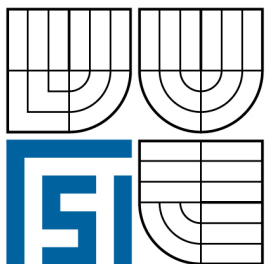


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE INFORMATIKY
FAKULTY OF MECHANICAL ENGINEERING
INSTITUT OF AUTOMATION AND COMPUTER SCIENCE

Návrh a implementace vysoce dostupné aplikace v unixovém prostředí pro "call back system"

Design and implementation of a call back system utilizing unix clustering

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE
AUTHOR

ING. JURAJ VAKULA

VEDOUcí PRÁCE
SUPERVISOR

ING. JAN ROUPEC, PH.D.

BRNO 2009

ZADÁNÍ ZÁVĚREČNÉ PRÁCE

(na místo tohoto listu vložte originál a nebo kopii zadání Vaší práce)

ABSTRAKT

Diplomová práca sa zaoberá využitím a uplatnením vysokej dostupnosti a spoľahlivosti systémov využívaných v praxi, kde hlavnou úlohou je zabezpečiť plynulý chod spoločnosti a jej aplikácií, ktoré využíva. Cieľom tejto práce je vytvoriť prostredie pre callback systém, ktorý pozostáva z rady softwarových produktov ako sú databáze, web, java a mnohé ďalšie. Každý z týchto produktov musí byť zabezpečený tak, aby sa za každých okolností (výpadok hardwaru, ľudskej chyby) služba nestala nedostupnou.

ABSTRACT

The thesis is focused on use and implement High Availability cluster in the productive environment, where is needed to concentrate on business critical application. The main aim of this thesis is to create the environment for callback system, which consist from many software products, for example database, web, java and etc. A company has to consolidate every application in an flexible environment resistant to hardware failure, human error in order to be running 24/7.

KĹÚČOVÉ SLOVÁ:

Vysokodopný cluster, Linux, Virtualizácia

KEYWORDS:

High Availability cluster, Linux, Virtualization

Pod'akovanie

Vyjadrujem pod'akovanie vedúcemu diplomovej práce Ing. Janovi Roupcovi, Ph.D., bez ktorého pomoci, rád a odborných pripomienok by táto práca nemohla vzniknúť. Taktiež by som rád pod'akoval Josephovi Jerzewskemu za možnosť realizácie tohto projektu.

Obsah

ABSTRAKT.....	6
ABSTRACT.....	6
POĎAKOVANIE.....	8
OBSAH.....	10
ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK.....	12
1. ÚVOD.....	16
.....	16
.....	17
2. CALLBACK SYSTÉM.....	18
2.1. CHARAKTERISTIKA CALLBACK SYSTÉMU.....	18
2.2. CALLBACK SLUŽBY.....	20
3. TECHNICKÁ ŠPECIFIKÁCIA TXTMAIL.....	21
4.1. HIGH AVAILABILITY CLUSTER.....	23
4.2. VÝPOČTOVÝ CLUSTER.....	23
4.3. CLUSTER S ROZLOŽENÍM ZÁŤAŽE.....	24
5. VIRTUALIZÁCIA.....	25
5.1. PLNÁ VIRTUALIZÁCIA.....	25
5.2. ČIASTOČNÁ VIRTUALIZÁCIA.....	25
5.3. PARAVIRTUALIZÁCIA.....	25
5.4. VIRTUALIZÁCIA NA ÚROVNI OPERAČNÉHO SYSTÉMU.....	26
5.5. APLIKAČNÁ VIRTUALIZÁCIA.....	26
5.6. EMULÁCIA.....	26
6. STORAGE AREA NETWORK.....	27
6.1. DRUHY SIETÍ.....	27
6.2. FC PROTOKOL	27
6.2.1. ZÁKLADNÝ KAMEŇ FIBRE SIETÍ.....	28
6.2.2. KOMUNIKÁCIA V RÁMCI FABRIC.....	28
6.3. LUN MASKING.....	28
6.4. ZDIELANIE MÉDIÍ.....	29
6.5. VIRTUALIZOVANÝ ÚLOŽNÝ PRIESTOR.....	29
7. MONITOROVACIE NÁSTROJE PRE LINUX.....	30
7.1. MONITOROVANIE HARDVÉROVÝCH SÚČASTÍ SYSTÉMU.....	30
7.2. MONITOROVANIE SIEŤOVÝCH AKTIVÍT SYSTÉMU.....	31
7.3. MONITOROVANIE INTEGRITY SÚBOROVÉHO SYSTÉMU.....	32
7.4. MONITOROVANIE SPUSTENÝCH PROCESOV.....	32
7.5. MONITOROVANIE SYSTÉMOVÝCH LOG SÚBOROV.....	32
7.6. MONITOROVANIE ÚTOKOV A POKUSOV O PRIENIK.....	33

