index.html>
[10] Wensong Zhang, Shiyao Jin, Quanyuan Wu, (National Laboratory for Parallel & Distributed Processing Changsha) Creating Linux Virtual Servers, [online] URL: <http://www.linuxvirtualserver.org/linuxexpo.html>
[11] Pustka Martin, Rozkládání zátěže a zajištění dostupnosti systémů a služeb v počítačových sítích, [online] URL: <http://www.fs.vsb.cz/akce/2000/asr2000/Sbornik/papers/pustka.pdf>
[12] Linux virtual server Documentation, [online] URL: <http://www.linuxvirtualserver.org/Documents.html>
[13] Network file system, [online] URL: <http://en.wikipedia.org/wiki/Distributed_file_system>
[14] Replication, [online], URL: <http://en.wikipedia.org/wiki/Replication_(computer_science)>
[15] Závodský Daniel, Network File Systém, 2007 [online] URL: <http://www.linux-expres.cz/praxe/network-file-system>
[16] Pužmanová Rita, DAS, SAN, NAS: Varianty řešení ukládání a zálohování dat [online] URL: <http://www.systemonline.cz/site/data-warehousing/04_02puzm.htm>
[17] Content Delivery Network, [online], URL: <http://en.wikipedia.org/wiki/Content_Delivery_Network>
[18] RAID, [online], URL: <http://cs.wikipedia.org/wiki/RAID>
[19] Mark Nottingham, Caching Tutorial, 2007, [online], URL: <http://www.mnot.net/cache_docs/>
[20] Obrazový materiál, [online] URL: <www.wikipedia.org>
[21] www.apache.org
[22] www.isc.org/index.pl?/sw/bind
[23] haproxy.1wt.eu
[24] www.mysql.org
[25] www.lighttpd.net
[26] www.proftpd.org
[27] code.google.com/p/mysql-master-master

Příloha A – Konfigurační soubory BIND DNS serveru

Hlavní konfigurační soubor BIND – named.conf

```
// Default named.conf generated by install of bind-9.2.4-24.EL4
options {
    directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
};
include "/etc/rndc.key";
view cz { match-clients { !82.208.49.146; country_CZ; };
        zone "miton.eu" IN {
                type master;
                file "pri/miton.eu";
                notify yes;
        };
};
view all { match-clients { any; };
    zone "miton.eu" IN {
                type master;
                file "pri/miton.eu.other";
                notify yes;
        };
};
```

Zónový soubor pro pohled view cz – miton.eu

```
localhost 1D IN A 127.0.0.1
miton.eu. 1D IN A 82.100.58.131
* 1D IN A 82.100.58.131
@ 1D IN SOA ns.miton.cz. hostmaster.miton.cz. (
                        2007022115 ; serial
                        8H ; refresh
                        2H ; retry
                        1W ; expire
                        1D ; default_ttl
                        )
@ 1D IN MX 5 mx.miton.cz.
@ 1D IN MX 10 mx.mitoncz.com.
@ 1D IN NS barbucha.miton.cz.
@ 1D IN NS ns.mitoncz.com.
cdn 10M IN A 87.236.198.195
        IN A 88.86.107.35
        IN A 82.100.58.155
cdn1 1H IN CNAME amalka.miton.cz.
cdn2 1H IN CNAME skubanek.miton.cz.
cdn3 1H IN CNAME rumcajs.miton.cz.
cdn4 1H IN CNAME machal.miton.cz.
```

Zónový soubor pro pohled view all – miton.eu.other

```
localhost 1D IN A 127.0.0.1
miton.eu. 1D IN A 82.100.58.131
* 1D IN A 82.100.58.131
@ 1D IN SOA ns.miton.cz. hostmaster.miton.cz. (
                        2007022134 ; serial
                        8H ; refresh
                        2H ; retry
                        1W ; expire
                        1D ; default_ttl
                        )
@ 1D IN MX 5 mx.miton.cz.
@ 1D IN MX 10 mx.mitoncz.com.
@ 1D IN NS barbucha.miton.cz.
@ 1D IN NS ns.mitoncz.com.
cdn 2M IN A 82.208.49.149
cdn1 1H IN CNAME amalka.miton.cz.
cdn2 1H IN CNAME skubanek.miton.cz.
cdn3 1H IN CNAME rumcajs.miton.cz.
cdn4 1H IN CNAME machal.miton.cz.
```

