

Load Balancer

Softwarový projekt na MFF UK Praha

Uživatelská dokumentace

Poděkování:

Prof. RNDr. Milanu Tichému, DrSc. za propůjčení deseti počítačů v laboratoři labTS MFF UK pro účely testování.

Mgr. Janu Redlovi za umožnění praktického testu ve společnosti Czech OnLine.

Obsah

1	Úvod	6
1.1	Řešitelský kolektiv - rozdělení prací	6
1.2	Úvod do problematiky	6
1.3	Definice pojmů	7
1.4	Architektura Load Balanceru	7
2	Instalace a spuštění	10
2.1	Instalace	10
2.1.1	Instalace ODBC	11
2.2	Nastavení Load Balanceru	12
2.3	Nastavení konfiguračního souboru	12
2.3.1	Nastavení rozhraní Load Balanceru	13
2.3.2	Nastavení poslouchacích portů, serverů a skupin	15
2.3.3	Nastavení pravidel první vrstvy	16
2.3.4	Další základní nastavení	18
2.4	Spuštění	18
3	Proces init a ovládání Load Balanceru	19
3.1	Spuštění Load Balanceru	20
3.2	Běh Load Balanceru	22
3.3	Restart Load Balanceru	22
3.4	Ukončení Load Balanceru	23
4	Jádro	24
4.1	Architektura jádra	24
4.2	Servery	24
4.2.1	Tabulka portů	25
4.2.2	Tabulka serverů	25
4.2.3	Tabulka skupin	25
4.3	TCP a UDP Vlákná	26
4.4	Obsluha spojení	27

4.4.1	UDP spojení	27
4.4.2	TCP spojení	27
4.5	Výběr serveru	28
4.6	Navázání spojení	29
4.7	Komunikace se statistikami	30
4.8	Ping vlákno	31
5	Konfigurace	32
5.1	Úvod	32
5.2	Konfigurační soubor	32
5.2.1	Základní struktura	32
5.2.2	Princip modulů, sekcí a aliasů	33
5.2.3	Sekce MAIN	34
5.2.4	Sekce ALIASES	34
5.2.5	Uživatelské sekce	35
5.2.6	Jednoduché datové typy	35
5.2.7	Tabulky	37
5.2.8	Typ pro servery	38
5.2.9	Výčtové typy	39
5.2.10	Kontrola správnosti	40
5.2.11	Ukázkový konfigurační soubor	40
6	Webové rozhraní	43
6.1	Monitoring	43
6.2	Zobrazování statistik	44
6.2.1	Vytváření dotazu	44
6.2.2	Zobrazení výsledku dotazu	46
6.3	Konfigurace	47
6.3.1	Administrace	48
6.3.2	Náhled konfiguračního souboru	48
6.3.3	Main	48
6.3.4	Aliases	49
6.3.5	Sekce	50
7	Podmoduly	55
7.1	Knihovna první vrstvy	55
7.2	Knihovna druhé vrstvy	56
7.3	Knihovny třetí vrstvy	59
7.3.1	Statické algoritmy	60
7.3.2	Dynamické algoritmy	61
7.4	Knihovna na datové vrstvě	63

A Testy	64
A.1 Testy funkčnosti	64
A.2 Testy stability a zátěžové testy	66
A.3 Testy ve WAN	69
A.4 Praktický test	70

Kapitola 1

Úvod

Load Balancer je školní projekt, který má za cíl implementovat free softwarový modulární load balancer, který se bude svými vlastnostmi blížit a v některých aspektech i překonávat komerční load balancery. Důraz je kladen převážně na modularitu, stabilitu a výkon.

1.1 Řešitelský kolektiv - rozdělení prací

- Lukáš Hlůže – prezentační modul, webové rozhraní pro konfiguraci, dokumentace
- Václav Nidrle – statistický modul, balancovací algoritmy, dokumentace
- Přemysl Volf – jádro Load Balanceru, init modul, balancovací algoritmy, dokumentace
- Stanislav Živný – konfigurace, dokumentace

1.2 Úvod do problematiky

Běžnou technikou škálování systémů při jejich větším zatížení je výstavba serverových farem. Distribuce požadavků mezi nimi může být prováděna pomocí DNS round-robin, kdy jsou požadavky cyklicky rovnoměrně rozdělovány mezi jednotlivé servery ve farmě. Rozložení požadavků mezi servery ale pak není rovnoměrné vzhledem k jejich výkonu a není možné ho jemně ladit, natož pak specificky směřovat dotazy určitého typu na specializované servery.

Právě z výše uvedených důvodů se nasazují tak zvané load balancery, které na základě zadaných pravidel rozkládají zátěž mezi servery, které mohou být nesterjně vykonané a různě optimalizované (statické stránky, cgi

skripty, streaming servery). Load balancer se rozhoduje na základě administrátorem určené konfigurace a různých parametrů jako jsou dostupnost serverů, jejich aktuální vytížení, výkon za poslední dobu, typ požadavku, zdrojová a cílová adresa či port. Nasazení load balanceru tak přináší zrychlení vyřízení požadavků, snížení nároků a nákladů na hardware serverů a ochranu před výpadky jednotlivých serverů.

1.3 Definice pojmů

Modul Load Balanceru jsou procesy Load Balanceru, kde každý může běžet na různém počítači. Jedná se o init, jádro a statistiky. Jádro a statistiky tvoří dvojice procesu - řídicí a výkonný.

Výkonný proces je proces, který přímo provádí funkce daného modulu.

Řídicí proces je proces, který komunikuje s procesem init a stará se o běh výkonného procesu, ale sám nic neprovádí. V případě restartu Load Balanceru výkonný proces zabije a znovu spustí. Při zapnuté automatické kontrole zjišťuje, jestli výkonný proces odpovídá na signály a pokud neodpovídá, tak ho restartuje.

Vrstva je část jádra kterou postupně prochází spojení. První až třetí vrstvou procházejí nová spojení a rozhoduje se v nich, na který server se spojení přepojí. Další vrstvou je datová vrstva, kterou procházejí všechna data ve směru od klienta i od serveru. Reálné funkce, které vykonávají rozhodování na jednotlivých vrstvách se jmenují podmoduly.

Podmodul je funkce, která je uložena v dll knihovně. Podmoduly se volají při průchodu jednotlivými vrstvami. Druhá a třetí vrstva může postupně volat více podmodulů.

Statický algoritmus je balancovací algoritmus, který se řídí pouze údaji načtených z konfiguračního souboru a informacemi o konkrétním spojení.

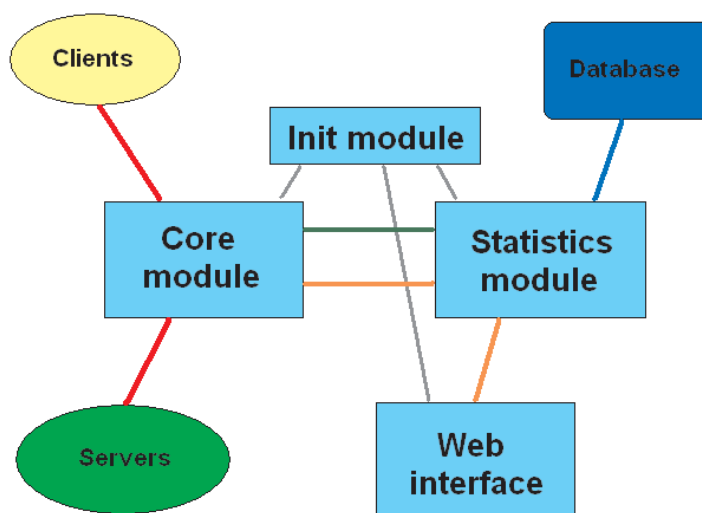
Dynamický algoritmus je balancovací algoritmus, který pro svoje rozhodování využívá dotazů do statistického modulu.

1.4 Architektura Load Balanceru

Load Balancer je rozdělen do několika modulů. Tím je umožněno dosáhnout různé úrovně funkčnosti a výkonu. Aby se co nejvíce omezila možnost, kdy

se Load Balancer stane "úzkým hrdlem", jsou jednotlivé moduly samostatné procesy, které mohou běžet na různých počítačích a komunikují spolu pomocí protokolu TCP/IP a UDP/IP.

Proces Init slouží ke spuštění, řízení a ukončení Load Balanceru. Jádro je výkonnou částí, která provádí samotné přepojování spojení. Statistika ukládají informace o provedených spojeních, odpovídají na dotazy jádra o stavu serverů a poskytují informace pro webové statistiky a informace. Webové rozhraní umožňuje vzdálenou správu Load Balanceru a prezentaci statistik.



Obrázek 1.1: Architektura Load Balanceru

Na obrázku je znázorněna architektura Load Balanceru a tok dat mezi jednotlivými částmi. Load balancer tvoří části označené světle modrou barvou. Tmavě modrou barvou je naznačena externí databáze, která je k Load Balanceru připojena přes ODBC rozhraní. Žlutou barvu mají klienti, kteří se připojují z Internetu na Load balancer a ten jejich požadavky přeměrovává na servery (zelené), které jsou na lokální síti.

Barevné spojnice na představují tok dat. Červeně jsou naznačena uživatelská data, která se přenesou k jádru a to je po výběru serveru přenesena na vybraný server. Tato zátěž je vysoká a spoje by proto měly být dostatečně dimenzovány.

Šedou barvou je naznačená servisní komunikace mezi modulem init a ostatními částmi Load Balanceru. Používá se pro přenos konfiguračních dat a restartování Load Balanceru.

Zelenou barvou jsou naznačena data odesílaná pomocí protokolu UDP z jádra do statistik o prováděných spojeních. Objem těchto dat se dá regulovat pomocí konfiguračního souboru.

Oranžová barva představuje dotazy jádra a webového rozhraní do statistik. Jádro dotazy používá pro balancování v dynamických algoritmech. Webové rozhraní pro zobrazování statistik o spojeních.

Modrá spojnice představuje práci statistického modulu s externí databází.

Celý Load Balancer by z důvodu výkonosti měl běžet na lokální síti. Jednotlivé části ale lze spustit i na WAN. Servisní komunikace (šedá barva) probíhá nezabezpečeně a proto se toto nastavení nedoporučuje.

Kapitola 2

Instalace a spuštění

2.1 Instalace

Instalace Load Balanceru ze zdrojového balíčku, který se nachází v adresáři package na CD.

```
balancer-0.1.tar.bz2
```

se provede následujícím způsobem. Pomocí

```
tar jxvf balancer-0.1.tar.bz2
```

balíček nejdříve rozbalíme. Pak se spustí konfigurační skript a následně kompilace pomocí příkazů:

```
./configure  
make  
make install
```

Poslední zmíněný příkaz nainstaluje Load Balancer do systému. Tím je na systému nainstalován Load Balancer.

Dalším krokem je instalace php extension balancer, která je potřeba k chodu webového rozhraní. Instalace je velice podobná. V rozbaleném balíčku se přejde do adresáře *present/php-ext/balancer* a php extension balancer se nainstaluje podobným způsobem jako Load Balancer.

```
./configure  
make  
make install
```

Dalším krokem je překopírování adresářové struktury *present/www* do kořenového adresáře webového serveru (dále jen *webroot*).

Dalším krokem je zkopírování skriptů `copy` a `restart` z adresáře *present/bin* do adresáře, ze kterého budou spustitelné z webového rozhraní a zároveň budou mít práva k manipulaci s konfiguračními soubory */etc/lb.cfg* a *webroot/config.cfg-default* a k posílání signálu modulu `init`. Tato situace se dá vyřešit nástrojem *sudo*.

2.1.1 Instalace ODBC

Pokud má být použita databáze, musí se nainstalovat ODBC rozhraní pro komunikaci mezi Load Balancerem a danou databází. Nejprve musí být nainstalován nějaký ODBC driver manager pro systém na bázi UNIXu (např. `unixODBC` - <http://www.unixodbc.org>), který se stará o správu ODBC driverů různých databází a jejich propojení s Load Balancerem.

Poté je ještě nutné nainstalovat ODBC driver pro danou databázi, která má být použita (např. `MySQL Connector/ODBC driver` pro databázi `MySQL` - <http://dev.mysql.com/downloads/connector/odbc/3.51.html>). Ten provádí překlad konkrétních požadavků do SQL pro příslušný databázový server.

Instalace probíhá klasickým způsobem ze zdrojových balíčků sekvencí příkazů

```
tar jxvf source.tar.bz2
./configure
make
make install
```

Pro úspěšnou instalaci je nutné mít alespoň následující verze knihoven a nástrojů :

glibc 2.3.4

flex 2.5.4

bison 1.875d

myodbc-3.51 3.51

unixODBC 2.2.6

php 4.3.10

gcc 3.3.5

gnu make 3.80

autoconf 2.59

2.2 Nastavení Load Balanceru

Pro správný běh webového rozhraní je nutné nastavit konfigurační soubor *webroot/config.php*. Je třeba nastavit tyto proměnné:

statsIP na IP adresu statistického modulu.

statsPORT na port, kde poslouchá statistický modul na dotazy od webového rozhraní. Standardně na portu 44793.

init_pid_file Jméno souboru, kam modul *init* ukládá číslo svého procesu. Webové rozhraní ho využívá pro restartování a ukončení Load Balanceru. Pokud je hodnota prázdná, tato funkce nebude zapnuta. Standardní je hodnota */var/run/lbinit.pid*

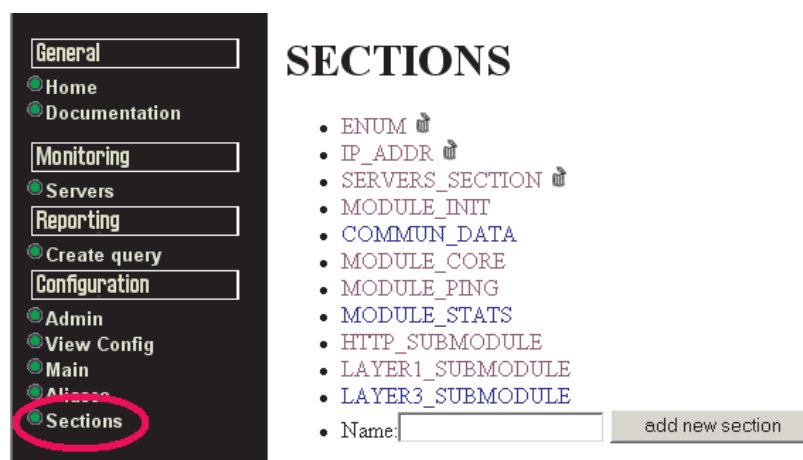
running_config na cestu ke konfiguračnímu souboru, který načítá modul *init*. Standardní je hodnota */etc/lb.cfg*.

bin_dir na cestu ke skriptům *copy* a *restart*.

2.3 Nastavení konfiguračního souboru

V tomto okamžiku je přístupná konfigurace pře webové rozhraní a je nastaven implicitní konfigurační soubor. Po instalaci se automaticky použije implicitní konfigurační soubor.

Pro nastavení jednoduché farmy serverů stačí změnit pouze několik základních nastavení. Jednotlivé sekce jsou přístupné z položky *Section* v části *Configuration*.