8. TOMCAT.....	34
8.1 ŠTRUKTÚRA TOMCAT.....	35
9. POUŽITIE PROGRAMU RSYNC.....	36
.....	36
10. NÁVRH RIEŠENIA.....	37
11.1. CPU VIRTUALIZÁCIA.....	39
11.2. PAMÄŤOVÁ VIRTUALIZÁCIA.....	39
11.3. DISKOVÁ VIRTUALIZÁCIA.....	40
11.4. SIEŤOVÁ VIRTUALIZÁCIA.....	40
12.1. ARCHITEKTÚRA HEARTBEAT.....	41
12.2. INŠTALÁCIA A NASTAVENIE	43
12.3. INŠTALÁCIA MySQL.....	44
12.4. INŠTALÁCIA HEARTBEAT.....	46
12.5. IMPLEMENTÁCIA HEARTBEAT.....	46
12.6. KONFIGURÁCIA STONITH A CLUSTER RESOURCE.....	46
12.7. KONFIGURÁCIA RESOURCE.....	47
12.8. PROCES MIGRÁCIE NA INÝ UZOL CLUSTERU.....	47
12.9. RUČNÁ KONFIGURÁCIA CLUSTEROVANÝCH ZDROJOV.....	48
12.10. KONFIGURÁCIA A RIADENIE CLUSTEROVANÝCH ZDROJOV.....	49
12.11. VYTvorenie CLUSTEROVANÝCH ZDROJOV.....	49
12.12. DEFINÍCIA FAILOVER UZLOV.....	50
12.13. DEFINÍCIA ZDROJOV FAILBACK UZLOV.....	50
12.14. KONFIGURÁCIA MONITOROVANIA.....	51
12.15. KONFIGURÁCIA HEARTBEAT CLUSTER RESOURCE GROUP.....	53
12.16. KONFIGURÁCIA A HEARTBEAT KLONOVANIE ZDROJOV.....	55
12.17. MIGRÁCIA CLUSTROVANÝCH ZDROJOV.....	56
13.VYSVETLENIE PRÍKAZOV PRI PRÁCI S HEARTBEAT.....	60
14. ZÁVER.....	64

Zoznam použitých symbolov a skratiek

cluster

Cluster je skupina počítačov (skutočných alebo virtuálnych) zdieľajúcich rozloženie záťaže aplikácie a tým dosiahnuť výpočet v čo najkratšom čase. Vysokodostupný cluster je primárne určený na dosiahnutie najvyššej dostupnosti pre služby.

cluster partition

Ak zlyhá komunikácia medzi jedným, alebo viacerými uzlami a zvyškom clusteru nastane rozdelenie clusteru. Uzly v clusterovej časti sú stále aktívne a schopné komunikácie, ale nie sú schopné komunikovať so zvyškom nedostupných uzlov.

consensus cluster membership (CCM)

CCM určí, ktorý uzol vytvorí cluster a bude zdieľať tieto informácie so zvyškom clusteru. CCM modul beží na každom uzle clusteru.

cluster information base (CIB)

Reprezentuje konfiguráciu a stav celého clusteru (členstvo uzlov, zdrojov, obmedzení, atď.), ktoré sú zapísané v XML a uložené do pamäti. Primárny CIB vedie a udžiava záznamy na DC a replikuje ich na zvyšné uzly.

cluster resource manager (CRM)

Je hlavný riadiaci člen, ktorý zodpovedá za koordináciu všetkých nelokálnych interakcií. Každý uzol clusteru má svoju vlastnú CRM.