Příloha B – Konfigurační soubor HAproxi

```
# this config needs haproxy-1.1.23
global
    log 127.0.0.1 local0
    maxconn 8128 #4096
    chroot /usr/share/haproxy
    uid 99
    gid 99
    daemon
    #debug
    #quiet
defaults
    log global
    mode http
    option httplog
    option dontlognull
    retries 3
    redispatch
    maxconn 8000 #4000
    contimeout 5000
    clitimeout 50000
    srvtimeout 50000
listen wwwcluster 82.100.58.144:80
    mode http
    cookie SERVERID rewrite
    balance roundrobin
    server liberty1 82.100.58.213:80 cookie cookieliberty1 weight 8 check
           inter 2000 rise 2 fall 5
    server liberty2 82.100.58.214:80 cookie cookieliberty2 weight 10 check
          inter 2000 rise 2 fall 5
    stats enable
    stats uri /stats
    stats auth admin:natalka
 listen wwwclusterssl 82.100.58.144:443
    mode tcp
    cookie SERVERID rewrite
    balance roundrobin
    server liberty1 82.100.58.213:443 cookie cookieliberty1 weight 8 check
           inter 2000 rise 2 fall 5
    server liberty2 82.100.58.214:443 cookie cookieliberty2 weight 10 check
           inter 2000 rise 2 fall 5
    option ssl-hello-chk
```

Příloha C – Konfigurační soubor webového serveru liberty1 a externích definic Virtual host.