Obrázek 2.1: Stránka pro nastavení jednotlivých sekcí

2.3.1 Nastavení rozhraní Load Balanceru

Rozhraní Load Balanceru se nastavuje v sekci *IP_ADDR*. Pomocí této sekce se nastavují IP adresy jednotlivých součástí a rozhraní do vnější a vnitřní sítě. IP adresa se může zadat číselnou formou (např. 127.0.0.1) nebo jménem (např. localhost, www.mff.cuni.cz).

- IP_ADDR 🗑

The image shows four configuration panels for the Load Balancer interface, each with a title, a value field, a 'change' button, and a descriptive note.

- CORE_IP (varstring) 🗑**
Value: 192.168.0.1
(IP address of core module)
- STATS_IP (varstring) 🗑**
Value: 192.168.0.2
(IP address of statistics module)
- BALANCER_INNER_IP (varstring) 🗑**
Value: 192.168.0.1
(Inner IP address of Load Balancer connected to WAN)
- BALANCER_OUTER_IP (varstring) 🗑**
Value: www.mff.cuni.cz
(Outer IP address of Load Balancer connected to LAN)

Obrázek 2.2: Nastavení rozhraní Load Balanceru

Proměnná *CORE_IP* určuje IP adresu jádra. Na tuto adresu se připojuje modul *init*.

Proměnná *STATS_IP* určuje IP adresu statistik. Na tuto adresu se připojuje modul *init*.

Proměnná *BALANCER_INNER_IP* určuje rozhraní Load Balanceru do vnitřní sítě. Tato adresa se využije při překladu některých protokolů, u kterých se pro servery použije jako adresa klienta.


Proměnná *BALANCER_OUTER_IP* určuje rozhraní Load Balanceru do vnější sítě. Tato adresa se využije při překladu některých protokolů, u kterých se pro klienty použije jako adresa serveru.


U typického zapojení Load Balanceru je modul jádra na počítači se dvěma rozhraními. Jako IP adresa jádra a vnitřní IP adresa se použije adresa jed-

noho rozhraní. Jako vnější IP adresa se použije adresa druhého rozhraní. Jako adresa statistik se použije IP adresa z vnitřní sítě.

2.3.2 Nastavení poslouchacích portů, serverů a skupin


Seznamy portů a serverů se nastavují v sekci *SERVERS_SECTION*. Tato sekce obsahuje tři tabulky, které nastavují poslouchací porty, servery a skupiny serverů.


LISTEN_PORTS (tabledef) 

	IP protocol	IP port
	TCP <input type="text" value=""/>	80 <input type="text" value=""/>
	TCP <input type="text" value=""/>	<input type="text" value=""/>

Obrázek 2.3: Nastavení poslouchacích portů


V tabulce *LISTEN_PORTS* nastavují porty a protokoly, na kterých Load Balancer poslouchá ve vnější síti. Klienti se připojují na tyto porty.


SERVERS (tabledef) 

	Name	IP protocol	IP address	IP port	Static weight
	my server <input type="text" value=""/>	TCP <input type="text" value=""/>	192.168.0.3 <input type="text" value=""/>	80 <input type="text" value=""/>	100 <input type="text" value=""/>
	<input type="text" value=""/>	TCP <input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>

Obrázek 2.4: Nastavení serverů

V tabulce *SERVERS* je seznam serverů, které tvoří farmu. Pro každou službu musí být vytvořen nový záznam. Každý záznam tvoří uživatelský název server, jeho identifikace (protokol, IP adresa a port) a statická váha serveru. Ta určuje relativní poměr výkonnosti serverů. Pokud jsou všechny servery stejně výkonné, pak by měly mít nastavenou stejnou statickou váhu.

GROUPS (tabledef) 

	Name	List of groups divided by comma.
	<input type="text" value="all"/>	<input type="text" value="my server"/>
	<input type="text"/>	<input type="text"/>

Obrázek 2.5: Nastavení skupin serverů

V tabulce *GROUPS* je seznam serverů, které se používají v pravidlech na první až třetí vrstvě. Pokud jsou všechny servery ekvivalentní, stačí nastavit jednu skupinu obsahující všechny servery.

2.3.3 Nastavení pravidel první vrstvy

Pravidla, která využívá funkce *common* na první vrstvě, se nastavují v sekci *LAYER1_SUBMODULE*. Tato sekce obsahuje tabulku *L1_RULES*, která definuje jednotlivá pravidla.

L1_RULES (tabledef)

	Name	Send to stats	IP protocol	listen port	source port	source IP address	source mask	library on second layer	function on second layer	lib
	HTTP	YES	TCP	80	0					sta
	default TCP	YES	TCP	0	0					sta
	default UDP	YES	UDP	0	0					sta
		YES	TCP							

change

library on second layer	function on second layer	library on third layer	function on third layer	library on data layer	function on data layer	servers
		static_layer3.so	static_select_first			all
		static_layer3.so	static_select_first			all
		static_layer3.so	static_select_first			all
						all

Obrázek 2.6: Nastavení pravidel první vrstvy

Pro jednoduchá nastavení stačí nastavit tyto sloupce:

Name Název je součástí identifikace datového typu spojení ve statistikách. Tím je umožněno počítat statistiky pro jednotlivé typy spojení.

IP protocol Určuje, na který protokol se pravidlo vztahuje.

Listen port Určuje, na jakém portu bylo přijato spojení od klienta (např. 80 pro HTTP, 22 pro SSH, atd.)

Library on third layer Určuje název knihovny, ze které se použije funkce na třetí vrstvě. Pro jednoduché farmy vyhovuje knihovna static_layer3.so.

Function on third layer Určuje název funkce z výše zmíněné knihovny. Pro jednoduché farmy vyhovuje funkce static_select_first.

servers Určuje skupinu serverů, mezi které se rozděluje spojení. Pro jednoduché farmy se nastaví skupina všech serverů.

2.3.4 Další základní nastavení

Použití statistického modulu se nastavuje pomocí proměnné *INIT_STATS_START* v sekci *MODULE_INIT*.

2.4 Spuštění

Pokud je zapnutá funkce statistického modulu s databází, je nutné, aby běžel databázový server. Pokud běží, je možné spustit modul jádra a modul statistik. Po spuštění těchto modulů se spustí modul *init*, který spustí celý Load Balancer. Po inicializaci všech částí je Load Balancer připraven k použití a také je možné začít využívat webové rozhraní pro změny konfigurace, řízení Load Balanceru a zobrazování statistik o provedených spojeních.

Kapitola 3

Proces init a ovládání Load Balanceru

Veškeré řízení Load Balanceru uživatelem je po spuštění prováděno procesem Init. Nepřímo může být obsluhován přes webové prostředí. Nastavení procesu init a komunikace jednotlivých modulů se provádí v sekcích *<IP_ADDR>*, která je využívána všemi moduly a *<MODULE_INIT>*.

Seznam proměnných v sekci *<IP_ADDR>* :

CORE_IP - IP adresa jádra.

STATS_IP - IP adresa statistik.

BALANCER_INNER_IP - Vnitřní IP adresa Load Balanceru, která je využita při přepojování některých protokolů. Tato adresa je použita, pokud server vyžaduje adresu klienta.

BALANCER_OUTER_IP - Vnější IP adresa Load Balanceru, která je využita při přepojování některých protokolů. Tato adresa je použita, pokud klient vyžaduje adresu serveru.

Seznam proměnných v sekci *<MODULE_INIT>* (všechny hodnoty u parametrů *TIMEOUT* jsou v sekundách) :

INIT_STATS_START - Hodnota "YES" určuje, že mají být spuštěny statistiky, hodnota "NO" naopak.

INIT_CORE_PORT - Port, na který se připojí init k jádru, implicitně 44783.

INIT_STATS_PORT - Port, na který se připojí init ke statistikám, implicitně 44784.

INIT_CORE_STARTUP_TIMEOUT - Timeout určený pro spuštění a inicializaci jádra.

INIT_STATS_STARTUP_TIMEOUT - Timeout určený pro spuštění a inicializaci statistik.

INIT_INIT_CHECK_TIMEOUT - Timeout initu určený pro rozesílání a přijímání kontrolních zpráv. Pokud je nastaven na 0, nejsou kontroly prováděny.

INIT_CORE_CHECK_TIMEOUT - Timeout jádra určený pro rozesílání a přijímání kontrolních zpráv. Pokud je nastaven na 0, nejsou kontroly prováděny.

INIT_STATS_CHECK_TIMEOUT - Timeout statistik určený pro rozesílání a přijímání kontrolních zpráv. Pokud je nastaven na 0, nejsou kontroly prováděny.

INIT_CORE_FREEZE_CONTROL - Pokud je hodnota nastavena, provádí řídicí proces jádra kontrolu běhu výkonného procesu jádra.

INIT_STATS_FREEZE_CONTROL - Pokud je hodnota nastavena, provádí řídicí proces statistik kontrolu běhu výkonného procesu statistik.

INIT_SYSLOG_LEVEL - Nastavení úrovně logování do systémového logu ve všech modulech. Pokud není nastaveno, loguje se na úrovni *LOG_ERR*.

3.1 Spuštění Load Balanceru

Nejdříve se spouští jádro a statistiky (pokud je nastaveno jejich použití). Oba procesy se mohou spouštět s těmito přepínači:

-d Nespouštět jako službu daemon. Všechny procesy se implicitně spouští na pozadí jako služba daemon.

-p PORT Jádro (statistiky) bude poslouchat na připojení initu na portu PORT. Implicitně poslouchá na portu 44783 (44784).

-l IP_ADDR Jádro (statistiky) bude poslouchat na připojení initu z adresy IP_ADDR. Implicitně poslouchá na adrese INADDR_ANY.

Po spuštění a zpracování přepínačů čekají na připojení initu. Ten je možné spustit s těmito přepínači:

- d Nespouštět jako službu daemon. Init se implicitně spouští na pozadí jako služba daemon.
- f **FILE** Jako konfigurační soubor použít FILE. Implicitně se použije soubor /etc/lb.cfg
- o Provést pouze kontrolu konfiguračního souboru a nespouštět Load Balancer.
- a Při nastavení tohoto příznaku se hledají všechny chyby v konfiguračním souboru. Implicitně se kontrola souboru zastaví po nalezení první chyby
- e Vypisovat nalezené chyby na chybový výstup stderr. Implicitně se chyby zapisují pouze do syslogu.

Po spuštění init zpracuje přepínače, načte a zkontroluje konfigurační soubor a naváže komunikaci s jádrem a statistikami. Odešle jim konfigurační soubor a čeká na dokončení jejich inicializace. Během startovací sekvence se jádro a statistiky rozdělí na dva procesy. Rodičovský proces je řídicí proces a zajišťuje komunikaci s initem a případnou kontrolu synovského řídicího procesu.

Pokud konfigurační soubor není v pořádku, init nahlasí místo nalezené chyby a skončí. Výše popsanými přepínači lze dosáhnout výpisu chyby nejen do logu, ale i na stderr a také výpis všech nalezených chyb. Pokud se jedná o syntaktické chyby v konfiguračním souboru (chybějící středník, chybějící uvozovky, nesprávný počet parametrů v sekci *ALIASES* apod.), vždy se vypíše pouze první nalezená chyba, protože parser nemůže pokračovat v práci. V takovém případě se vypíše číslo řádky, kde byla chyba nalezena. Uživatel pak může chybu nalézt na dané řádce či na některé z předchozích řádek. Situace je vysvětlena příkladem:

```
30: varint "MAX" "100" "" ""
31:
32: # zalohovani
33:
34: varbool "zalohovat" "NO" "" "";
```

Na řádce číslo 30 chybí středník. Parser přeskakuje řádky 31 a 33 (protože jsou prázdné) a řádku 32 (protože se jedná pouze o komentář) a na řádce 34 nachází klíčové slovo (varbool) namísto středníku. Je nahlášena chyba na řádce 34. Uživatel zkontroluje řádku 34, která je v pořádku. Kontroluje řádky směrem k začátku konfiguračního souboru, ignoruje prázdné a zakomentované řádky, a nachází chybu na řádce 30.

Chybějící středník v sekci *MAIN* či *ALIASES* se nahlásí jako chyba o špatném či neexistujícím přiřazení.

```
<MAIN>
core coremain coreext2
stat statisticssection;
</MAIN>
```

V tomto případě chybí na první řádce sekce *MAIN* ukončovací středník a další slovo, které parser načte, je *stat* a parser ho považuje za další uživatelskou sekci, která je přiřazena modulu *core*. Pokud taková sekce neexistuje, je nahlášena chyba. Stejně chování má i chybějící středník v sekci *ALIASES*.

Druhá část kontroly konfiguračního souboru hlídá např. správný počet parametrů v uživatelských sekcích, existenci sekcí, které se vyskytují v definicích aliasů apod. Pro tuto část testu se dá parametry modulu *init* určit, zda se vypíše jen první nalezená, či všechny chyby.

3.2 Běh Load Balanceru

Pokud jsou nastaveny příslušné parametry, kontroluje *init* běh jádra a statistik. Pokud s nimi ztratí kontakt, čeká na jejich opětovné spuštění nebo obnovení spojení a podle stavu Load Balanceru nastaví všechny části do normálního stavu. Pokud je nastavena kontrola výkonných procesů, kontrolují je řídicí procesy. Pokud výkonný proces neodpoví řídicímu, je považován za nefunkční. Řídicí proces ukončí jeho běh, znovu spustí a provede jeho inicializaci. Za běhu by se neměly posílat jakékoliv signály jinému procesu než procesu *init*.

3.3 Restart Load Balanceru

V případě změny konfiguračního souboru je nutné celý Load Balancer restartovat. To se provede zasláním signálu *SIGHUP* procesu *init* nebo pomocí příslušného tlačítka ve webovém rozhraní. Během restartu jsou ukončeny výkonné procesy jádra a statistik a znovu spuštěny s novým konfiguračním souborem. Během restartování Load Balancer nepracuje a dojde k výpadkům současných spojení a nepřijímání nových spojení po dobu restartu.

3.4 Ukončení Load Balanceru

Load Balancer se ukončuje zasláním signálu SIGTERM procesu init nebo pomocí příslušného tlačítka ve webovém rozhraní. Při ukončování zašle init jádru a statistikám zprávu o ukončení a sám ukončí svoji činnost. Řídící procesy ukončí výkonné procesy a pak sami ukončí svoji činnost.

Kapitola 4

Jádro

Jádro je výkonnou částí Load Balanceru, které přepojuje spojení. Po přijetí spojení ho předá buď příslušnému UDP vláknu nebo vybere jedno z volných TCP vláken, které nový požadavek obslouží.

4.1 Architektura jádra

U jádra je kladem důraz především na modularitu, škálovatelnost, stabilitu a výkon. Je postaveno, aby mohlo vykonávat alespoň základní činnosti bez spuštění některých dalších součástí Load Balanceru, případně po jejich pádu.

Při samotném přepojování dat od klientů k serverům je kladen důraz především na modularitu a výkon. Pro výběr správného serveru se používají dll knihovny na třech vrstvách, které se mohou nahrávat i přidávat za běhu Load Balanceru (nová knihovna vyžaduje úpravu konfiguračního souboru a proto následné restartování Load Balanceru)

Výkonnost jádra lze nastavovat automatickým řízením počtu obsluhujících TCP vláken na základě konfiguračního souboru. Je tak možnost kompromisu mezi výkonem a množstvím systémových prostředků na základě aktuálního vytížení Load Balanceru.