DAS

Direct Attach Storage. Úložný priestor priamo pripojený do serveru alebo počítača, bez použitia siete.

designated coordinator (DC)

Je uzol, kde sa nachádza práve kópia CIB. Všetky ostatné uzly clusteru obdržia informácie o ich konfigurácii a informácie o zdrojoch zo súčasného DC. DC je vybrané zo všetkých uzlov v clusteri po zmene vlastníctva.

Distributed replicated block device (drbd)

DRBD je blokové zariadenie vytvorené pre „high availability“ cluster. Celé blokové zariadenie je zrkadlené cez špeciálne vyhradenú sieť a vnímané ako sieťový RAID-1.

failover

Nastáva, ak zlyhajú zdroje, alebo uzol a afektované zdroje sú spustené na inom uzle.

FCP

Fibre channel protokol. Mapujúci s SCSI pomocou optického alebo metalického média.

fencing

Popisuje koncept prevencie prístupu na zdieľaný diskový zdroj „non-cluster“ členom. Táto situácia môže nastať, ak v prípade pádu príde k uzamknutiu zdrojov z uzla, ktorého členstvo a štatút je neistý. Taktiež sa rozlišuje medzi „fencingom“ uzla a zdroja.

HBA

Host Bus Adapter, pripojenie počítača k úložnému zariadeniu.

Heartbeat resource agent

Heartbeat resource agent vykonáva štart / stop operácie použitím skriptov uložených v */etc/ha.d/resource.d* alebo */etc/init.d*.

hot migrácia

Označuje sa tým premiestnenie (migrácia), kedy bežiaci virtuálny server je prenesený z jedného HW na iný bez nutnosti výpadku.

HW

Hardware, označuje fyzicky existujúce technické vybavenie počítača.

JSP

JavaServer Pages, produkt spoločnosti Apache, ktorý obsahuje zásuvné moduly pre Tomcat Apache.

LAN

Local area network, označuje počítačovú sieť, ktorá geograficky pokrýva malé územie (domácnosti, malé firmy).

LUN

Logical Unit Number, je číslo, ktoré je pridelené logickej jednotke.

LRM

local resource manager je zodpovedný za činnosť zdrojov. Používa „resource agent“ skripty na uskutočnenie požiadavky. LRM potrebuje na svoje fungovanie DC, ktorý mu povie, čo treba vykonať.

LSB resource agent

Sú štandardné LSB init skripty. LSB init skripty nie sú limitované použitím v kontexte vysokej dostupnosti. Linux systém využíva LSB init skripty na kontrolu služieb. LSB agent sa nachádza v */etc/init.d*

LVM

Logical volume manager je to metóda správy logických diskov.

node

Počítač (skutočný alebo virtuálny), ktorý je členom clusteru a je nepostrehnuteľný pre užívateľa.

OCF resource agent

„OCF resource agent“ je podobný funkcii LSB. OCF musí spúšťať, zastavovať a monitorovať štatút nastavenia. OCF je uložené v adresári */usr/lib/ocf/resource.d/provider*.

OS

Operačný systém.

pingd

Ping daemon. Pravidelne kontaktuje jeden, alebo viac serverov, mimo cluster pomocou ICMP pingu.

policy engine (PE)

Činnosti ktoré sú potrebné na implementáciu pravidiel zaznamenaných do CIB. Tieto informácie sú potom vykonané v nastavení clusteru. PE vždy beží na DC.

resource

Je typ služby, alebo aplikácie, ktorú využíva Heartbeat.

resource agent (RA)

RA je skript, ktorý sa správa ako proxy na riadenie zdrojov. Sú tri druhy resource agentov: OCF resource agent, LSB resource agent (štandardný LSB init script) a heartbeat resource agent.