httpd.conf – obsahuje jen relevantní položky. Kompletní konfigurační soubor je na přiloženém CD.

```
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.

### Section 1: Global Environment
ServerTokens OS
ServerRoot "/etc/httpd"
PidFile run/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15
Listen 80
<IfModule prefork.c>
    StartServers       8
    MinSpareServers    5
    MaxSpareServers   20
    ServerLimit      256
    MaxClients       256
    MaxRequestsPerChild  4000
</IfModule>
<IfModule worker.c>
    StartServers         2
    axClients          150
    MinSpareThreads     25
    MaxSpareThreads     75
    ThreadsPerChild     25
    MaxRequestsPerChild  0
</IfModule>
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_alias_module modules/mod_authn_alias.so
LoadModule authn_anon_module modules/mod_authn_anon.so
LoadModule authn_dbm_module modules/mod_authn_dbm.so
LoadModule authn_default_module modules/mod_authn_default.so
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule authz_owner_module modules/mod_authz_owner.so
LoadModule authz_groupfile_module modules/mod_authz_groupfile.so
LoadModule authz_dbm_module modules/mod_authz_dbm.so
LoadModule authz_default_module modules/mod_authz_default.so
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
LoadModule include_module modules/mod_include.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule logio_module modules/mod_logio.so
LoadModule env_module modules/mod_env.so
LoadModule ext_filter_module modules/mod_ext_filter.so
LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule expires_module modules/mod_expires.so
LoadModule deflate_module modules/mod_deflate.so
LoadModule headers_module modules/mod_headers.so
LoadModule usertrack_module modules/mod_usertrack.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule mime_module modules/mod_mime.so
LoadModule dav_module modules/mod_dav.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule info_module modules/mod_info.so
LoadModule dav_fs_module modules/mod_dav_fs.so
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
LoadModule actions_module modules/mod_actions.so
```

```
LoadModule speling_module modules/mod_speling.so
LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule cache_module modules/mod_cache.so
LoadModule suexec_module modules/mod_suexec.so
LoadModule disk_cache_module modules/mod_disk_cache.so
LoadModule file_cache_module modules/mod_file_cache.so
LoadModule mem_cache_module modules/mod_mem_cache.so
LoadModule cgi_module modules/mod_cgi.so
Include conf.d/*.conf
ExtendedStatus On
User zakaznik
Group zakaznik

### Section 2: 'Main' server configuration
ServerAdmin root@miton.cz
ServerName liberty1.miton.cz:80
UseCanonicalName Off
DocumentRoot "/var/www/html"
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
<IfModule mod_userdir.c>
    UserDir disable
</IfModule>
DirectoryIndex index.html index.html.var
AccessFileName .htaccess
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
TypesConfig /etc/mime.types
DefaultType text/plain
<IfModule mod_mime_magic.c>
#   MIMEMagicFile /usr/share/magic.mime
    MIMEMagicFile conf/magic
</IfModule>
HostnameLookups Off
ErrorLog logs/error_log
LogLevel warn
LogFormat "i %{X-Forwarded-For}i %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
Agent}i\"" vcombined
LogFormat "%{X-Forwarded-For}i %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
CustomLog logs/access_log combined
ServerSignature On
Alias /icons/ "/var/www/icons/"
<Directory "/var/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
<IfModule mod_dav_fs.c>
    # Location of the WebDAV lock database.
    DAVLockDB /var/lib/dav/lockdb
</IfModule>
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
```

```
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
IndexOptions FancyIndexing VersionSort NameWidth=* HTMLTable
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*
AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core
AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^
DefaultIcon /icons/unknown.gif
ReadmeName README.html
HeaderName HEADER.html
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
AddLanguage ca .ca
AddLanguage cs .cz .cs
AddLanguage da .dk
AddLanguage de .de
AddLanguage el .el
AddLanguage en .en
AddLanguage eo .eo
AddLanguage es .es
AddLanguage et .et
AddLanguage fr .fr
AddLanguage he .he
AddLanguage hr .hr
AddLanguage it .it
AddLanguage ja .ja
AddLanguage ko .ko
AddLanguage ltz .ltz
AddLanguage nl .nl
AddLanguage nn .nn
AddLanguage no .no
AddLanguage pl .po
AddLanguage pt .pt
AddLanguage pt-BR .pt-br
AddLanguage ru .ru
AddLanguage sv .sv
AddLanguage zh-CN .zh-cn
AddLanguage zh-TW .zh-tw
LanguagePriority en ca cs da de el eo es et fr he hr it ja ko ltz nl nn no pl pt
pt-BR ru sv zh-CN zh-TW
ForceLanguagePriority Prefer Fallback
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
AddHandler type-map var
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
Alias /error/ "/var/www/error/"
<IfModule mod_negotiation.c>
<IfModule mod_include.c>
```

```
<Directory "/var/www/error">
    AllowOverride None
    Options IncludesNoExec
    AddOutputFilter Includes html
    AddHandler type-map var
    Order allow,deny
    Allow from all
    LanguagePriority en es de fr
    ForceLanguagePriority Prefer Fallback
</Directory>
#    ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
#    ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
#    ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
#    ErrorDocument 404 /error/HTTP_NOT_FOUND.html.var
#    ErrorDocument 405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
#    ErrorDocument 408 /error/HTTP_REQUEST_TIME_OUT.html.var
#    ErrorDocument 410 /error/HTTP_GONE.html.var
#    ErrorDocument 411 /error/HTTP_LENGTH_REQUIRED.html.var
#    ErrorDocument 412 /error/HTTP_PRECONDITION_FAILED.html.var
#    ErrorDocument 413
     /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
#    ErrorDocument 414
     /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
#    ErrorDocument 415
     /error/HTTP_UNSUPPORTED_MEDIA_TYPE.html.var
#    ErrorDocument 500
     /error/HTTP_INTERNAL_SERVER_ERROR.html.var
#    ErrorDocument 501 /error/HTTP_NOT_IMPLEMENTED.html.var
#    ErrorDocument 502 /error/HTTP_BAD_GATEWAY.html.var
#    ErrorDocument 503 /error/HTTP_SERVICE_UNAVAILABLE.html.var
#    ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var
</IfModule>
 </IfModule>
    BrowserMatch "Mozilla/2" nokeepalive
    BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-
                response-1.0
    BrowserMatch "RealPlayer 4\.0" force-response-1.0
    BrowserMatch "Java/1\.0" force-response-1.0
    BrowserMatch "JDK/1\.0" force-response-1.0
    BrowserMatch "Microsoft Data Access Internet Publishing
                  Provider" redirect-carefully
    BrowserMatch "MS FrontPage" redirect-carefully
    BrowserMatch "^WebDrive" redirect-carefully
    BrowserMatch "^WebDAVFS/1.[0123]" redirect-carefully
    BrowserMatch "^gnome-vfs/1.0" redirect-carefully
    BrowserMatch "^XML Spy" redirect-carefully
    BrowserMatch "^Dreamweaver-WebDAV-SCM1" redirect-carefully
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from .miton.cz 127.0.0.1 195.39.76.220 62.209.200.246
    80.188.39.164 213.168.180.180 85.119.90.202
</Location>

### Section 3: Virtual Hosts
NameVirtualHost *:80
Include conf/vh/
```