4.2 Servery

Tabulka serverů je uvedena v sekci `<SERVERS_SECTION>` v konfiguračním souboru. Tato sekce obsahuje tabulky `LISTEN_PORTS`, `SERVERS` a `GROUPS`. Názvy serverů a skupin se zobrazí jako rozbalovací nabídka všude, kde je použitý typ `varservers`.

4.2.1 Tabulka portů

Tabulka *LISTEN_PORTS* obsahuje seznam portů, které jádro otevře pro poslouchání. Na tyto porty se připojují klienti z vnější sítě. Výjimku tvoří porty otevřené jako pomocná datová spojení v podmodulu na datové vrstvě. Tabulku tvoří dva sloupce:

protokol Řetězec nabývajících hodnoty TCP nebo UDP. Určuje protokol příchozího spojení.

port Číslo určující otevíraný port.

4.2.2 Tabulka serverů

Tabulka *SERVERS* obsahuje seznam serverů, mezi které se rozdělují příchozí spojení. Každá služba na serveru musí mít samostatný záznam v tabulce serverů. Tabulku tvoří pět sloupců:

název Název serveru, který se bude objevovat v nabídce názvů u proměnných typu *varservers*.

protokol Protokol služby daného serveru. Řetězec nabývajících hodnoty TCP nebo UDP.

IP adresa Řetězec obsahující IP adresu serveru.

port Číslo určující port serveru (služby).

statická váha Číslo určující statickou váhu služby. Ve skupině serverů vybraných pro dané spojení bude poměr přidělených spojení odpovídat poměrům statických vah (pokud je zvolen příslušný algoritmus).

4.2.3 Tabulka skupin

Tabulka *GROUPS* obsahuje seznam skupin. Každá skupina obsahuje seznam serverů z tabulky *SERVERS*. Jednotlivé servery se mohou objevit ve více skupinách. Tabulku tvoří dva sloupce:

název Název skupiny, který se bude objevovat v nabídce názvů u proměnných typu *varservers*.

seznam serverů Datový typ *varlist*, který obsahuje seznam názvů serverů z tabulky *SERVERS*.

4.3 TCP a UDP Vlákna

Pro zvýšení výkonu jádro pracuje ve více vláknech současně. Ke každému otevřenému portu protokolu UDP se vytvoří jedno vlákno, které spojení na tomto portu obsluhuje. Počet TCP vláken se řídí nastavením proměnných v konfiguračním souboru. Jejich počet se může za běhu zvyšovat nebo snižovat podle aktuálního zatížení jádra.

Při startu jádra se vytvoří *CORE_THREADS_INIT* TCP vláken. Při dostatečném vytížení jádra se jejich počet může zvyšovat až na *CORE_THREADS_MAX*. Při nevytíženém jádru se může počet vláken snížit na *CORE_THREADS_MIN*. Pokud je minimální počet vláken menší než počáteční, počet vláken by se po startu jádra (než provoz dosáhne běžné úrovně) začal snižovat až na minimální hodnotu. Proto je první snížení počtu vláken možné až po prvním zvýšení počtu vláken. Pak již počet vláken může nabývat neomezeně od minima po maximum.

Jádro po přidělení nového spojení kontroluje poměr počtu existujících vláken ku počtu aktuálně pracujících vláken. Pokud tento poměr přesáhne hodnotu *CORE_THREADS_FULL*, pokusí se jádro zvětšit počet vláken. Vynásobí aktuální počet existujících vláken koeficientem *CORE_THREADS_RATE_UP*, který by měl být větší než jedna. Pokud nový počet přesáhne hodnotu *CORE_THREADS_MAX*, je na tuto hodnotu snížen.

Pokud poměr klesne pod hodnotu *CORE_THREADS_EMPTY*, pokusí se jádro zmenšit počet vláken. Vydělí aktuální počet existujících vláken koeficientem *CORE_THREADS_RATE_DOWN*, který by měl být větší než jedna. Zároveň by měla platit nerovnost $1/CORE_THREADS_RATE_DOWN \geq CORE_THREADS_EMPTY$. Pokud je nový počet menší než hodnota *CORE_THREADS_MIN*, je na tuto hodnotu zvýšen.

Proměnné *CORE_THREADS_FULL* a *CORE_THREADS_EMPTY* by měly být v intervalu $\langle 0, 1 \rangle$.

Seznam všech proměnných ovlivňující počet TCP vláken:

- CORE_THREADS_INIT*** Výchozí počet TCP vláken při spuštění jádra.
- CORE_THREADS_MIN*** Minimální počet TCP vláken.
- CORE_THREADS_MAX*** Maximální počet TCP vláken.
- CORE_THREADS_FULL*** Poměr existujících a pracujících vláken, při kterém se jádro pokusí zvětšit počet TCP vláken.
- CORE_THREADS_RATE_UP*** Hodnota, kterou se vynásobí počet vláken při zvyšování jejich počtu.

CORE_THREADS_EMPTY Poměr existujících a pracujících vláken, při kterém se jádro pokusí zmenšit počet TCP vláken.

CORE_THREADS_RATE_DOWN Hodnota, kterou se vydělí počet vláken při snižování jejich počtu.

4.4 Obsluha spojení

4.4.1 UDP spojení

Protože UDP komunikace není spojovaná, ale probíhá po jednotlivých packetech, dá se u ní rozlišit začátek, ale nedá se rozpoznat konec komunikace (uzavření spojení u TCP spojení). Proto je po vytvoření socketu pro komunikaci určitého klienta s určitým serverem nutné zachovat socket otevřený pro případnou další komunikaci. Tím vzniká problém postupného růstu otevřených socketů. Nelze totiž rozlišit, jestli bude po výměně jednoho packetu spojení dále nevyužíváno nebo po jednom packetu od klientat bude server jednou za den posílat nějaká data.

Tato situace je řešena pomocí pole o délce *CORE_UDP_CONNECTIONS*, které obsahuje záznam o otevřených spojeních. Nejdříve se postupně zaplňuje toto pole a po jeho naplnění se vezme nejstarší spojení (s nejstarším datem posledního packetu), uzavře se a otevře se pro nově příchozí spojení.

Po přijetí spojení se pomocí projití třech vrstev algoritmů vybere příslušný server. K vybranému serveru se otevře socket a přepošlou se přijatá data.

4.4.2 TCP spojení

Na nové požadavky čeká jádro v hlavním vláknu. Na všech otevřených portech je zavolána funkce listen s parametrem *CORE_LISTEN_CLIENTS*, který určuje délku fronty kompletně navázaných spojení, které čekají na přijetí. Po výskytu nového spojení je probuzeno jedno TCP vlákno k obslužení nového spojení.

Po probuzení TCP vlákna hlavním vláknem se provede minimum operací, aby hlavní vlákno mohlo co nejrychleji pokračovat v činnosti.

Po přijetí spojení se pomocí projití třech vrstev algoritmů vybere příslušný cílový server. Pokud není vybrán nebo není dostupný žádný server, je spojení uzavřeno a vlákno se připraví do stavu na přijetí dalšího požadavku.

V tomto okamžiku je již navázáno spojení s vybraným serverem. Pokud je vyžadováno použití datového podmodulu, zavolá se s parametrem o zahájení spojení. Pak se na server odešlou data, která si načetly podmoduly na první

až třetí vrstvě. Pokud je vyžadováno použití datového podmodulu, zavolá se před odesláním těchto dat, aby je mohl případně modifikovat.

4.5 Výběr serveru

Výběr serveru probíhá stejným mechanismem pro TCP i UDP spojení. Vlákno postupně volá funkce podmodulů na třech vrstvách. Funkcím jsou předány všechny potřebné parametry a jejich výstupem je seznam serverů s prioritami, které rozhodují o pořadí serverů při přepojování spojení.

Podmodul a jeho funkce na první vrstvě se určuje v konfigurační části jádra a volá se pro každé spojení. Knihovna, která obsahuje funkci pro první vrstvu, je v proměnné *CORE_LAYER1_LIBRARY* a funkce je v proměnné *CORE_LAYER1_FUNCTION*. Samotná funkce se dá plně konfigurovat, ale v případě speciálních požadavků lze napsat a vložit svoji vlastní. Funkce na první vrstvě je povinná. Rozhoduje se na základě portu, na kterém Load Balancer poslouchá, protokolu, adresy klienta a dalších parametrů. Výstupem této funkce je skupina serverů, které mohou daný požadavek obsloužit, knihovna a funkce druhé nebo třetí vrstvy, případně datového podmodulu a data načtená pro vykonání rozhodnutí.

Podmodul a jeho funkce na druhé vrstvě je volána na základě výsledku funkce na první vrstvě. Typicky je pro každý port/protokol napsán zvláštní podmodul (často nemusí být žádný). Druhá vrstva nadále upřesňuje skupinu serverů, které jsou schopny požadavek řešit a které dostala od podmodulu na první vrstvě. Výstupem je knihovna a funkce pro třetí vrstvu, případně datový podmodul a data načtená pro vykonání rozhodnutí. Typicky se podmoduly na druhé vrstvě rozhodují podle obsahu dat (*Layer 7 switching*). Funkce na druhé vrstvě se mohou volat opakovaně a tak zjemňovat dělení. Naimplementovaný je podmodul pro rozhodování na základě obsahu dat v protokolu HTTP.

Po průchodu prvních dvou vrstev by skupina vybraných serverů měla být ekvivalentní z hlediska schopnosti vyřízení požadavku daného typu spojení. Ve funkcích v podmodulech na třetí vrstvě se určuje pořadí vybrané skupiny serverů buď na základě statických hodnot (statická váha serveru, zdrojová adresa, ...) nebo pomocí dotazování se statistik. Jádro pak postupně zkouší vytvořit spojení na těchto serverech podle jejich priority. Funkce na třetí vrstvě by neměla být volána pouze v případě, kdy daný požadavek může vyřídit pouze jeden server nebo pokud je prioritou serverů nastavená z předchozích vrstev.

Po projití všemi funkcemi jednotlivých vrstev začne vlákno vybírat cílový server podle seznamu serverů, které vrátí funkce podmodulů. Seznam ob-

sahuje položku priority, která rozhoduje o pořadí serverů. Seznam se prochází od začátku a vybere se první server, který má nejvyšší prioritu a je dostupný. Na vybraný server se vlákno pokusí připojit. Pokud se to nepodaří, pokračuje se v procházení seznamu a vezme se další server s nejvyšší prioritou. Pokud už žádný takový není, prochází se seznam znovu od začátku a vybírá se server s druhou nejvyšší prioritou atd., dokud je jejich priorita větší než nula. U všech serverů, ke kterým se nepodařilo připojit, se zvýší počet neúspěšných pokusů a při přesažení hodnoty *PING_TTD* je označen za nefunkční server.

Dalším podmodulem, který lze využít, je datový podmodul. Tento podmodul lze využít pouze pro TCP spojení. Slouží k řízení nových spojení a k úpravě přenášených dat. Datový modul se nahraje a používá, pokud je nastaven při průchodu první až třetí vrstvou. Volá se při vytvoření hlavního spojení, při ukončení hlavního spojení a vždy při přenosu dat oběma směry v hlavním spojení. V tom případě se nejdříve data načtou, pak se zavolá funkce datového podmodulu a následně se data zapíše. Datový podmodul má možnost měnit data před jejich zapsáním a vytvářet, otevírat a zavírat pomocná datová spojení potřebná pro příslušný protokol směrem od i k serveru. Po ukončení hlavního spojení jsou všechna datová spojení také ukončena. Naimplementovaný je datový podmodul pro protokol FTP, který umožňuje přepojování aktivního i pasivního přenosu.

Všechny podmoduly se nahrávají jako dll knihovny a pro další rozšíření o nové protokoly tak stačí upravit nebo připsat nové dll knihovny. Load Balancer pak stačí restartovat pro uplatnění změn. Tím je dosaženo vysoké modularity a možnosti dalšího rozšíření i o zcela speciální nestandardní protokoly.

4.6 Navázání spojení

Navazování spojení se provádí voláním funkce `connect()`. Ta má v systému nastavena příliš dlouhý timeout, který by při vysokém zatížení mohl zablokovat celý Load Balancer. Proto je k dispozici několik nastavení, která mohou implicitní timeout zkrátit. K nastavení se používají tyto proměnné:

CORE_CONNECT_TIMEOUT Určuje délku timeoutu v milisekundách. Pokud je nastaveno na nulu, použije se autokonfigurace.

CORE_CONNECT_SEND_TO_STATS Pokud je nastaveno, odesílají se informace o úspěchu či neúspěchu do statistik.

CORE_CONNECT_STARTING_TIME Určuje počáteční délku timeoutu při autokonfiguraci.

CORE_CONNECT_NUMBER Určuje počet spojení, které reprezentuje průměrná hodnota.

CORE_CONNECT_RATE Určuje, jakou hodnotou se má vynásobit průměrná hodnota pro nastavení délky timeoutu.

Connect timeout se nastavuje proměnnou *CORE_CONNECT_TIMEOUT*, která určuje délku timeoutu v milisekundách. Pokud je nastavena na nulovou hodnotu, použije pro stanovení délky timeoutu autokonfigurace.

Při autokonfiguraci si jádro počítá průměrnou hodnotu trvání connectu (spojení se serverem). Průměr se počítá z *CORE_CONNECT_NUMBER* hodnot, kde při výpočtu nové hodnoty průměru se použije aktuální hodnota trvání a *CORE_CONNECT_NUMBER-1* hodnot předchozího průměru. Při startu se jako počáteční průměrná hodnota použije *CORE_CONNECT_STARTING_TIME* zadaná v milisekundách. Pro nastavení timeoutu pro spojení se spočtený průměr vynásobí hodnotou proměnné *CORE_CONNECT_RATE*. Pro shrnutí se nová průměrná hodnota spočítá takto:
$$\frac{average_connect_time * (CORE_CONNECT_NUMBER - 1) + new_connect_time}{CORE_CONNECT_NUMBER}$$
 kde *average_connect_time* je průměrný čas předchozích spojení a *new_connect_time* je connect čas aktuálního spojení.

4.7 Komunikace se statistikami

Pokud je nastaveno spuštění statistik a je nastavený příslušný parametr po průchodu podmodulů pro vybírání serveru, jsou statistikám odeslána data o spojení. Tyto data se používají pro dotazování se podmodulů na aktuální stav serverů (jejich zatížení, odezvy, ...) a pro výpočet statistik za různá časová období a jejich prezentace na webovém rozhraní.

Pro UDP spojení se odesílá jedna informace při přeposlání packetu na server a jedna informace při opačném směru.

Pro TCP spojení jsou informace odesílány v těchto okamžicích:

- Při otevření nového spojení.
- Při přenosu dat od klienta k serveru.
- Při přenosu dat od serveru ke klientovi.
- Při uzavření spojení ze strany klienta.
- Při uzavření spojení ze strany serveru.

- Při přenosu dat od klienta k serveru pomocným datovým spojením.
- Při přenosu dat od serveru ke klientovi pomocným datovým spojením.