SCSI

Small Computer System Interface je rozhranie a súbor príkazov pre výmenu dát medzi externým alebo interným počítačovým zariadením a počítačovou zbernicou.

Single Point of Failure (SPOF)

Je komponent clusteru, ktorý by mal predísť pádu celého clusteru.

split brain

Je riešenie v ktorom sú uzly clusteru rozdelené do dvoch, alebo viacerých skupín, ktoré o sebe nevedia. Zabrániť „split brain“ situácii, kedy by mohol nastať nechcený dopad na cluster, musí byť správne nakonfigurovaný STONITH ako resource.

STONITH

Komponent, ktorý vypnutím uzla v clustery zabráni problémom ktoré by vznikli, ak sa cluster nespráva podľa definovaných požiadaviek.

TCP/IP

Rodina protokolov TCP/IP. Obsahuje súbor protokolov pre komunikáciu v počítačovej sieti, označovaná ako hlavný protokol celosvetovej siete internet.

Transition engine (TE)

Cluter preberá príkazy z PE a uskutočňuje ich. TE beží na DC uzle. Odtiaľ potom dáva inštrukcie lokálnemu resource managerovi na ostatných uzloch, ktoré vykonávajú dané úlohy.

Virtualizácia

Označuje v počítačovom prostredí postupy a techniky, ktoré umožňujú k dostupným zdrojom pristupovať iným spôsobom než akým fyzicky existujú.

1. Úvod

Potreba vysokej dostupnosti a spoľahlivosti aplikácií využívaných čoraz väčším počtom malých a aj veľkých spoločností vedie k tvorbe a potrebám nasadzovania nových technológií v praxi pre svojich zákazníkov. Dalším a podstatným faktorom sú čo najnižšie výdaje týkajúce sa oblasti informačných technológií a telekomunikácií. So zaujímavým nápadom ako šetriť náklady prišla firma TxTmail. Vytvorila Callback systém, ktorý využíva dnes veľmi rozšírený spôsob telefonovania cez internet - VOIP.

Pri využívaní callback systému je nosným pilierom technológia, na ktorej je tento systém vybudovaný. Tu je veľmi dôležité navrhnuť a vytvoriť prostredie, ktoré zabezpečí funkčnosť systému ako celku, a to aj v prípade neočakávaného výpadku, prípadne plánovaných aktualizácií.

Hlavnou úlohou je zabezpečiť plynulý chod kritickej databázy, zdieľaných súborov, web a business aplikácií. Ak nastanú neočakávané chyby, dôjde k prebraniu týchto zdrojov iným serverom alebo servermi v prostredí. Pri hľadaní riešenia využijeme nasadenie „high available cluster“ s vysokou dostupnosťou, ktorý dokáže eliminovať výpadok hardwaru, chybu spôsobenú ľudským faktorom, výpadok pri aktualizácii systému, atď.

Dnes na šetrenie nákladov je bežné využiť virtualizáciu, kedy namiesto množstva serverov sa využijú len dve, prípadne tri virtualizačné platformy. Tento krok pomôže ušetriť počiatočné investície, prevádzkové náklady, zníži veľkosť realizačného tímu.

V tejto práci chcem navrhnuť systém, v ktorom budú odstránené vyššie uvedené problémy. Prvoradou úlohou je správne aplikovať a definovať vysoko dostupné prostredie, pri ktorom bude využitá virtualizácia a spojím tak riešenie dvoch najpodstatnejších problémov – dostupnosť aplikácie a zníženie nákladov.