Externí konfigurace jednotlivých Virtual host

```
<VirtualHost *:80>
   ### nazvy
   ServerName liberty1.miton.cz
   ServerAlias 82.100.58.213
   ### ostatni
   DocumentRoot "/home/users/miton/liberty1.miton.cz/web/"
</VirtualHost>
<VirtualHost *:80>
   ### nazvy
   ServerName stage.cz
   ServerAlias  *.stage.cz myband.eu *.myband.eu
   liberty.miton.cz
```

```
   ### rewrite
   RewriteEngine on
   RewriteMap  lowercase  int:tolower
 <directory /home/users/liberty/stage.cz/www/ >
   Options FollowSymLinks
   AllowOverride All
 </directory>
   ### ostatni
   DocumentRoot /home/users/liberty/stage.cz/www/
   php_admin_value open_basedir
   "/home/users/liberty/:/tmp/:/usr/share/pear/"
</VirtualHost>
<VirtualHost *:443>
   ### nazvy
   ServerName stage.cz
   ServerAlias  *.stage.cz myband.eu *.myband.eu
   liberty.miton.cz
   ### rewrite
   RewriteEngine on
   RewriteMap  lowercase  int:tolower
 <directory /home/users/liberty/stage.cz/www/ >
   Options FollowSymLinks
   AllowOverride All
 </directory>
   ### ostatni
   DocumentRoot /home/users/liberty/stage.cz/www/
   php_admin_value open_basedir
  "/home/users/liberty/:/tmp/:/usr/share/pear/"
   SSLEngine on
   SSLCertificateFile /etc/pki/tls/certs/localhost.crt
   SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
</VirtualHost>
```

Příloha D – Konfigurační soubor MySQL serveru

my.cnf – liberty1 (db1)

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Default to using old password format for compatibility with mysql 3.x
# clients (those using the mysqlclient10 compatibility package).
old_passwords=1
server-id=1
log_bin
replicate-do-db=tmobile_liberty
master-host=82.100.58.214
master-user=replication
master-password=tojeheslo
#logovani pomalych dotazu
log-slow-queries = /tmp/mysql-slow-query.log
long_query_time = 1
# pro lepsi vykon, staci interni lockovani mysql, lockovani souboru od OS
nepotrebujeme
skip-external-locking
#innodb_force_recovery = 4
max_connections = 1000
max_connect_errors = 10000
max_user_connections = 400
# kolik spojeni se smi ulozit, aby se zbytecne nenavazovaly znovu
thread_cache_size = 20
# slouzi pri obnove indexu pri pouziti ALTER, REPAIR, OPTIMIZE TABLE (muze byt
velky, pouzije se zridka a muze to dost zrychliti vytvareni indexu)
myisam_sort_buffer_size = 100M
# kolik spojeni se smi ulozit, aby se zbytecne nenavazovaly znovu
thread_cache_size = 20
#zvetsi vykon ORDER BY. Je alokovana per client, tak to radsi nepresvihnout :)
read_rnd_buffer_size = 10M
# zvetsi vykon pri dotazech nepouzivajicich indexy (per client)
read_buffer_size = 10M
# buffer po JOINY nevyuzivajici indexy
join_buffer_size = 1M
# buffer pro MYISAM indexy. Staci maly, na hurvinkovi je hlavne innoDB
key_buffer_size = 100M
# kolik tabulek se smi ukladat o cache, aby se nemusely nacitat z disku
table_cache = 300
# razeni vysledku, buffer per client. Je dobry pro ORDER a GROUP
sort_buffer_size = 30M
## INNODB
# hlavni cache pro INNODB. Na strojich vylucne s innoDB by melo byt 80-90% RAM
innodb_buffer_pool_size = 800M
# cache pro interni struktury INNODB. Staci 16M
innodb_additional_mem_pool_size = 16M
## TMP tabulky. Pouzije se mensi z techto dvou hodnot
# max velikost jakekoliv tmp tabulky v pameti
tmp_table_size = 700M
# max velikost explicitne vytvoreny tabulky v pameti
max_heap_table_size = 100M
# vyuziva se pro cachovani dotazu
query_cache_size = 100M
# Try number of CPU's*(2-4) for thread_concurrency
thread_concurrency = 24