Pro odesílání informací od jádra ke statistikám se využívá UDP spojení na portu *COMMUN_STATS_PORT*. Pro dotazování podmodulů do statistik se využívá TCP spojení na portu *COMMUN_CORE3RD_PORT*. Statistický modul může současně zpracovávat maximálně *COMMUN_CLIENTS3RD_MAX* spojení.

4.8 Ping vlákno

Pokud je v konfiguračním souboru nastavena proměnná *PING_ENABLE* je při inicializaci vytvořeno a spuštěno ping vlákno. Vlákno postupně prochází seznam serverů a provede test dostupnosti portu všech serverů (služeb). Pokud dojde ke změně jeho stavu, je odeslána zpráva statistikám (pokud jsou spuštěny) a příslušně změněny proměnné v jádru, aby daný server byl / nebyl využíván. Ping vlákno provádí testování každých *PING_DELAY* milisekund.

Součástí Load Balanceru je testování těchto služeb: SMTP, HTTP, MYSQL, FTP, POP a IMAP. Další služby je možné přidat jednoduchým upravením knihovny *ping.c* popsané v programátorské části návodu.

Kapitola 5

Konfigurace

5.1 Úvod

Jednotlivé moduly Load Balanceru mohou běžet na různých počítačích, konfigurační soubor je ale pouze na jednom počítači a to na počítači, na kterém běží modul Init. Standardní jméno konfiguračního souboru je *lb.cfg* a standardní umístění je v adresáři */etc*. Modulu Init může být při startu zadán jiný než standardní konfigurační soubor. Modul Init také umožňuje kontrolu syntaktické správnosti zadaného (či standardního) konfiguračního souboru.

Jednotlivé moduly Load Balanceru si mezi sebou obsah konfiguračního souboru předají.

Konfigurace Load Balanceru může být prováděna přímo změnou konfiguračního souboru a posléze restartu modulu Init (zasláním signálu) nebo přes webové rozhraní.

5.2 Konfigurační soubor

5.2.1 Základní struktura

Konfigurační soubor se skládá (v daném pořadí) z povinné sekce *MAIN*, povinné sekce *ALIASES* a nepovinných uživatelských sekcí. Začátek sekce *SEC* je uveden řádkem

```
<SEC>
```

a ukončena řádkem

```
</SEC>
```

,kde za *SEC* můžeme substituovat *MAIN*, *ALIASES* nebo libovolný identifikátor uživatelské sekce. Přípustný identifikátor uživatelské sekce je posloupnost znaků začínající podtržítkem nebo písmenem anglické abecedy a pokračující libovolným písmenem anglické abecedy, cifrou či podtržítkem.

Každá řádka v libovolné sekci (tj. sekci *MAIN*, *ALIASES* či uživatelské sekci) je ukončena povinným středníkem.

Konfigurační soubor je tzv. *case-sensitive*, tedy rozlišuje malá a velká písmena (jak je v Unix-ovém světě zvykem).

V konfiguračním souboru se mohou vyskytovat i řádky s komentářem uvozené znakem mřížky (*#*), ale ty jsou při změně konfiguračního souboru přes webové rozhraní odstraněny. Pro komentář jednotlivých proměnných konfiguračního souboru slouží k tomu určený, dále popsany, mechanismus.

Celková koncepce konfiguračního souboru byla navržena s cílem snadného rozšíření o nové balancovací moduly, které mohou snadným a jednoduchým způsobem získat data z konfiguračního souboru.

5.2.2 Princip modulů, sekcí a aliasů

Konfigurační soubor slouží k uchování dat. Používají ho moduly samotného Load Balanceru i moduly, jež se starají o distribuci příchozích požadavků na server mezi servery ve farmě. Každý takový modul zná své jméno a může si do konfiguračního souboru uložit své proměnné.

Uživatelské sekce slouží k pojmenování jedné oblasti uchováující více proměnných. Byly vytvořeny proto, aby stejný modul mohl mít několik alternativních nastavení svých interních proměnných.

Sekce *MAIN* umožňuje přiřadit modulům uživatelské sekce. Pokud pak daný modul zažádá o data z konfiguračního souboru, přičemž se identifikuje svým jménem, dostane hodnoty proměnných z uživatelských sekcí přiřazených modulu téhož jména.

Uživatelské sekce mohou být mezi moduly sdílené, tedy jedna uživatelská sekce může být přiřazena obecně více modulům.

Pro větší pohodlí uživatele byl vytvořen koncept aliasů. Alias si lze představit jako "několik řádek v sekci *MAIN*". Alias je tedy zkratka za obecně několik přiřazení, kde každé jednotlivé přiřazení přiřazuje modulu uživatelské sekce. Hlavní důvod pro vytvoření aliasů byla možnost rychlého a snadného přepínání mezi různými balancovacími algoritmy (viz. ukázkový příklad konfiguračního souboru dále) - stačí změnit příslušný alias v sekci *MAIN*.

Sekce *ALIASES* slouží pro definici aliasů, které se mohou používat v sekci *MAIN*.

5.2.3 Sekce MAIN

Sekce *MAIN*, jak již bylo řečeno, slouží pro přiřazení uživatelských sekcí danému modulu. Toho lze dosáhnout přímo nebo použitím aliasů. Jednotlivé řádky sekce *MAIN*, ukončené (jako všechny řádky všech sekcí) povinným středníkem, mohou být dvojího typu. Buď je to řádka obsahující jediné slovo a středník. V tom případě se jedná o název aliasu, který se má použít. A výsledný efekt je stejný, jako kdyby všechny řádky ze sekce *ALIASES*, jež mají první slovo na řádce právě daný alias, byly vloženy do sekce *MAIN*.

Druhá možnost je, že řádka v sekci *MAIN* obsahuje alespoň dvě slova a v tomto případě je to interpretováno tak, že modulu (první slovo na řádce) jsou přiřazeny uživatelské sekce (zbylá slova na řádce).

Sekce *MAIN* je povinná, i když může být prázdná. Bez ní bude konfigurační soubor prohlášen za syntakticky nesprávný.

Uvedení stejného identifikátoru uživatelské sekce vícekrát v řádce přiřazující uživatelské sekce danému modulu nemá smysl. Pokud jsou danému modulu přiřazeny uživatelské sekce na několika řádcích sekce *MAIN*, je výsledně modulu přiřazeno sjednocení těchto uživatelských sekcí.

Obdobně vícenásobné použití jednoho a téhož aliasu v sekci *MAIN*.

Ukázkový příklad sekce *MAIN*:

```
<MAIN>
  core core_base core_ext SERVERS_SECTION;
  stat statistics SERVERS_SECTION;
  round_robin1;
  prezent2;
</MAIN>
```

5.2.4 Sekce ALIASES

Sekce *ALIASES* je stejně jako sekce *MAIN* povinná, i když může být také prázdná. Bez ní je konfigurační soubor prohlášen za syntakticky nesprávný.

Sekce *ALIASES* slouží k definici aliasů, které se mohou používat v sekci *MAIN*.

Každá řádka sekce *ALIASES* musí obsahovat alespoň tři slova. První je jméno aliasu, druhé jméno modulu, kterému chceme přiřazovat a třetí (a případně další) slova jsou jména uživatelských sekcí, které danému modulu přiřazujeme.

Obdobně jako v sekci *MAIN*, nemá smysl v jedné řádce sekce *ALIASES* uvádět stejné identifikátory uživatelských sekcí vícekrát a ani na různých řádkách přiřazujících stejnému modulu v rámci téhož aliasu.

Aliases, které se vyskytly v sekci *MAIN* ale nebyly definovány v sekci *ALIASES* jsou ignorovány.

Ukázkový příklad sekce *ALIASES*

```
<ALIASES>
round_robin1 mod1basic mod1A mod1C;
round_robin1 mod2php PHP;
round_robin2 mod1basic mod1B mod1C;
round_robin2 mod2php PHPnew;
prezent1 prezent prezent1 prezent_old_style;
prezent2 preeznt prezent1 prezent_new_style;
</ALIASES>
```

5.2.5 Uživatelské sekce

Uživatelské sekce jsou pojmenováním oblasti, která obsahuje obecně více proměnných. Slouží k tomu, aby moduly mohly mít několik alternativních konfigurací, jež se liší třeba jen v nepatrných změnách hodnot některých proměnných a šlo mezi nimi jednoduše přepínat. Dále mohou posloužit ke sdílení sekcí mezi moduly. Příklady uživatelských sekcí jsou dále v příkladu kompletního konfiguračního souboru. Nejprve bude vysvětleno, jak lze v uživatelských sekcích definovat proměnné různých datových typů.

Jak bude popsáno dále, kontroluje se sice syntaktická správnost konfiguračního souboru (správný počet argumentů, neprázdnost identifikátorů atd.), ale pokud se vyskytnou dvě sekce se stejným jménem, jsou jednoduše spojeny do jedné. Pokud uživatel ví, co dělá, není to problém. Pokud ale obsahy nejsou disjunktní (tj. například se vyskytne dvojí definice téže proměnné), bude modulu předána hodnota proměnné jež byla v konfiguračním souboru dříve uvedena a obecně není chování jednoznačné.

Všechny řádky uživatelských sekcí mají stejnou strukturu: první je klíčové slovo a pak je několik hodnot v uvozovkách, vše ukončené povinným středníkem. Počet slov za klíčovým slovem závisí na klíčovém slově samotném.

5.2.6 Jednoduché datové typy

Následuje seznam tzv. jednoduchých datových typů spolu s příslušným klíčovým slovem pro deklaraci proměnné daného typu. V závorce je uvedena implementaci daného typu v C, pro lepší pochopení o jaký že datový typ se to vlastně jedná.

Podporovány jsou tyto datové typy:

- **varint** - celé číslo (*int*)
- **varlint** - velké celé číslo (*long*)
- **vardouble** - číslo v pohyblivé řádové čárce (*double*)
- **varchar** - znak (*char*)
- **varbool** - pravdivostní hodnota (*int*)
- **varstring** - řetězec (** char*)
- **varlist** - seznam (**char[]*)

Všechny tyto typy mají stejný způsob zápisu v konfiguračním souboru, liší se jen klíčovým slovem. Daná řádka musí mít tvar

```
keyword "identifier" "value" "range" "comment";
```

nebo

```
keyword "identifier" "value" "range" "comment" "longcomment";
```

tedy obsahuje klíčové slovo (*keyword*), což je jedno z výše uvedených a dále čtyři popřípadě pět hodnot v uvozovkách, vše ukončené středníkem.

Význam jednotlivých hodnot:

- *identifier* - identifikátor proměnné (musí být neprázdný řetězec)
- *value* - hodnota proměnné, může být prázdný řetězec (vyhodnocování viz. níže)
- *range* - rozsah hodnot (seznam hodnot oddělených čárkami, o výčtových typech dále)
- *comment* - popis proměnné pro webové rozhraní
- *longcomment* - podrobné vysvětlení proměnné pro webové rozhraní (není povinné)

Zde je vidět, jak správně komentovat proměnné v konfiguračním souboru.

Vyhodnocování hodnot proměnných. Pro datový typ **varint**, **varlint** a **vardouble** je hodnota získána pomocí funkcí (v tomto pořadí) *atoi()*, *atol()* a *atof()*. Tedy pro řetězec nepředstavující číselnou hodnotu to bude typicky 0.

Typ **varchar** vrátí první znak daného řetězce.

Typ **varlist** je řetězec hodnot oddělených čárkami.

Typ **varbool** nabývá hodnoty 1 (true) pokud se jedná o string "YES" (v tomto jediném případě nezáleží na velikosti písmen), nebo string "TRUE" (opět nezáleží na velikosti písmen) nebo je daný string vyhodnocen funkcí *atoi()* na kladnou hodnotu.

Ukázka sekce obsahující definice proměnných všech jednoduchých datových typů

```
<prezent1>
  varint      "MAX" "100" "" "max threadu" "";
  varlint     "EE" "20000" "" "";
  vardouble   "pi" "3.14" "" "";
  varchar     "zn" "a" "" "";
  varbool     "print_all" "yes" "" "vypisovat vsechny podrobnosti";
  varstring   "email" "ab@cd.net" "mail na spravce" "in case";
  varlist     "dny" "po,ut" "po,ut,st,cp,pa" "udrzba" "zalohovaci dny";
</prezent1>
```

5.2.7 Tabulky

Pro definice tabulek máme následující klíčová slova:

- **tabledef** "id" "n" " t_1 " ... " t_n " - definuje tabulku s daným jménem (id), počtem sloupců (n) a datovými typy daných sloupců (t_1 , ..., t_n). Datové typy mohou být jen jednoduchých datových typů (drobné upřesnění v další sekci).
- **tablerow** "id" " v_1 " ... " v_n " - řádka tabulky s danými hodnotami (v_1 , ..., v_n)
- **tablehelp** "id" "comment" - popis tabulky pro webové rozhraní (může být prázdné)
- **tablecomm** "id" " c_1 " ... " c_n " - popis sloupců tabulky pro webové rozhraní
- **tablecomm** "id" " c_1 " ... " c_n " - podrobný popis sloupců tabulky pro webové rozhraní
- **tabledlist** "id" " l_1 " ... " l_n " - rozsah hodnot pro sloupce tabulky (použit pro tzv. droplist v webovém rozhraní), podobně jako *range* u jednoduchých datových typů, seznam hodnot oddělených čárkami (o výčtových typech dále)

Pro minimální použití tabulek je povinná deklarace pomocí **tabledef** a pro vložení dat deklarace **tablerow**. Ostatní jsou nepovinné, ale měly by být použity kvůli webovému rozhraní.

Všechny definice týkající se jedné tabulky, by měly být v jedné uživatelské sekci.

Ukázka sekce obsahující použití datového typu tabulka

```
<prezent2>
tabledef   "pravidla" "3" "varint" "varlist" "varstring";
tablecomm  "pravidla" "priorita" "seznam dnu" "e-mail";
tablelcomm "pravidla" "" "" "";
tabledlist "pravidla" "1,2,3,4,5" "po,ut,st,ct,pa" "";
tablehelp  "pravidla" "seznam pravidel kam se posilaji chybove hlasky";
tablerow   "pravidla" "2" "po,st" "ab@cz.net";
tablerow   "pravidla" "1" "po,ut,pa" "root@localhost";
</prezent2>
```

5.2.8 Typ pro servery

Většina balancovacích modulů bude mít jako stěžejní část svých proměnných tabulku s pravidly, kam se příchozí požadavky přesměrovávají. Jeden ze sloupců takové tabulky typicky bude obsahovat server(y). Pro zpříjemnění práce uživatele pomocí webového rozhraní byl vytvořen speciální datový typ uvozený klíčovým slovem **varservers**. Proměnné tohoto typu se deklarují stejně jako proměnné jednoduchých datových typů, tj.

```
varservers "identifier" "value" "range" "comment";
```

nebo

```
varservers "identifier" "value" "range" "comment" "longcomment";
```

tedy pátý argument (podrobný komentář) je nepovinný. Ostatní argumenty mají naprosto stejný význam jako u jednoduchých datových typů.