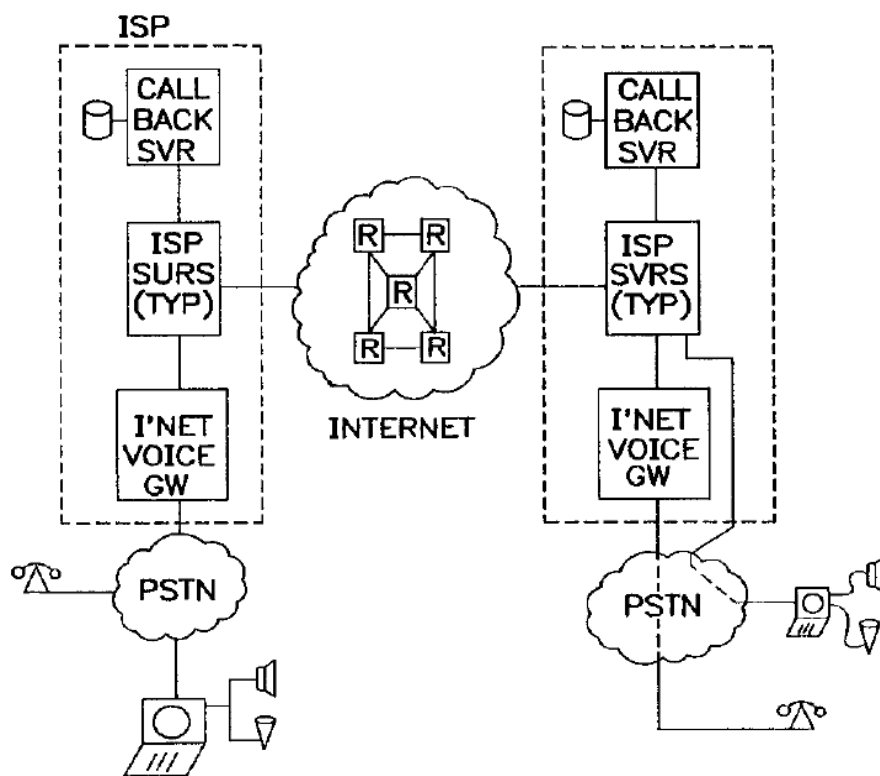
2. Callback systém

2.1. Charakteristika callback systému

Charakteristika tohto systému by sa dala jednoducho zhrnúť do nasledujúceho popisu. Je to systém implementácie pre telefonické hovory uskutočnené pomocou internetového telefonického spätného volania. Volajúca a volaná stanica sú pripojené do systému cez miestneho a vzdialeného poskytovateľa služieb (poskytovateľa vrátane hlasovej brány a spätné volajúceho serveru).

Telefónny hovor vytvorený volajúcou stanicou prostredníctvom volajúcej strany je spracovaná miestnym poskytovateľom služieb a miestnym spätné volajúcim „callback“ serverom, ktorý vygeneruje kontrolnú správu a tá spája telefónny hovor. Vzdialený poskytovateľ služby spracuje a spojí telefónny hovor, ak je prijatý volanou stranou a ak bola zaslaná správa na zmazanie kontrolnej správy z miestneho „callback“ servera. Ak je však telefónny hovor neprijatý (nezodpovedaný), vzdialený „callback“ server uloží kontrolnú správu a počká pokiaľ volaná strana nezaháji spiatočný hovor.

V prípade úspešného spojenia, ktoré bolo vytvorené, vzdialený „callback“ server pošle správu miestnemu „callback“ serveru na poskytnutie spätného hovoru volajúcej strane. Môže však vzniknúť situácia, že volajúca strana je medzičasom nedostupná, alebo prebieha na nej iný hovor. Úlohou miestneho a vzdialeného servera je vyhodnotiť situáciu a označiť požiadavku ako rezervovanú. Vzdialený (predtým miestny) „callback“ server uloží správu a vytvorí spätnú odpoveď, kedy volaná strana (predtým volajúca) vygeneruje správu. Na obrázku 1 je načrtnutá schéma callback systému.



Obr. 1: Callback systém.