[mysql.server]
user=mysql
basedir=/var/lib

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

Příloha E – Konfigurační soubory MySQL Master-Master replication manager

mmm_mon.conf

```
# Master-Master Manager config (monitor)

# Debug mode
debug no
# Paths
pid_path /usr/local/mmm/var/mmmd.pid
status_path /usr/local/mmm/var/mmmd.status
bin_path /usr/local/mmm/bin
# Logging setup
log mydebug
    file /usr/local/mmm/var/mmm-debug.log
    level debug
log mytraps
    file /usr/local/mmm/var/mmm-traps.log
    level trap
# MMMD command socket tcp-port
bind_port 9988
agent_port 9989
monitor_ip 82.100.58.254
# Cluster interface
cluster_interface eth0
# Cluster hosts addresses and access parami
# Tato sekce popisuje IP a nastaveni, které se používají pro monitorování
#Je nutne mit instanci MySQL poslouchající na těchto ip adresách a povolit
pristup monitorovacimu serveru do instance MySQL.
host db1
    ip 82.100.58.213
    port 3306
    user rep_monitor
    password RepMonitor
    mode master
    peer db2
host db2
    ip 82.100.58.214
    port 3306
    user rep_monitor
    password RepMonitor
    mode master
    peer db1
# Define roles
# Tato část definuje role ,a které IP pro ně používáme
active_master_role writer
# Mysql Reader role
role reader
    mode balanced
    servers db1, db2
    ip 82.100.58.233, 82.100.58.234
# Mysql Writer role
role writer
    mode exclusive
    servers db1, db2
    ip 82.100.58.235

# Checks parameters
# Ping checker
check ping
    check_period 1
    trap_period 5
    timeout 2
# Mysql checker
check mysql
    check_period 1
    trap_period  2
    timeout 2
# Mysql replication backlog checker
check rep_backlog
    check_period 5
```

```
    trap_period 10
    max_backlog 60
    timeout 2
# Mysql replication threads checker
check rep_threads
    check_period 1
    trap_period 5
    timeout 2
```

mmm_agent.conf – db1

```
# Master-Master Manager config (agent)
# Debug mode
debug no
# Paths
pid_path /opt/mmm/var/mmmd_agent.pid
bin_path /opt/mmm/bin
# Logging setup
log mydebug
    file /opt/mmm/var/mmm-debug.log
    level debug
log mytraps
    file /opt/mmm/var/mmm-traps.log
    level trap
# MMMD command socket tcp-port and ip
bind_port 9989
# Cluster interface
cluster_interface eth0
# Define current server id
this db1
mode master
# For masters
peer db2
# Cluster hosts addresses and access params
host db1
    ip 82.100.58.213
    port 3306
    user rep_monitor
    password RepMonitor
host db2
    ip 82.100.58.214
    port 3306
    user rep_monitor
    password RepMonitor
```

mmm_agent.conf – db2

```
# Master-Master Manager config (agent)
# Debug mode
debug no
# Paths
pid_path /opt/mmm/var/mmmd_agent.pid
bin_path /opt/mmm/bin
# Logging setup
log mydebug
    file /opt/mmm/var/mmm-debug.log
    level debug
log mytraps
    file /opt/mmm/var/mmm-traps.log
    level trap
# MMMD command socket tcp-port and ip
bind_port 9989
# Cluster interface
cluster_interface eth0
# Define current server id
this db2
mode master
# For masters
peer db1
# Cluster hosts addresses and access params
host db1
    ip 82.100.58.213
    port 3306
```

```
    user rep_monitor
    password RepMonitor
host db2
    ip 82.100.58.214
    port 3306
    user rep_monitor
    password RepMonitor
```