V čem je tedy tento datový typ speciální? Přípustnou hodnotou je buď jméno serverů nebo jméno skupiny serverů (chci-li použít více serverů, musím vytvořit skupinu serverů). Hodnota parametru *range* je tedy irelevantní. Obdobně, pokud se datový typ **varservers** vyskytne jako *i*-tý sloupeček tabulky, pak *i*-tý sloupeček příslušné deklarace **tabledlist** je irelevantní.

V konfiguračním souboru se očekává sekce s jménem *SERVERS_SECTION* a v ní tabulka *SERVERS*, s informacemi o serverech. Dále se očekávají tabulky s jmény *GROUPS*, s informacemi o skupinách

serverů, a *LISTEN_PORTS*, s informacemi o portech, na kterých Load Balancer poslouchá. Webové rozhraní pak díky speciálnímu datovému typu ví, že se jedná o servery a ví, kde je má hledat a práce se tak stává příjemnější.

Datového typu **varservers** může být definována jak jednoduchá proměnná tak i sloupeček tabulky.

Příklad sekce *SERVERS*:

```
<SERVERS_SECTION>
  tabledef "LISTEN_PORTS" "2" "varstring" "varint";
  tablerow "LISTEN_PORTS" "TCP" "11230";
  tablerow "LISTEN_PORTS" "TCP" "11231";

  tabledef "SERVERS" "5" "varstring" "varstring" "varstring" "varint"
"varint";
  tablerow "SERVERS" "serv0" "TCP" "195.113.27.248" "1111" "100";
  tablerow "SERVERS" "serv1" "TCP" "195.113.27.248" "1112" "200";

  tabledef "GROUPS" "2" "varstring" "varstring";
  tablerow "GROUPS" "all" "serv0,serv1,serv2,serv3,serv4,serv5,serv6";
  tablerow "GROUPS" "group1" "serv0,serv1,serv2";
</SERVERS_SECTION>
```

5.2.9 Výčtové typy

Pro zpříjemnění práce uživatele je možné v konfiguračním souboru definovat výčtové typy. To lze pomocí klíčového slova **defenum**, jež očekává identifikátor typu a pak seznam hodnot oddělených čárkami. Příklad:

```
defenum "vsedni_dny" "po,ut,st,ct,pa";
defenum "ang_dny" "mo,tu,we,th,fr,sa,su";
```

Dané typy se pak mohou použít v *range* pro danou proměnnou (či sloupeček tabulky). Do této chvíle jsme věděli, že *range* je seznam hodnot oddělených čárkami. Pokud je některá hodnota uvozena znakem mřížky (#), považuje se za výčtový typ. Pokud daný výčtový typ není v konfiguračním souboru nalezen, bude možná hodnota daný řetězec včetně počátečního znaku mřížky.

Pokud se někde použije výčtový typ, hledá se jeho definice nejprve ve speciální sekci s názvem *ENUM* a následně ve všech uživatelských sekcích daného konfiguračního souboru. Kvůli rychlosti je doporučeno umístit všechny výčtové typy do uživatelské sekce *ENUM*.

Příklad použití výčtového typu:

```
<core_base>
  varstring "den" "po" "#vsedni_dny,so,ne";
</core_base>

<ENUM>
  typedef "vsedni_dny" "po,ut,st,ct,pa";
  typedef "ang_dny" "mo,tu,we,th,fr,sa,su";
</ENUM>
```

5.2.10 Kontrola správnosti

Load Balancer si před svým spuštěním kontroluje daný konfigurační soubor. Kromě syntaktické správnosti i správný počet argumentů za příslušným klíčovým slovem, správný počet sloupečků v tabulce, známé datové typy. Ale neprovádí sémantickou kontrolu, viz. například dříve zmíněný problém o sjednocení nedisjunktních uživatelských sekcí se stejným jménem. V takových případech nese plnou zodpovědnost uživatel.

Startovací modul Load Balanceru Init lze také spustit pouze pro kontrolu standardního či zadaného konfiguračního souboru.

5.2.11 Ukázkový konfigurační soubor

Následuje kompletní ukázkový konfigurační soubor, jehož jednotlivé části bylo možno vidět v průběhu této kapitoly.

```
<MAIN>
  core core_base core_ext SERVERS_SECTION;
  stat statistics SERVERS_SECTION;
  round_robin1;
  prezent2;
</MAIN>

<ALIASES>
  round_robin1 mod1basic mod1A mod1C;
  round_robin1 mod2php PHP;
  round_robin2 mod1basic mod1B mod1C;
  round_robin2 mod2php PHPnew;
  prezent1 prezent prezent1 prezent_old_style;
  prezent2 preeznt prezent1 prezent_new_style;
</ALIASES>
```

```

<core_base>
  varstring "den" "po" "#vsedni_dny,so,ne" "den kdy se zalohuje";
</core_base>

<core_ext>
  varint "core_n" "2" "1,2" "pocet kopii";
</core_ext>

<mod1A>
  varint "A" "10" "" "";
</mod1A>

<mod1B>
  varint "A" "20" "" "";
  varint "velikost" "10" "" "";
</mod1B>

<mod1C>
  varint "C" "100" "" "";
</mod1C>

<mod1D>
  varint "C" "200" "" "";
  varint "size" "5" "" "";
</mod1D>

<PHP>
  varstring "php" "3.0" "" "";
</PHP>

<PHPnew>
  varstring "php" "4.0" "" "";
  varstring "php2" "3.0" "" "";
</PHPnew>

<prezent_old_style>
  varlist "colors" "black,white" "" "";
</prezent_old_style>

<prezent_new_style>
  varlist "colors" "red,blue,white" "" "";
</prezent_new_style>

```

```

<prezent1>
  varint    "MAX" "100" "" "max threadu" "";
  varlint   "EE" "20000" "" "";
  vardouble "pi" "3.14" "" "";
  varchar   "zn" "a" "" "";
  varbool   "print_all" "yes" "" "vypisovat vsechny podrobnosti";
  varstring "email" "ab@cd.net" "" "mail" "pouzijte v nouzi";
  varlist   "dny" "po,ut" "po,ut,st,ct,pa" "udrzba" "kdy zalohovat";
</prezent1>

<prezent2>
  tabledef  "pravidla" "3" "varint" "varlist" "varstring";
  tablecomm "pravidla" "priorita" "seznam dnu" "e-mail";
  tablelcomm "pravidla" "" "" "";
  tabledlist "pravidla" "1,2,3,4,5" "po,ut,st,ct,pa" "";
  tablehelp "pravidla" "pravidla kam se posilaji chybove hlasky";
  tablerow  "pravidla" "2" "po,st" "ab@cz.net";
  tablerow  "pravidla" "1" "po,ut,pa" "root@localhost";
</prezent2>

<SERVERS_SECTION>
  tabledef "LISTEN_PORTS" "2" "varstring" "varint";
  tablerow "LISTEN_PORTS" "TCP" "11230";
  tablerow "LISTEN_PORTS" "TCP" "11231";

  tabledef "SERVERS" "5" "varstring" "varstring" "varstring" "varint"
"varint";
  tablerow "SERVERS" "serv0" "TCP" "195.113.27.248" "1111" "100";
  tablerow "SERVERS" "serv1" "TCP" "195.113.27.248" "1112" "200";

  tabledef "GROUPS" "2" "varstring" "varstring";
  tablerow "GROUPS" "all" "serv0,serv1,serv2,serv3,serv4,serv5";
  tablerow "GROUPS" "group1" "serv0,serv1,serv2";
</SERVERS_SECTION>

<statistics>
  varint "max" "15" "" "";
</statistics>

<ENUM>
  defenum "vsedni_dny" "po,ut,st,ct,pa";
  defenum "ang_dny" "mo,tu,we,th,fr,sa,su";
</ENUM>

```

Kapitola 6

Webové rozhraní

6.1 Monitoring

V části monitoring je možné zjišťovat aktuální stav všech serverů (služeb) na farmě. Zobrazované údaje jsou následující:

název Název serveru, který je mu přiřazen v konfiguračním souboru.

IP adresa IP adresa daného serveru.

port Port, na kterém služba běží.

stav Aktuální stav daného serveru.

Na následujícím obrázku je zobrazen příklad zobrazení stavu serverů.

Name	IP address	Port	State
test 1	195.113.17.191	80	✓
test 2	195.113.21.150	80	✗
test 3	195.113.20.12	80	?

Obrázek 6.1: Stav serverů

Server *test 1* v pořádku běží, stav serveru *test 3* není známý (např. mohlo dojít k timeoutu při posledním connectu na server) a na serveru *test 2* došlo k chybě a není funkční.

Dále je možné zobrazit všechny servery, které byly na farmě použity, i když v současné době nejsou používány (tj. nejsou příslušně nastaveny v

konfiguračním souboru). Tyto servery se samozřejmě zobrazí pouze v případě, že pro ukládání statistických informací je použito databáze.

Aktuální stav serverů v monitorovací části je obnovován každých 10 sekund. Pokud není dostupný statistický modul, nemohou být zobrazeny žádné údaje.

6.2 Zobrazování statistik

V této části je možné zobrazovat různé statistické informace o práci Load Balanceru. Uživatel si vytvoří dotaz, kde specifikuje jaká data a v jakém časovém intervalu chce zobrazit. Po odeslání dotazu se mu zobrazí požadovaná data jednak v podobě grafu a jednak v podobě tabulky výsledných hodnot. Výsledná data se aktualizují každou minutu.

6.2.1 Vytváření dotazu

V první části zadávání dotazu si uživatel může vybrat ze seznamu serverů a jim příslušejících datových typů, o kterých z nich chce zobrazit informace. Aby mohl být položen dotaz, musí být vybrán alespoň jeden datový typ u nějakého serveru.

service :

Name	IP address	Port	data_type	
serv0	212.20.96.20	80	def_tcp	<input type="checkbox"/>
			tcp	<input checked="" type="checkbox"/>
			www	<input type="checkbox"/>
serv1	212.20.96.22	80	www	<input type="checkbox"/>
			tcp	<input checked="" type="checkbox"/>
serv2	212.20.96.23	80	www	<input type="checkbox"/>
			tcp	<input checked="" type="checkbox"/>
serv3	212.20.96.24	80	www	<input type="checkbox"/>
			tcp	<input checked="" type="checkbox"/>

Obrázek 6.2: Výběr serverů a datového typu

Na příkladě si uživatel vybral zobrazení informací o datovém typu *tcp* u všech serverů .

Pozn. Názvy datových typů jsou vytvářeny z názvů protokolů a vyvažovacích pravidel z konfiguračního souboru, kterým daná data vyhovovala.

V další části zadávání dotazu uživatel určuje typ informací, které chce zobrazit. Může si vybrat z následujících typů (*info_type*):

RESPONSE Odezva serveru (v milisekundách).

FLOW Datový tok (v Bytech za sekundu).

FLOW FROM SERVER Datový tok od serveru ke klientovi (v Bytech za sekundu).

FLOW FROM CLIENT Datový tok od klienta ke serveru (v Bytech za sekundu).

CONNECTIONS PER SECOND Počet uskutečněných spojení za vteřinu.

REQUESTS PER SECOND Počet zpracovaných požadavků za vteřinu.

CONNECT TIME Doba trvání vytvoření spojení se serverem.

U každého z těchto typů uživatel může zvolit, jestli chce zobrazit maximální, průměrné nebo minimální hodnoty (*value_type*). Dále je možné zvolit ještě dva speciální typy **CONNECTIONS** (počet uskutečněných spojení) a **REQUESTS** (počet zpracovaných požadavků), u kterých maximální, průměrné ani minimální hodnoty nemají smysl. Aby mohl být položen dotaz, musí být některý z typů informací vybrán.

V poslední části zadávání dotazu uživatel určuje časový interval, pro který chce vybrané informace zobrazit. Může si buď vybrat z listu standardních hodnot nebo může časový interval určit přímým zadáním jeho mezních hodnot. Na základě zadaného intervalu se automaticky určí velikost agregace tj. jak dlouhé časové intervaly se agregují do jednotlivých hodnot výsledku, aby byl přehledný. Většinou tyto intervaly bývají o řád nižší je řád intervalu, o kterém se hodnoty zobrazují (pokud se například uživatel zeptá na hodnoty z poslední hodiny, jednotlivé hodnoty odpovědi se zobrazí agregované po minutách).

Na následujícím obrázku je příklad s kompletním dotazem:

service :

Name	IP address	Port	data_type	
serv0	212.20.96.20	80	def_tcp	<input type="checkbox"/>
			tcp	<input checked="" type="checkbox"/>
			www	<input type="checkbox"/>
serv1	212.20.96.22	80	www	<input type="checkbox"/>
			tcp	<input checked="" type="checkbox"/>
serv2	212.20.96.23	80	www	<input type="checkbox"/>
			tcp	<input checked="" type="checkbox"/>
serv3	212.20.96.24	80	www	<input type="checkbox"/>
			tcp	<input checked="" type="checkbox"/>

info type : or info type :

value type :

time range :

or

time from:
hour: minute: second:
day: month: year:

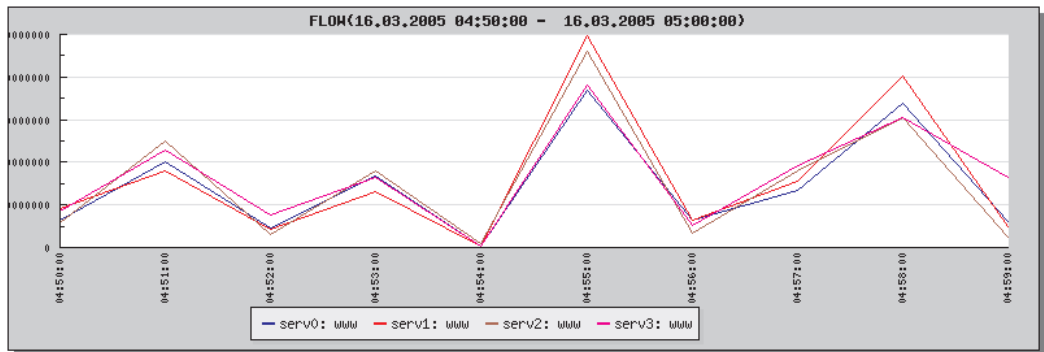
time to:
hour: minute: second:
day: month: year:

Obrázek 6.3: Kompletní dotaz

6.2.2 Zobrazení výsledku dotazu

Po stisknutí tlačítka *Show* se odešle dotaz statistickému modulu a zobrazí se příchozí odpověď. Pro každý datový typ každého serveru jsou výsledné hodnoty zobrazeny jako samostatná křivka v grafu a samostatný sloupec hodnot v následné tabulce.