2.2. Callback služby

Callback služby umožňujú telefónnym užívateľom uskutočniť hovor na veľkú vzdialenosť a to zavolaním na callback server, na ktorom sa overí užívateľský aktivačný kód a prepojí volajúceho spolu s cieľovou stranou:

- uskutočnenie hovoru zavolaním na určené číslo (nie je spoplatnené, pretože server, ktorý prijíma požiadavku odmietne hovor)
- ozve sa obsadzovací tón a zákazník ukončí hovor
- systém zavolá späť do 5 sekúnd
- systém sa spýta na overenie identifikačného čísla
- systém si vyžiada cieľové číslo
- systém vytvorí hovor, ktorého priebeh sa uskutoční metódou VOIP

3. Technická špecifikácia TxTmail

TxTmail bol vytvorený tak, aby vyžadoval čo najmenej možných závislostí a zároveň aby používal lacné komponenty a open source komponenty, ktoré umožnia čo najjednoduchší a najflexibilnejší vývoj. TxTmail sa snaží používať posledné verzie pre každý komponent.

Mobilný klient komunikuje so serverom na porte 8080. Server pozostáva z rozličných servletov, z ktorých každý poskytuje špecifickú funkciu ako napríklad autorizáciu, inbox (e-mail), multimedialny inbox. Servlety sa rozhodujú či správa bude doručená jedným z dvoch možností - užívateľovi, ktorý je práve prihlásený do siete, alebo užívateľovi s momentálne vypnutým zariadením, pomocou SMSC. Servlet komunikuje s SMSC pomocou HTTP.

Teoreticky je možné použiť akúkoľvek konfiguráciu vyššie spomenutých „elementov“, a úspešne rozmiestniť servlety. Avšak pri tejto operácii môžu byť potrebné jemné úpravy ako napríklad prekompilovanie servletov pre danú verziu bežiacu na serveri.

V ideálnom prípade pre rýchle rozostavenie by sa nainštalovali aktuálne verzie prvkov na cieľový server. Význam verzie operačného systému nie je podstatný. Ako Java aplikačný server môže slúžiť Tomcat, ale existujú aj iné Java aplikačné servery, ktoré by poskytli vylepšený výkon. Ako databáza môže byť použitá akákoľvek relačná databáza. S vysvetlením pojmu Tomcat sa stretneme v kapitole 8.

Zahrnutie do existujúceho registračného a účtovného systému je prehľadné. Pre jednoduchosť a jedinečnosť sa používa telefónne číslo ako užívateľské meno pre aplikáciu (na zaistenie príjmu a odoslania správy). V súčasnosti TxTmail používa na registráciu skript napísaný v pythone. Pravdepodobne bude potrebné upraviť aktuálnu databázovú štruktúru na to, aby sa mohla podporovať existujúca databáza užívateľov. Na účtovanie bude pridaná ďalšia funkcia do servletu na autorizáciu a to preto, aby sa zjednodušilo posielanie požiadaviek cez HTTP do účtovacieho systému.

4. Počítačový clustering

Počítačový cluster je zoskupenie voľne viazaných počítačov, ktoré spolu úzko spolupracujú, takže navonok môžu pracovať ako jeden počítač. Clustery sú obvykle nasadzované pre zvýšenie výpočtovej rýchlosti, alebo spoľahlivosti s väčšou efektivitou akú by poskytoval jeden počítač. Pričom sú lacnejšie ako jediný počítač o zrovnateľnej rýchlosti alebo spoľahlivosti.