Příloha F – Skript zajišťující distribuci dat na jednotlivé datové servery

```
#######################################################################
# multithreaded ftp command distributor by michal babuska, miton cz
#                                     michal.babuska@miton.cz
#######################################################################
use strict;
use File::Basename qw(basename);
use Sys::Syslog qw(:DEFAULT setlogsock);
use Net::FTP;
use File::Basename;
use File::Spec::Unix;
use threads;
use Thread::Semaphore;
use POSIX qw(setsid);
#jednoducha konfigurace, prvni k prvnim druhej kdruhym atakdal....
# pocet serveru, pozor pokud das pocet serveru mensi nez kolik jich opravdu je
nebude to uplne fungovat...
my $server_count = 3;
# jejich adresy
my @server_address = ("88.86.107.35", "82.208.49.149", "82.100.58.155");
               # skubanek a rumcajs
#my @server_address = ("10.2.1.2");
# uzivatele a hesla...
my @server_user = ("cdn", "cdn", "cdn");
# pole hesel
my @server_pass = ("", "", "");
# root adresar pro ftp uloziste
my $ftp_root = '/home';
# kdyz se nam zachce debugovat...
my $debug_level = 0;
# promeny pro logovani
my $program = basename($0);
my $faillog = '/var/log/distftp_err';
my $fifo = '/var/log/proftpd-log.fifo';
my $syslog_facility = 'daemon';
my $syslog_level = 'info';
# budem delat vlakna....
my @ftpThreads;                                  # pole vlaken, v cecku sou to
ukazatele na vlakna, ale v perlu??
my $semaphore = Thread::Semaphore->new();        # semafor na ovladani
filedescriptoru, kterej pouzivam na logovani erroru
# vyrobime daumona
sub daemonize()
{
    chdir ("/")
        or die "chdir("/") failed: $!";
    open(STDIN, "/dev/null")
        or die "open() failed: $!";
    open (STDOUT, ">> /dev/null")
        or die "open() failed: $!";
    open (STDERR, ">> /dev/null")
        or die "open() failed $!";
    defined(my $pid = fork)
        or die "fork() failed: $!";
    exit if $pid;
    setsid
        or die "Can't start a new session: $!";
    umask 0;
}
# odchytavani signalu, no jeste nevim jaky odchytavat, ale urco to bude
potreba...
sub intHandler()
{
    close(FIFO);
    exit(0);
}
sub termHandler()
{
    close(FIFO);
```

```
    exit(0);
}
# prace s errorem...
sub myError($$$)              # $command, $arguments
{
    $semaphore->down;                          # protoze pouzivame tudle fci ve
vice vlaknach, musime ji uzamykat. kdybysme otevyrali vic stejnejch deskriptoru,
tak by se to rozbilo
    my $command = shift @_;
    my $path = shift @_;
    my $file = shift @_;

    my $parameters = join(' ', $path, $file);
    my $log_line = join(' ', $command, $parameters);
    open(FAILLOG, ">> $faillog");
    print FAILLOG $log_line;
    close FAILLOG;
    $semaphore->up;
}
# rutiny pro praci s ftp
sub ftpPut($$$$$$)
{
    my $host        = shift @_;
    my $ftp_user    = shift @_;
    my $ftp_pass    = shift @_;
    my $dir         = shift @_;
    my $localfile   = shift @_;
    my $filename    = shift @_;
    print "ftpPut(), connecting\n";
    my $ftp = Net::FTP->new($host, Debug => $debug_level);
    if (!$ftp) {
        myError("STOR ", $dir, $localfile);
        die "cannot connect: $@";
    }
    $ftp->login($ftp_user, $ftp_pass)
        or die "cannot login ", $ftp->message;
    $ftp->binary();
    print "cwd()\n";
    $ftp->cwd($dir);
    print "put()\n";
    $ftp->put($localfile, $filename);
    print "quit()\n";
    $ftp->quit;
}
sub ftpMkd($$$$$)
{
    my $host        = shift @_;
    my $ftp_user    = shift @_;
    my $ftp_pass    = shift @_;
    my $path        = shift @_;
    my $dir        = shift @_;

#    print "ftpMkd(), connecting\n";
    my $ftp = Net::FTP->new($host, Debug => $debug_level)
        or die "cannot connect to some.host.name: $@";
#    print "ftpMkd, loggingin $ftp_user $ftp_pass\n";
    $ftp->login($ftp_user, $ftp_pass)