Na následujícím obrázku je zobrazen příklad odpovědi na dotaz průměrného toku dat v intervalu 10 minut na vybraných serverech:



FLOW

16.03.2005 04:50:00 - 16.03.2005 05:00:00

	serv0: www	serv1: www	serv2: www	serv3: www
16.03.2005 04:50:00	12911073	18444902	11499486	16829491
16.03.2005 04:51:00	39962627	35780886	50233930	45734776
16.03.2005 04:52:00	9107820	8336780	6304561	15347077
16.03.2005 04:53:00	33411771	26425383	35959374	33017732
16.03.2005 04:54:00	499154	399624	1957623	563161
16.03.2005 04:55:00	73799652	99155361	91903437	76103637
16.03.2005 04:56:00	12691033	13013891	6670939	10574994
16.03.2005 04:57:00	26814288	30933811	35737838	38211602
16.03.2005 04:58:00	67798458	80251549	60826853	61183421
16.03.2005 04:59:00	12361838	9975147	4631462	32650961

Obrázek 6.4: Odpověď na dotaz

6.3 Konfigurace

V části konfigurace je umožněno měnit nastavení Load Balanceru pomocí změn v konfiguračním souboru a propagovat je do běžícího Load Balanceru.

Konfigurace obsahuje pět částí:

- Admin
- View Config
- Main
- Aliases
- Sections

6.3.1 Administrace

Tato sekce umožňuje práci s jednotlivými konfiguračními soubory, restartování a ukončení Load Balanceru. Webové rozhraní pracuje se třemi konfiguračními soubory:

running aktuální konfigurace Load Balanceru, při startu ho načítá modul `init`.

default obsahuje základní nastavení Load Balanceru.

editing uchovává změny provedené webovým rozhraním.

Sekce administrace obsahuje pět tlačítek:

Load Nahraje do editovaného konfiguračního souboru právě běžící konfigurační soubor. Pokud tento soubor neexistuje, nahraje se místo něho defaultní konfigurační soubor.

Load default Nahraje do editovaného konfiguračního souboru defaultní konfigurační soubor.

Save Uloží editovaný konfigurační soubor do právě běžícího konfiguračního souboru. (změny se projeví při novém spuštění nebo při restartu)

Restart Restartuje celý Load Balancer. Modul `init` načte ze souboru konfigurační soubor a rozešle ostatním modulům signál restartu. Ty po svém restartu načtou nový konfigurační soubor.

Shutdown Ukončí běh celého Load Balanceru.

6.3.2 Náhled konfiguračního souboru

V této části je vidět editovaný konfigurační soubor v textové podobě. Při změně v některé části přes webové rozhraní je možné tyto změny vidět i v textové podobě. Tento soubor je případně uložen jako běžící a ostatní části Load Balanceru ho využívají v této podobě.








6.3.3 Main

V této části je možné upravovat sekci `MAIN` z konfiguračního souboru. Každému záznamu odpovídá jedna odrážka. Jednotlivé záznamy lze mazat pomocí ikony koše vedle názvu modulu. Pokud modul obsahuje více sekcí, pak je možné je mazat ikonou koše vedle jejich názvu. Na konci seznamu

sekcí daného modulu je možnost přidat existující sekci výběrem ze seznamu a tlačítkem add new section.

Pod seznamem všech modulů je řádek umožňující přidání nového modulu. Po zadání názvů a jedné sekce, kterou bude obsahovat se vytvoří stisknutím tlačítka add new module.

MODULES

- **init** 
 - [MODULE_INTT](#)
 - 
- **core** 
 - [MODULE_CORE](#) 
 - [MODULE_PING](#) 
 - 
- Modul: Section: 

Obrázek 6.5: Editace sekce MAIN

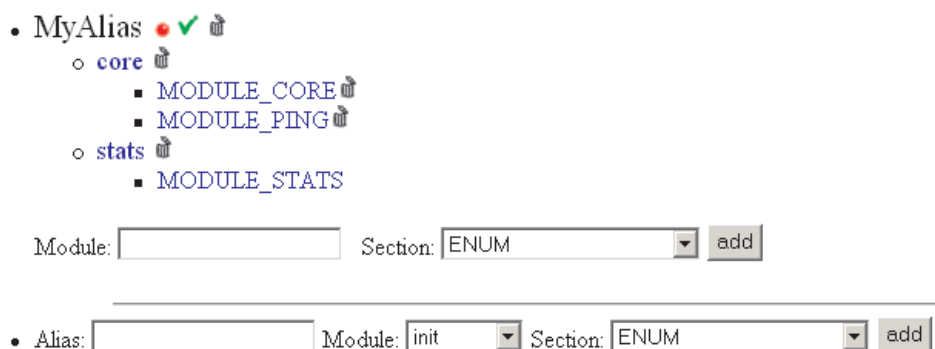
Ve spodní polovině je tabulka informující o sekci ALIASES. Je rozdělena do dvou sloupců. V levé polovině je seznam aktivních aliasů (tzn. které jsou použity v sekci MAIN). Po stisknutí tlačítka Disable jsou všechny aliasy, které byly zaškrtnuty přesunuty do neaktivních.

V pravé polovině je seznam neaktivních aliasů. Po stisknutí tlačítka Enable jsou všechny aliasy, které byly zaškrtnuty přesunuty do aktivních.

6.3.4 Aliases

V této části je možné upravovat sekci ALIASES z konfiguračního souboru.

ALIASES



Obrázek 6.6: Editace sekce ALIASES

Každý alias je zahájen odrážkou a ukončen horizontální oddělovací čarou. Vedle názvu je trojice ikon. První informuje o stavu aliasu. Zelený kroužek označuje aktivní alias, červený neaktivní. Druhá ikona umožňuje aktivaci nebo deaktivaci aliasu a třetí ikona koše provede zrušení aliasu.

Pod názvem je uvedený seznam modulů, které alias nahrazuje v sekci MAIN. Vedle každého názvu modulu je ikona koše pro zrušení modulu. U každého modulu je lokální seznam sekcí (stejný jako u modulů v sekci MAIN). Pokud je uvedena více než jedna sekce, je u každé zobrazena ikona koše pro zrušení příslušné sekce.

Na konci seznamu modulů je u každého aliasu je přidání nové sekce do aliasu. Pokud je sekce přidávána do už existujícího modulu, přidá se do seznamu daného modulu. Pokud modul neexistuje, nejdříve se vytvoří a pak je k němu jako jediná položka vložena nová sekce. Sekce se přidá po stisknutí tlačítka add.

Pod seznamem všech aliasů je část umožňující přidání nového aliasu. Pro vytvoření je nutné zadat název, modul a sekci a stisknout tlačítko add.

6.3.5 Sekce

Na úvodní stránce je zobrazen seznam všech sekcí v konfiguračním souboru. Sekce, které nejsou uvedeny u žádného modulu ani aliasu, mají vedle svého názvu ikonu koše pro jejich zrušení. Pod seznamem sekcí je část umožňující přidání nové sekce. Sekce se přidá po zadání názvu a stisknutí tlačítka add new section.

Po kliknutí na vybranou sekci se ve spodní části stránky zobrazí jednotlivé proměnné, výčtové typy a tabulky, které vybraná sekce obsahuje.

Každá položka je zobrazeny v samostatném rámečku. Vedle názvu položky je uveden v závorce datový typ. Vedle datového typu je zobrazena ikona koše, která umožňuje položky ze sekce odstranit

Výčtové typy by měly být definovány v sekci ENUM, která je pro tyto typy vyhrazená, ale mohou být definovány i v uživatelských sekcích.

log_level (defenum)

(if you can delete leave field empty)

Value: Value: Value:

Value: Value: Value:

Value: Value:

Add new:

(use comma for add more new values - value1,value2,value3)

Obrázek 6.7: Editace výčtového typu

V horní částí výčtového typu je zobrazen seznam hodnot. Hodnota se mění přímou editací a následným stisknutím tlačítka change. Pokud je nová hodnota prázdný řetězec, je daná hodnota z výčtového typu odstraněna. Nová hodnota se přidá pomocí editačního okna Add new a stisknutím tlačítka change. Více hodnot se odděluje pomocí čárky.

INIT_STATS_STARTUP_TIMEOUT (varint)

Value:

(Stats sartup timeout in sec.)


INIT_SYSLOG_LEVEL (varstring)


(Set level of messages written to syslog in all modules.)

Obrázek 6.8: Editace proměnné

Pokud má proměnná definovaný rozsah hodnot, je hodnota zobrazena jako rozbalovací seznam hodnot. Její změna se provede výběrem jiné hodnoty a stisknutím tlačítka change.

Pokud proměnná nemá definovaný rozsah hodnot, je hodnota zobrazena v editačním okně. Změna hodnoty se provede přímou editací a stisknutím tlačítka change.

SERVERS (tabledef) 

	Name	IP protocol	IP address	IP port	Static weight
	<input type="text" value="my server"/>	<input type="text" value="TCP"/>	<input type="text" value="192.168.0.3"/>	<input type="text" value="80"/>	<input type="text" value="100"/>
	<input type="text"/>	<input type="text" value="TCP"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Obrázek 6.9: Editace tabulky

První řádek tabulky zobrazuje názvy jednotlivých sloupců. Pak jsou uvedeny jednotlivé řádky s daty. Poslední řádek je prázdný a slouží k zadávání nových hodnot. Změna hodnot se provádí podobně jako u proměnných přímou editací nebo výběrem nové hodnoty a stisknutím tlačítka change. Přidání nové hodnoty vyplněním spodního prázdného řádku a stisknutím tlačítka change.

Na konci stránky pod všemi položkami obsaženými v sekci je tlačítko Add, které umožňuje přidání nových položek. Po stisknutí tlačítka se otevře nová stránka, na které je trojice rámečků pro přidání nové proměnné, tabulky a výčtového typu.

New variable

Choose type:

Write name:

Write value:

Write range:

Write short comment:

Write long comment:

In range you can use list of values or enums separated by comma.

Obrázek 6.10: Přidání nové proměnné

Pro přidání nové proměnné je nutné vyplnit její datový typ a název. Další hodnoty, které se mohou vyplnit jsou hodnota, rozsah hodnot, krátký komentář a dlouhý komentář. Nová proměnná se přidá tlačítkem Add.

New table

Write name:

Write number of collums:

Write comment:

Obrázek 6.11: Přidání nové tabulky - první krok

Pro přidání nové tabulky je nutné název a počet sloupců. Další hodnota, která se může vyplnit je komentář k celé tabulce. K dalšímu kroku vytvoření tabulky se postoupí stisknutím tlačítka Next.

MyNewTable

Write name of collumnus to next row.

First	Second	Third	Fourth	Fifth
-------	--------	-------	--------	-------

Write type of collumnus to next row.

varint	vardouble	varchar	varbool	varservers
--------	-----------	---------	---------	------------

Write range of collumnus to next row.

			#bool	
--	--	--	-------	--

Write comment of collumnus to next row.

--	--	--	--	--

Add

This is my brand new table. I will use to store my own values.

Obrázek 6.12: Přidání nové tabulky - druhý krok

V druhém kroku je nutné pouze správně nastavit datové typy jednotlivých sloupců, které se vybírají ze seznamu hodnot na druhém řádku. Ostatní parametry jsou volitelné. Na prvním řádku je možné zadat názvy jednotlivých sloupců, na třetím rozsahy hodnot a na čtvrtém komentáře. Tabulka se přidá stisknutím tlačítka Add.

New enum

Write name: MyEnum

Write values: Sun,Mon,Tue,Wen

Add

!!! Separate values by comma !!!

Obrázek 6.13: Přidání nového výčtového typu

Pro přidání nového výčtového typu je nutné vyplnit jeho název a seznam hodnot. Jednotlivé hodnoty se oddělují čárkou. Výčtový typ se přidá stisknutím tlačítka Add.

Kapitola 7

Podmoduly

V této části jsou popsány jednotlivé algoritmy, které jsou součástí Load Balanceru. Popis, jak přidat nové vlastní podmoduly nebo upravit existující, je uveden v programátorské části dokumentace.

7.1 Knihovna první vrstvy

Součástí Load Balanceru je knihovna *lib_layer1_main.so* s funkcí *common*. Tato funkce se řídí podle tabulky pravidel *L1_RULES* v sekci *<LAYER1_SUBMODULE>* konfiguračního souboru. Funkce postupně prochází tabulkou a hodnoty každého pravidla porovnává s hodnotami aktuálního spojení. Pravidlo se uplatní, pokud vyhovuje ve všech parametrech.

Pravidlo tvoří 14 sloupců:

název Název pravidla. Pokud se pravidlo uplatní, přidá se jeho název do datového typu spojení pro jeho identifikaci ve statistikách.

stats Pokud je parametr nastaven, budou se informace o spojení odesílat do statistik.

protokol Seznam protokolů, na které se pravidlo vztahuje. Pokud není uveden žádný protokol, je pravidlo ignorováno.

cílový port Port služby, na které poslouchá jádro (např. 80 pro HTTP). Pokud se rovná nule, vyhovuje pravidlu libovolný port daných protokolů.

zdrojový port Port, ze kterého se připojil klient. Pokud se rovná nule, vyhovuje pravidlu libovolný port.

zdrojová IP adresa

zdrojová maska adresy Porovnává IP adresu klienta s adresou a maskou v pravidlu. Pravidlo vyhovuje pokud je splněna alespoň jedna z následujících podmínek:

- IP adresa je prázdná nebo se po převedení na číslo rovná nule.
- Maska je prázdná nebo se po převedení na číslo rovná nule a zároveň se zdrojová IP adresa rovná IP adrese pravidla.
- Zdrojová adresa spadá do rozsahu adres definovaných IP adresou a maskou pravidla.

knihovna na druhé vrstvě

funkce na druhé vrstvě Dvojice parametrů určující funkci a knihovnu na druhé vrstvě. Pokud jsou oba parametry platné a pravidlo bude uplatněno, zavolá se tato funkce po skončení funkce common.

knihovna na třetí vrstvě

funkce na třetí vrstvě Dvojice parametrů určující funkci a knihovnu na třetí vrstvě. Pokud jsou oba parametry platné, pravidlo bude uplatněno a na druhé vrstvě nebyla zadána platná knihovna nebo funkce, zavolá se tato funkce po skončení funkce common.

knihovna na datové vrstvě

funkce na datové vrstvě Dvojice parametrů určující funkci a knihovnu na datové vrstvě. Pokud jsou oba parametry platné a pravidlo bude uplatněno, je tato funkce volána v jádru během provádění spojení v okamžiku volání funkce podmodulu na datové vrstvě.

servery Identifikace serveru nebo skupiny. Pokud je pravidlo uplatněno, je všem serverům, které odpovídají parametru, nastavena priorita 1. Mezi těmito servery se vybírá na dalších vrstvách. Pokud parametru neodpovídá žádný server, je pravidlo ignorováno.

7.2 Knihovna druhé vrstvy

Součástí Load Balanceru je knihovna *http_layer2.so* s funkcí *http_headers*. Parametry nastavující tuto funkci jsou v sekci `<HTTP_SUBMODULE>` konfiguračního souboru. Tato funkce je určena pro balancování HTTP protokolu

na aplikační vrstvě podle standardu RFC2616. Funkce načte HTTP hlavičku a na základě jejího obsahu rozdělí spojení. Načítání HTTP hlavičky je omezeno těmito parametry:

MAX_URI Maximální délka URI.

MAX_URI_HEADER_BODY Maximální délka těla záznamu header.

MAX_HEADERS Maximální počet záznamů header.

Z načtené hlavičky funkce získá jednotlivé části, které je pak možné porovnávat v pravidlech. Pokud není zadáno vyhledávání v záznamech headers, jsou přípustné hodnoty parametru "kde aplikovat" v tabulce pravidel tyto:

default Pokud je uveden tento parametr, je pravidlo uplatněno bez ohledu na další parametry.

method Metoda protokolu (např. GET, POST, ...).

request URI Nezměněná URI, tak jak je přijata v hlavičce.