        or die "cannot login ", $ftp->message;
    $ftp->cwd($path)
        or die "cwd(): $ftp->message";
#    print "ftpMkd(), mkdir\n";
    $ftp->mkdir($dir);
    $ftp->quit;
#    print "ftpMkd quit()\n";
}
sub ftpRmf($$$$$)
{
    my $host        = shift @_;
    my $ftp_user    = shift @_;
    my $ftp_pass    = shift @_;
    my $dir        = shift @_;
    my $file      = shift @_;

#    print "connecting\n";
```

```
    my $ftp = Net::FTP->new($host, Debug => $debug_level)
        or die "cannot connect to some.host.name: $@";
#    print "logging in\n";
    $ftp->login($ftp_user, $ftp_pass);
    $ftp->cwd($dir);
    $ftp->delete($file);
    $ftp->quit;
}
sub ftpRmd($$$$$)
{
    my $host        = shift @_;
    my $ftp_user    = shift @_;
    my $ftp_pass    = shift @_;
    my $dir         = shift @_;
    my $file        = shift @_;

#    print "connecting 85.207.120.155\n";
    my $ftp = Net::FTP->new($host, Debug => $debug_level)
        or die "cannot connect to some.host.name: $@";
#    print "logging in, $ftp_user, $ftp_pass\n";
    $ftp->login($ftp_user, $ftp_pass)
        or die "cannot login ", $ftp->message;
#    print "deleting\n";
    $ftp->cwd($dir);
    $ftp->rmdir($file);
#    print "$file deleted\n";
    $ftp->quit;
#    print "quit() ok\n";
}
sub ftpRename($$$$$)
{
    my $host        = shift @_;
    my $ftp_user    = shift @_;
    my $ftp_pass    = shift @_;
    my $from        = shift @_;
    my $to          = shift @_;

    my $ftp = Net::FTP->new($host, Debug => $debug_level)
        or die "cannot connect to some.host.name: $@";
    $ftp->login($ftp_user, $ftp_pass)
        or die "cannot login ", $ftp->message;
    $ftp->rename($from, $to);
    $ftp->quit;
}
#daemonize;
open(FIFO, "< $fifo") or die "$program: unable to open $fifo: $!\n";
while (1) {
    while (<FIFO>) {
        print "log: ".$_;
        if (m/STOR (.*)/) {                                  # ukladani souboru
            my $filename = basename($1);
            my $localfile = $1;
            $_ =~ m/$ftp_root(.*\/)?(.+)/ig;         #(.*)/ig;
            my $dir = $1;
            chomp($dir);
            chomp($filename);
            print $filename." ".$dir."\n";
            for (my $i = 0; $i <= ($server_count -1); ++$i) {    # vyrobime tolik
vlaken kolik bude potreba
                $ftpThreads[$i] = threads->new(\&ftpPut,
"$server_address[$i]","$server_user[$i]","$server_pass[$i]","$dir", "$localfile",
"$filename");
            }
            for (my $i = 0; $i <= ($server_count - 1); ++$i) {   # pockame az se
vlakna dokoncej a pak je zrusime
                $ftpThreads[$i]->join;
            }
        }
        elsif (m/DELE (.*)/) {
            $_ =~ m/$ftp_root(.*\/)?(.*)/ig;
            my $path = $1;
            my $file = basename($_);
            chomp($file);
            print $file." ".$path."\n";
```

```
            for (my $i = 0; $i <= ($server_count -1); ++$i) {
                $ftpThreads[$i] = threads->new(\&ftpRmf, "$server_address[$i]",
"$server_user[$i]", "$server_pass[$i]", "$path", "$file");
            }
            for (my $i = 0; $i <= ($server_count - 1); ++$i) {
                $ftpThreads[$i]->join;
            }
        }
        elsif (m/MKD (.*)/) {
            $_ =~ m/$ftp_root(.*\/)?(.*)/ig;
            my $path = $1;
            my $dir = basename($_);
            chomp($dir);
            print $dir." ".$path."\n";
            for (my $i = 0; $i <= ($server_count -1); ++$i) {
                $ftpThreads[$i] = threads->new(\&ftpMkd, "$server_address[$i]",
"$server_user[$i]", "$server_pass[$i]", "$path", "$dir");
            }
            for (my $i = 0; $i <= ($server_count - 1); ++$i) {
                $ftpThreads[$i]->join;
            }
        }
    }
    sleep (1);
}
close(FIFO);
exit 0;
```

Příloha G – Konfigurační soubor Lighttpd

Obsahuje jen relevantní položky. Kompletní konfigurační soubor je na přiloženém CD.

```
# lighttpd configuration file
############ Options you really have to take care of ####################
## modules to load
server.modules = (
                "mod_rewrite",
                "mod_alias",
                "mod_access",
                "mod_status",
                "mod_fastcgi",
                "mod_cgi",
                "mod_secdownload",
                "mod_accesslog" )
alias.url = ( "/perl/" => "/home/pub/updatestahuj/" )
$HTTP["url"] =~ "^/[^/]*/[^/]*/[^/]*/secured/" {
  url.access-deny = ("")
}
$HTTP["host"] == "stat3.miton.cz" {
                server.document-root = "/home/stat/"
}
$HTTP["host"] == "ftp3.stahuj.cz" {
        server.document-root = "/home/pub/http/"
        url.rewrite = ( "^.*/$" => "/decrypt.php" )
}
$HTTP["host"] == "amalka.miton.cz" {
        server.document-root = "/home/www/xmiton_cz/www/"
}
url.rewrite-once = ( "^/ir/" => "/ir/index.php")
server.document-root = "/home/"
server.max-worker = 8
server.event-handler = "linux-sysepoll"
server.stat-cache-engine = "fam"
server.max-fds = 8192
server.max-keep-alive-requests = 4
server.max-keep-alive-idle = 4
server.network-backend = "linux-sendfile"
## where to send error-messages to
server.errorlog = "/var/log/lighttpd/error_log"
# files to check for if .../ is requested
server.indexfiles = ( "index.php", "index.html",
                      "index.htm", "default.htm" )
# mimetype mapping
mimetype.assign = (
  ".rpm" => "application/x-rpm",
  ".pdf" => "application/pdf",
  ".sig" => "application/pgp-signature",
  ".spl" => "application/futuresplash",
  ".class" => "application/octet-stream",
  ".ps" => "application/postscript",
  ".torrent" => "application/x-bittorrent",
  ".dvi" => "application/x-dvi",
  ".gz" => "application/x-gzip",
  ".pac" => "application/x-ns-proxy-autoconfig",
  ".swf" => "application/x-shockwave-flash",
  ".tar.gz" => "application/x-tgz",
  ".tgz" => "application/x-tgz",
  ".tar" => "application/x-tar",
  ".zip" => "application/zip",
  ".mp3" => "audio/mpeg",
  ".m3u" => "audio/x-mpegurl",
  ".wma" => "audio/x-ms-wma",
  ".wax" => "audio/x-ms-wax",
  ".ogg" => "application/ogg",
  ".wav" => "audio/x-wav",
  ".gif" => "image/gif",
  ".jpg" => "image/jpeg",
  ".jpeg" => "image/jpeg",
```

```
  ".png" => "image/png",
  ".xbm" => "image/x-xbitmap",
  ".xpm" => "image/x-xpixmap",
  ".xwd" => "image/x-xwindowdump",
  ".css" => "text/css",
  ".html" => "text/html",
  ".htm" => "text/html",
  ".js" => "text/javascript",
  ".asc" => "text/plain",
  ".c" => "text/plain",
  ".cpp" => "text/plain",
  ".log" => "text/plain",
  ".conf" => "text/plain",
  ".text" => "text/plain",
  ".txt" => "text/plain",
  ".dtd" => "text/xml",
  ".xml" => "text/xml",
  ".mpeg" => "video/mpeg",
  ".mpg" => "video/mpeg",
  ".mov" => "video/quicktime",
  ".qt" => "video/quicktime",
  ".avi" => "video/x-msvideo",
  ".asf" => "video/x-ms-asf",
  ".asx" => "video/x-ms-asf",
  ".wmv" => "video/x-ms-wmv",
  ".bz2" => "application/x-bzip",
  ".tbz" => "application/x-bzip-compressed-tar",
  ".tar.bz2" => "application/x-bzip-compressed-tar"
 )
#### accesslog module
accesslog.filename = "/var/log/lighttpd/access_log"
## deny access the file-extensions
url.access-deny = ( "~", ".inc" )
$HTTP["url"] =~ "\.pdf$" {
  server.range-requests = "disable"
}
######### Options that are good to be but not neccesary to be changed #######
## bind to port (default: 80)
server.port = 80
## to help the rc.scripts
server.pid-file = "/var/run/lighttpd.pid"
###### virtual hosts
$HTTP["url"] =~ "^/stats" {
server.dir-listing = "enable"
}
### only root can use these options
## change uid to <uid> (default: don't care)
server.username = "miton"
## change uid to <uid> (default: don't care)
server.groupname = "miton"
#### fastcgi module
fastcgi.server = ( ".php" =>
                     ( "localhost" =>
                         (
                           "socket" => "/tmp/php-fastcgi.socket",
                           "bin-path" => "/usr/bin/php-cgi"
                          )
                      )
                  )
#### CGI module
cgi.assign = ( ".pl" => "/usr/bin/perl",
                        ".cgi" => "/usr/bin/perl" )
#### status module
status.status-url = "/server-status"
status.config-url = "/server-config"
status.statistics-url = "/server-statistics"
#### setenv
secdownload.secret = "natalka"
secdownload.document-root = "/home/"
secdownload.uri-prefix = "/dl/"
secdownload.timeout = 30
```