HTTP version Verze HTTP protokolu, která je uvedena za URI.

scheme Schema požadavku (např. http).

host Hostitel (adresa serveru) požadavku. Pokud je v záznamech headers záznam "Host", použije se jeho adresa.

port Cílový port požadavku.

absolute path Absolutní cesta požadavku na serveru.

query Dotaz požadavku, obsahuje počáteční otazník.

Pokud v požadavku není uvedeno schema, doplní se na "http". Pokud v požadavku není uveden port, doplní se na hodnotu 80. Pokud v požadavku není uvedena absolutní cesta, doplní se na "/".

Pokud je zadáno vyhledávání v záznamech headers, použije se pro porovnání takový záznam header, který má shodný název s názvem v pravidlu.

Seznam pravidel je uvedený v tabulce *HTTP_RULES* a tvoří ji 14 sloupců:

název Název pravidla. Pokud se pravidlo uplatní, přidá se jeho název do datového typu spojení pro jeho identifikaci ve statistikách.

stats Pokud je parametr nastaven, budou se informace o spojení odesílat do statistik.

porovnávat v headers Pokud je parametr nastaven, hledá se řetězec v záznamu header určeným parametrem kde aplikovat. Pokud není nastaven, hledá se řetězec v hlavičce.

kde aplikovat Pokud je nastaveno hledání v záznamech headers, obsahuje název záznamu header. Pokud není nastaveno hledání v záznamech headers, může obsahovat jednu z hodnot "default", "method", "request URI", "HTTP version", "scheme", "host", "port", "absolute path", "query".

regulární výraz Pokud je parametr nastaven, bere se hledaný řetězec jako regulární výraz.

case sensitive Pokud je parametr nastaven, bere se hledaný řetězec jako case-sensitive.

řetězec Řetězec, který se porovnává s požadavkem podle výše zmíněných pravidel.

knihovna na druhé vrstvě

funkce na druhé vrstvě Dvojice parametrů určující funkci a knihovnu na druhé vrstvě. Pokud jsou oba parametry platné a pravidlo bude uplatněno, zavolá se tato funkce po skončení funkce http_headers. Tato funkce může být volána znovu pro jemnější ladění požadavků.

knihovna na třetí vrstvě

funkce na třetí vrstvě Dvojice parametrů určující funkci a knihovnu na třetí vrstvě. Pokud jsou oba parametry platné, pravidlo bude uplatněno a na druhé vrstvě nebyla zadána platná knihovna nebo funkce, zavolá se tato funkce po skončení funkce http_headers.

knihovna na datové vrstvě

funkce na datové vrstvě Dvojice parametrů určující funkci a knihovnu na datové vrstvě. Pokud jsou oba parametry platné a pravidlo bude uplatněno, je tato funkce volána v jádru během provádění spojení v okamžiku volání funkce podmodulu na datové vrstvě.

servery Identifikace serveru nebo skupiny. Pokud je pravidlo uplatněno, je všem serverům, které odpovídají parametru nastavena priorita 1. Mezi těmito servery se vybírá na dalších vrstvách. Pokud parametru neodpovídá žádný server, je pravidlo ignorováno.

Pokud je uplatněno pravidlo, ponechá se nastavená priorita u těch serverů, které vyhovují pravidlu. U ostatních se nastaví na nulu.

Další funkcí tohoto podmodulu je využití cache, která se použije, pokud je nastavená proměnná *USE_CACHE*.

Cache je seznam řádek, kde každý odpovídá jednomu hashovacímu číslu. Každý řádek obsahuje položky, které mají stejné hashovací číslo (kolizní doména). V každé položce je uložen hashovaný řetězec, id serveru a timestamp. V cache je uloženo číslo serveru, kam bylo naposled přeposláno spojení. Při dotazu podmodulu se postupně projde určená kolizní doména určená hashovacím číslem a pokud se v ní nalezne příslušný záznam, vrátí funkce číslo serveru a timestamp.

Parametr *CACHE_SIZE* určuje počet řádků cache, parametr *CACHE_DOMAIN_SIZE* určuje počet záznamů v kolizní doméně a parametr *CACHE_STRING_SIZE* určuje maximální délku hashovaného řetězce.

Pro použití cache se vybere řetězec, který je určen parametrem *CACHE_WHERE_APPLY*. Ten může obsahovat buď jednu z hodnot "method", "request URI", "HTTP version", "scheme", "host", "port", "absolute path", "query", která určuje, že se jako hashovací řetězec použije příslušná pevně zvolená část hlavičky nebo hodnotu "rule match", která jako hashovací řetězec použije hodnotu, která odpovídá uplatněnému pravidlu.

Pokud je z cache vrácen server, je mezi vybranými servery a záznam v cache není starší než *CACHE_AGE* sekund, je k jeho prioritě přičtena hodnota *CACHE_PRIORITY*.

7.3 Knihovny třetí vrstvy

Součástí Load Balanceru je knihovna *static_layer3.so*, která obsahuje statické algoritmy a knihovna *dynamic_layer3.so*, která obsahuje dynamické algoritmy, které pro rozhodování používají dotazování do statistik.

7.3.1 Statické algoritmy

Basic round robin

Základní algoritmus, který si pamatuje poslední vybraný server a prioritu nastaví serverům takovým způsobem, aby následujícímu serveru, který má prioritu větší než nula, nastavil největší prioritu a pak ji postupně snižuje. Algoritmus není vhodný pro takovou konfiguraci farmy serverů, kde se překrývá nějaká skupina serverů. Pokud jsou vybrány všechny servery a jsou v jedné skupině, algoritmus servery vybírá postupně podobně jako DNS-round robin algoritmus. Tento algoritmus je rychlý a proto je vhodný jako náhrada DNS - round robin algoritmu

Random select

Algoritmus, který vybere cílový server náhodně ze všech serverů, které mají prioritu větší než nula.

Hash IP select

Algoritmus, který vybere cílový server na základě zdrojové IP adresy. Číslo adresy vydělí počtem serverů, které mají prioritu větší než nula. Zbytek po dělení určuje index mezi těmito servery.

Self weighted round robin

Algoritmus, který rozšiřuje základní round robin algoritmus na použití, kdy jsou servery ve více skupinách. Algoritmus si ukládá počet spojení u každého serveru. Ze všech serverů s prioritou větší než nula vybere server s nejmenším počtem spojení. Pokud je nejmenší počet spojení u několika serverů, vybere server s nejmenším indexem. Tento algoritmus je vhodný pro použití, kde se nepřekrývají skupiny serverů a na každou skupinu by se použil algoritmus round robin.

Static select first

Algoritmus, který ke svému výpočtu využívá statických vah jednotlivých serverů. Použije se pokud mají servery různý výkon. Jejich výkon se jako statická váha nastaví v konfiguračním souboru. Algoritmus zajišťuje aby byla zátěž rovnoměrně podle statických vah. Algoritmus je použitelný, ikdyž se skupiny serverů překrývají.

U každého serveru si funkce ukládá počet spojení směřovaných na tento server. Během výpočtu se pro každý server, který má prioritu větší než

nula, spočítá aktuální váha jako $actual = \frac{weight}{static_weight * priority}$, kde *weight* je počet spojení uskutečněných už na daný server, *static_weight* je statická váha serveru z konfiguračního souboru a *priority* je priorita serveru z předchozích podmodulů. Aktuální váha vyjadřuje váhu serverů po započítání statické váhy serveru a jeho priorit (preferencí) z předchozích podmodulů. Tím se smaže rozdíl mezi výkonností serverů a algoritmus se pak snaží, aby ta to váha byla u všech serverů stejná. Proto vybere server s nejmenší aktuální váhou.

7.3.2 Dynamické algoritmy

Dynamické algoritmy používají ke svému rozhodování data uložená ve statistickém modulu. Uživatel má možnost nastavit, jestli chce tyto dotazy provádět při zpracování každého požadavku nebo stačí informace získané dotazem obnovit po určité době. To lze provést nastavením hodnot v tabulce *L3_QUERY_RATES* v konfiguračním souboru v sekci *<LAYER3_SUBMODULE>*:

pravidlo Název pravidla z tabulky *L1_RULES*, pro které zadané hodnoty použít. Je-li tato hodnota prázdná, použijí se zadané hodnoty pro všechna pravidla, které v této tabulce nejsou obsažena, jinak se pro všechna taková pravidla provádějí dotazy při zpracování každého požadavku.

počet požadavků Nový dotaz pro aktualizaci hodnot je položen vždy po zpracování daného počtu požadavků.

doba mezi dotazy Nový dotaz pro aktualizaci hodnot je položen vždy po dané době uvedené v milisekundách.

Minimal server response

Tento algoritmus třídí vybranou skupinu serverů na základě jejich maximální odezvy. Nejvyšší prioritu dostane přiřazenou ten server, jehož maximální odezva za poslední vteřinu je nejnižší.

Minimal average server response

Tento algoritmus třídí vybranou skupinu serverů na základě jejich průměrné odezvy. Nejvyšší prioritu dostane přiřazenou ten server, jehož průměrná odezva za poslední vteřinu je nejnižší. Vylepšuje tak předchozí algoritmus v případě, že odezva u některého ze serverů má větší rozptyl než u ostatních.

Least open connections

Tento algoritmus třídí vybranou skupinu serverů na základě počtu jejich aktuálně otevřených spojení. Nejvyšší prioritu dostane přiřazenou ten server, jenž má aktuálně nejméně otevřených spojení.

Weighted least open connections

Vylepšuje předchozí algoritmus o možnost nastavení různého výkonu jednotlivým serverům. Tento algoritmus třídí vybranou skupinu serverů na základě počtu jejich aktuálně otevřených spojení a na základě statické váhy přiřazené danému serveru. Nejvyšší prioritu dostane přiřazenou ten server, jenž má minimální poměr počtu aktuálně otevřených spojení ku statické váze.

Weighted minimal sessions count

Tento algoritmus třídí vybranou skupinu serverů na základě počtu nových spojení otevřených za poslední sekundu a na základě statické váhy přiřazené danému serveru. Nejvyšší prioritu dostane přiřazenou ten server, jenž má minimální poměr počtu nových spojení otevřených za poslední sekundu ku statické váze.

Weighted minimal flow from server

Tento algoritmus třídí vybranou skupinu serverů na základě velikosti toku dat od serveru za poslední sekundu a na základě statické váhy přiřazené danému serveru. Nejvyšší prioritu dostane přiřazenou ten server, jenž má minimální poměr velikosti toku dat od serveru za poslední sekundu ku statické váze.

Predictive minimal server response

Tento algoritmus třídí vybranou skupinu serverů na základě posledního vývoje jejich maximální odezvy. Čím víc se zmenšuje maximální odezva oproti jejím předchozím hodnotám, tím víc požadavků může být na server směrováno a tím větší by tedy měl mít prioritu. Servery jsou porovnávány na základě čísla $c = \frac{o_{10}}{o_5} * \frac{o_5}{o_1}$, kde o_{10} je maximální odezva z posledních deseti odezev serveru, o_5 je maximální odezva z posledních pěti odezev serveru a o_1 je poslední odezva serveru. Nejvyšší prioritu dostane přiřazenou ten server, u kterého je výsledné číslo c největší.

Predictive minimal flow from server

Tento algoritmus třídí vybranou skupinu serverů na základě posledního vývoje velikosti datového toku od serveru. Čím víc se zmenšuje velikost datového toku od serveru oproti jeho předchozím hodnotám, tím víc požadavků může být na server směrováno a tím větší by tedy měl mít prioritu. Servery jsou porovnávány na základě čísla $c = \frac{t_{10}}{t_5} * \frac{t_5}{t_1}$, kde t_{10} je průměrný datový tok od serveru za posledních deset sekund, t_5 je průměrný datový tok od serveru za posledních pět sekund a t_1 je datový tok od serveru za poslední sekundu. Nejvyšší prioritu dostane přiřazenou ten server, u kterého je výsledné číslo c největší.

Predictive minimal sessions count per second

Tento algoritmus třídí vybranou skupinu serverů na základě posledního vývoje počtu jejich spojení za sekundu. Čím víc se zmenšuje počet uskutečněných spojení za sekundu oproti předchozím počtům, tím víc požadavků může být na server směrováno a tím větší by tedy měl mít prioritu. Servery jsou porovnávány na základě čísla $c = \frac{s_{10}}{s_5} * \frac{s_5}{s_1}$, kde s_{10} je průměrný počet uskutečněných spojení za sekundu v průběhu posledních deseti sekund, s_5 je průměrný počet uskutečněných spojení za sekundu v průběhu posledních pěti sekund a s_1 je počet uskutečněných spojení za poslední sekundu. Nejvyšší prioritu dostane přiřazenou ten server, u kterého je výsledné číslo c největší.

7.4 Knihovna na datové vrstvě

Součástí Load Balanceru je knihovna *data_layer.so* s funkcí *ftp*. Tato funkce se nastaví jako podmodul na datové vrstvě během volání podmodulů na první až třetí vrstvě (typicky na první vrstvě). Jádro pak funkci volá v určených okamžicích průběhu spojení. Během volání funkce upravuje příchozí i odchozí data a otevírá příslušná pomocná datová spojení, aby protokol ftp fungoval v aktivním i pasivním režimu.

Dodatek A

Testy

A.1 Testy funkčnosti

Testy funkčnosti jsou určeny k otestování všech funkcí Load Balanceru a proto je stačí zkusit krátkodobě. Protože některé části, které nejsou nutné pro základní běh Load Balanceru, mohou běh zpomalovat nebo nemusí být dostatečně stabilní, je třeba vyzkoušet běh Load Balanceru bez těchto částí.

Minimální základ pro běh. Odkoušení této části je nejdůležitější a proto mu bylo věnováno nejvíce času. Z hlediska funkčnosti se jedná o testování nastavení portů pro poslouchání, nastavení serverů a samotné přepojování. Pro protokoly UDP a TCP je rozdílná implementace a obě varianty byly pečlivě testovány. Tato základní část je spolehlivá a stabilní a byla nejvíce testována i v dalších testech.

Management vláken je neoddelitelnou součástí základu jádra. Pokud si uživatel nepřeje, aby byla vlákna využívána, je možné nastavit příslušné parametry v konfiguračním souboru. Testování se zaměřovalo především na časté zvyšování a snižování počtu vláken a stabilitu běhu během těchto změn.

Autorestart Funkce která umožňuje automatickou detekci "zatuhnutí" jádra a statistik a jejich okamžitému automatickému restartování.

Knihovny pro výběr serveru jsou dynamicky linkované knihovny, které nejsou součástí jádra a uživatel má možnost přidávat si svoje vlastní knihovny. Z hlediska jádra se testovalo bezproblémové načítání knihoven, jejich zařazení za běhu a volání během rozhodování o výběru serveru.

Knihovny pro řízení spojení jsou dynamicky linkované knihovny, které nejsou součástí jádra a uživatel má možnost přidávat si svoje vlastní knihovny. Z hlediska jádra se testovalo bezproblémové načítání knihoven, jejich zařazení za běhu a jejich správné volání při provádění spojení. Jádro těmto knihovnám poskytuje prostředky na vytváření pomocných datových spojení, jejich management a zavírání. Dále se testovalo správné předávání dat těmto knihovnám.

Odesílání dat do statistik. Data se odesílají o většině událostí, které se v jádře dějí při průběhu spojení a zároveň jsou uvedena v konfiguračním souboru, že se mají odeslat. Testovány byly všechny kombinace odesílání dat.

Ping je samostatně běžící vlákno, které přímo spolupracuje s částí pro výběr serverů. Je implementován ping pro protokoly HTTP, MYSQL, FTP, POP, IMAP a SMTP.

Statické knihovny jádra jsou určeny pro podporu uživatelských dynamických knihoven. Dovolují uživatelským knihovnám využít některé funkce jádra. Bylo vyzkoušeno bezproblémové předávání parametrů a volání funkcí.

Uživatelské knihovny Součástí Load Balanceru je několik uživatelských knihoven, které obsahují základní algoritmy.

Dotazy do databáze Uživatelské knihovny mají možnost se dotazovat na data z databáze. Testováno bylo především navázání spojení se statistickým modulem a rychlým vyřízením požadavku.

Konfigurační knihovna zpracovává konfigurační soubor. Je odzkoušena na kontrolu syntaktických i sémantických chyb a na spolupráci s ostatními částmi Load Balanceru.

Ukládání dat přijatých od jádra statistickým modulem. Testováno správné ukládání dat jak do bufferu v paměti, tak i jejich agregace a ukládání do databáze.

Odpojení databáze Testováno chování statistického modulu po ztracení spojení s databází, kdy statistický modul začne data ukládat pouze do paměti.

Zpracování dotazů od modulu jádra a webového rozhraní.

Webové rozhraní je navázáno na statistický modul a konfigurační knihovnu. Je otestována komunikace se statistickým modulem a navázání na funkce konfigurační knihovny. Je testováno zaslání signálů modulu `init`.

A.2 Testy stability a zátěžové testy

Tyto testy byly prováděny v počítačové laboratoři labTS v Tróji MFF. K testování bylo použito deset počítačů. Jeden byl použit jako Load Balancer, obsahoval dvě síťová rozhraní, kde jedno bylo připojeno do podsítě klientů a druhé do sítě serverů. Klienty tvořily čtyři počítače, servery pět počítačů.

Konfigurace počítačů:

- procesor AMD Athlon XP 2500+
- paměť Kingston 512 MB
- pevný disk Maxtor, 80 GB
- síťové rozhraní 100Mbit/s

V této konfiguraci se nejdříve testovaly některé funkčnosti, které nešly otestovat dříve.

- Spuštění, restartování a ukončení Load Balanceru s umístěním různých modulů na různých počítačích.
- Testování výpadku spojení vytahováním kabelu a následné spojení u jednotlivých modulů, dodatečné doručení příkazu restartu modulům po výpadku spojení.
- Testování výpadku spojení se servery při žádné či minimální ztrátě spojení a následném zařazení serveru po obnově spojení.

Další testy byly provedeny se zaměřením na maximální dosažitelnou zátěž. Testovány byly protokoly ftp a http a testy byly provedeny opakovaně

Pro testování ftp byly zvoleny dva testy. Jeden na přenos velkých souborů a druhý na přenos velkého počtu malých souborů.

V prvním testu bylo opakovaně přeneseno několik souborů o velikostech 100-200MB. Při tomto testu byla dosažena stejná rychlost přenosu (11MB/s) jako při přímém propojení klienta se serverem.

V druhém testu bylo opakovaně přeneseno 10000 souborů o velikosti 500B. Přenos souborů probíhal poloviční rychlostí než u přímého propojení. To

je způsobeno protokolem ftp, který pro každý soubor otvírá nové spojení. Protože velikost souborů je zanedbatelná a při přenosu přes Load Balanceru je nutné otevřít dvojnásobný počet spojení, musí i čas být dvojnásobný.

Při testech protokolu http byl využit server apache a klient httpperf, který umožňuje generovat předem stanovený objem požadavků. Testy byly zaměřeny na maximální dosažitelný výkon a řádově trvaly v jednotkách minut. Dotazy probíhaly na dva soubory o velikosti 1.4KB. RR bude dále označovat request rate / sec (odchozí frekvence klienta) a T bude označovat délku testu v sec.

1. Přepojování z jednoho klienta mezi tři servery. RR=2500, T=200. Při testu byly přepojeny všechny spojení bez chyby.
2. Přepojování z jednoho klienta mezi tři servery s výpadky serverů. RR=2500, T=200. Při vytažení kabelu ze serveru došlo k výpadku jednotek spojení a poté se spojení rozložila mezi ostatní servery. Po zapojení serveru zpět do sítě byl opět zapojen do vyřizování požadavků.
3. Přepojování z jednoho klienta mezi tři servery s výpadkem všech serverů. RR=2500. Po zapojení alespoň jedno serveru a jeho detekci ping modulem okamžitý náběh všech spojení.
4. Přepojování z jednoho klienta mezi tři servery. RR=3500, T=400. Load Balancer byl schopný přepojovat 3050 +- 60 spojení / sec.
5. Přepojování z jednoho klienta mezi tři servery. RR=5000, T=400. Load Balancer byl schopný přepojovat 2200 +- 50 spojení / sec.
6. Přepojení požadavku http na soubor o velikosti 100MB. Přenos probíhal rychlostí 11MB/sec a odpovídal rychlosti přímého dotazu na server.

Následující dva testy byly zaměřeny na stabilitu Load Balanceru. Součty odchozích požadavků od klientů a příchozích počtů na servery nemusí souhlasit, protože mezi požadavky u serverů jsou započítávány i požadavky ping funkce.

1. První test probíhal ze čtyř klientů na pět serverů. Klienti 1 a 2 se dotazovali na port 11230 a klienti 3 a 4 na port 11231. Dotazy z portu 11230 se rozdělovaly mezi servery 1, 2 a 3. Dotazy z portu 11231 se rozdělovaly mezi servery 3, 4 a 5. Load Balancer požadavky rozděloval pomocí algoritmu static select first mezi pět serverů s těmito vahami:

- server 1 - 25

- server 2 - 275
- server 3 - 100
- server 4 - 100
- server 5 - 200

Každý klient odesílal požadavky rychlostí 250 požadavků / sec. Doručeny byly všechny požadavky.

Odeslané požadavky z klientů:

- klient 1 - 13 539 502
- klient 2 - 13 540 330
- klient 3 - 13 540 942
- klient 4 - 13 541 562

Přijaté požadavky serverů:

- server 1 - 1 931 842
- server 2 - 21 250 262
- server 3 - 7 728 141
- server 4 - 7 750 697
- server 5 - 15 501 394

2. Druhý test probíhal ze tří klientů na tři servery. Klienti 1 a 2 se dotazovali na port 11230 a klient 3 na port 11231. Dotazy z portu 11230 se rozdělovaly mezi servery 1 a 2. Dotazy z portu 11231 se rozdělovaly mezi servery 2 a 3. Load Balancer požadavky rozděloval pomocí algoritmu static select first mezi tři servery s těmito vahami:

- server 1 - 50
- server 2 - 50
- server 3 - 150

Klient 1 a 2 odeslal 10 mil. požadavků rychlostí 250 požadavků / sec. Klient 3 odeslal 20 mil. požadavků rychlostí 500 požadavků / sec. Na klientech byl nastaven timeout 5 sec. Doručeny byly všechny požadavky s odpovídajícím rozložením.

Přestože se několikahodinový test může zdát jako poměrně krátký, bylo při něm dosaženo několikanásobně vyššího počtu přepojených požadavků než nastává běžně v praxi. Na počet přeposlaných požadavků tak tyto testy odpovídají několika dnům běžného provozu běžných serverových farem.

A.3 Testy ve WAN

Cílem těchto testů bylo docílit možné reálné rozmístění jednotlivých částí. Z důvodů uvedených v úvodní kapitole byl celý Load Balancer umístěn na jedné lokální síti s přenosovou rychlostí 100Mbit / sec.

V jednotlivých konfiguracích byly opakovaně provedeny následující typy testů:

- Navázání a procházení, vytváření a mazání adresářů a souborů na ftp serveru.
- Přenos řádově stovek souborů o velikosti 1KB pomocí protokolu ftp.
- Přenos velkého souboru (velikost záležela na rychlosti nejpomalejšího spojení, aby byl test únosný a trval maximálně desítky minut) pomocí protokolu ftp.
- Přenos adresářové struktury s různě velkými soubory (opět aby celková doba testu nepřesáhla desítky minut) pomocí protokolu ftp.
- Dotazování serveru http protokolu na řádově stovky souborů o velikosti 1KB.
- Přenos velkého souboru (velikost záležela na rychlosti nejpomalejšího spojení, aby byl test únosný a trval maximálně desítky minut) pomocí protokolu http.

Pro simulaci reálných podmínek jsme využívali různé druhy spojení o různých rychlostech:

LAN Rozmístění v rámci jedné lokální sítě s přenosovou rychlostí 100Mbit / sec.

PASNET Pražská akademická WAN síť s přenosovou rychlostí 100/1000 Mbit / sec.

XDSL Připojení pomocí DSL firmy Nextra/TeleNor s přenosovou rychlostí 512Kbit / sec download a 128Kbit / sec upload.

Bezdrátové spojení - MW Připojení pomocí mikrovlnného spojení s přenosovou rychlostí 64Kbit / sec.

GPRS Připojení pomocí GSM telefonu s přenosovou rychlostí 57Kbit/sec download a 28Kbit/sec upload.

S výše uvedenými možnostmi konektivity jsme vyzkoušeli následující kombinace. Load Balancer - LB, Servery - S, Klienti - K:

- LB<->S - LAN, LB<->K - LAN
- LB<->S - LAN, LB<->K - PASNET
- LB<->S - LAN, LB<->K - XDSL
- LB<->S - LAN, LB<->K - MW
- LB<->S - LAN, LB<->K - GPRS
- LB<->S - PASNET, LB<->K - PASNET
- LB<->S - PASNET, LB<->K - XDSL
- LB<->S - PASNET, LB<->K - MW
- LB<->S - PASNET, LB<->K - GPRS
- LB<->S - XDSL, LB<->K - PASNET
- LB<->S - XDSL, LB<->K - XDSL
- LB<->S - XDSL, LB<->K - MW
- LB<->S - XDSL, LB<->K - GPRS

Během všech testů nedošlo k žádnému závažnějšímu výpadku. Při přehlcení spojení mezi servery a Load Balancerem došlo k vyřízení jen části požadavků. Přerušení spojení způsobené specifiky daného typu spojení (výpadky spojení GPRS) byly buď vyřízeny později pokud nevypršely timeouty daných protokolů nebo byly ztraceny a bylo ponecháno na vyšších vrstvách pokus o opětovné navázání spojení.

A.4 Praktický test

Praktický test byl proveden ve společnosti Czech OnLine. K testu nám byly zapůjčeny čtyři počítače. Jeden jsme využili jako Load Balancer s běžícími moduly initu, jádra a statistik. Zbylé tři počítače se využily jako klienti.

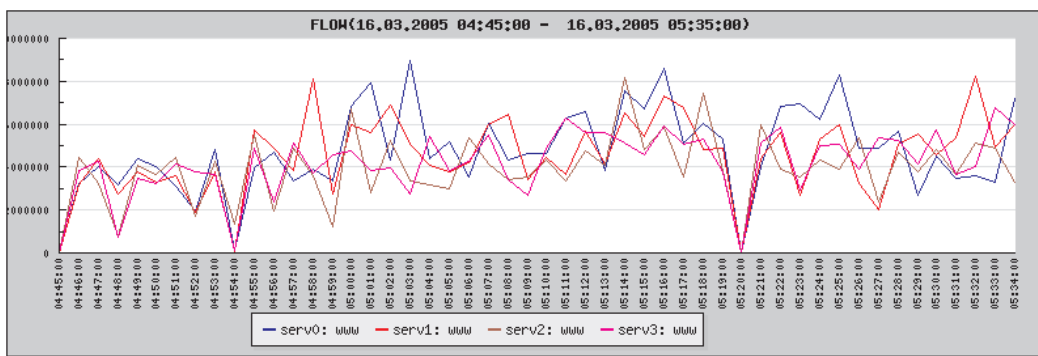
Pro testování nám byly zapůjčeny logy jedné hodiny reálné zátěže ze čtyř serverů ze špičky. Z těchto logů jsme získali informace o dotazech na jednotlivé servery. Tyto dotazy jsme pak přehrávali pomocí několika souběžně běžících klientů na každém klientském počítači. Počet simulovaných klientů

se pohyboval v rozmezí 50 až 250. Požadavky byly generovány programem siege, který je vybíral náhodně.

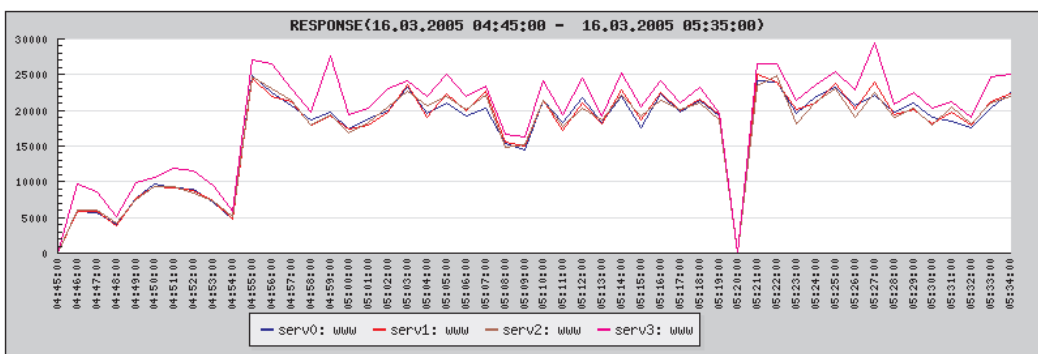
Aby nedošlo k přetížení serverů, byly testy prováděny v brzkých ranních hodinách. Z bezpečnostních důvodů se provedl postupný náběh trafficu, aby nedošlo k neočekávanému přetížení serverů. Po úvodní zavaděcí fázi se zvýšil traffic na maximální úroveň. Během testu byl použit statický algoritmus basic round robin, který traffic rozkládá rovnoměrně mezi všechny servery. Tento algoritmus jsme zvolili, protože je ze všech nejrychlejší a díky dostatečné dimenzovanosti a rovnoměrnosti výkonu serverů je plně dostačující.

Při testu byla dosahována přenosová rychlost na hranici možností síťového rozhraní (100Mbit / sec). Tím docházelo ke kolísání trafficu a k výpadku Load Balanceru, který byl zaviněn chybou v kernelu systému.

Průběh testu ilustrují následující grafy:



Obrázek A.1: Maximální datový tok / sec agregovaný v minutových intervalech



Obrázek A.2: Průměrná doba odezvy serveru agregovaná v minutových intervalech

Tento test prokázal schopnost Load Balanceru fungovat v reálném provozu i za podmínek zvýšené zátěže.