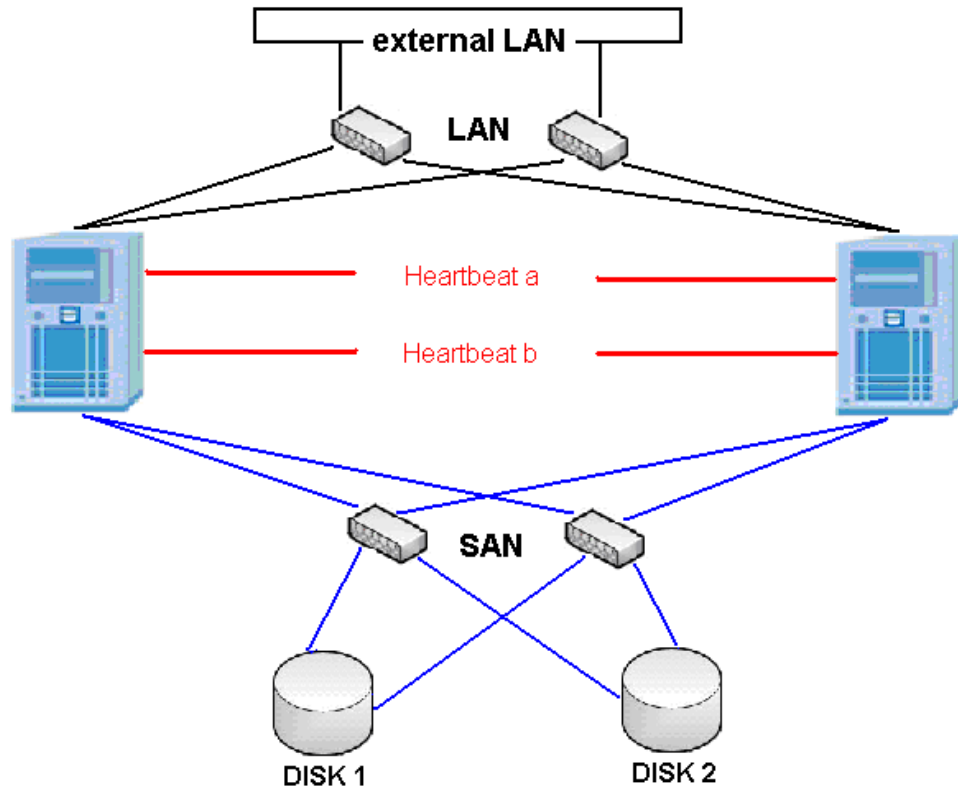
Vysoká dostupnosť aplikácií v dnešnej dobe je jedným z hlavných prvkov predovšetkým tam, kde je potrebné zaistiť prevádzku systémov 24 hodín denne bez, alebo s minimálnou nedostupnosťou služby. Pre zabezpečenie neprerušiteľnosti funkcie tohto systému je potrebné spojiť dva a viac serverov do tzv. Clusteru, ktorý zaistí, aby v prípade zlyhania jedného serveru sa aplikácia presunula na druhý server, alebo sa rovnomerne rozložila na viacej serverov v príslušnom clustery tak, aby užívateľ používajúci aplikáciu nič nespoznal. Tieto jednotlivé servery spojené do jedného clusteru sa nazývajú nody clusteru. V prípade, že príde k oprave serveru a tento je znova pripojený do infraštruktúry clusteru, môže tento automaticky opäť prevziať svoje aplikácie späť, to však závisí na type clusteru a nami zvolenej stratégii.

Clustrovanie rozdeľujeme:

- cluster s vysokou dostupnosťou (HA – high availability)
- cluster s rozložením záťaže (Load Balancing)
- cluster s vysokou výkonnosťou výpočtu (HPC – High Performance Computing)

4.1. High availability cluster

High availability cluster je cluster, ktorý zabezpečuje práve vysokú dostupnosť aplikácií a služieb. Pri tomto spôsobe je využitý „Heartbeat“ čo sú dva uzly clusteru, kde na každom z nich, bežia nami zvolené služby. Oba nody clusteru sa cez heartbeat pripojenie v pravidelných intervaloch sledujú a v prípade, že jeden z nich do stanovenej doby neodpovie, alebo sám vyšle správu o zmene svojho stavu, druhý prevezme jeho služby a aplikácie a následne aj jeho IP adresu. Doba výpadku potom zodpovedá súčtu doby nutnej k zaisteniu výpadku druhého uzlu, maximálny interval heartbeatov + timeout pre odpoveď, do doby potrebnej k naštartovaniu služieb potrebných pre beh aplikácií havarovaného serveru. Na obrázku 2 je schéma vysokodostupného clusteru s použitím heartbeatu medzi dvoma uzlami a pripojenie do externej siete.



Obr. 2: High availability cluster.

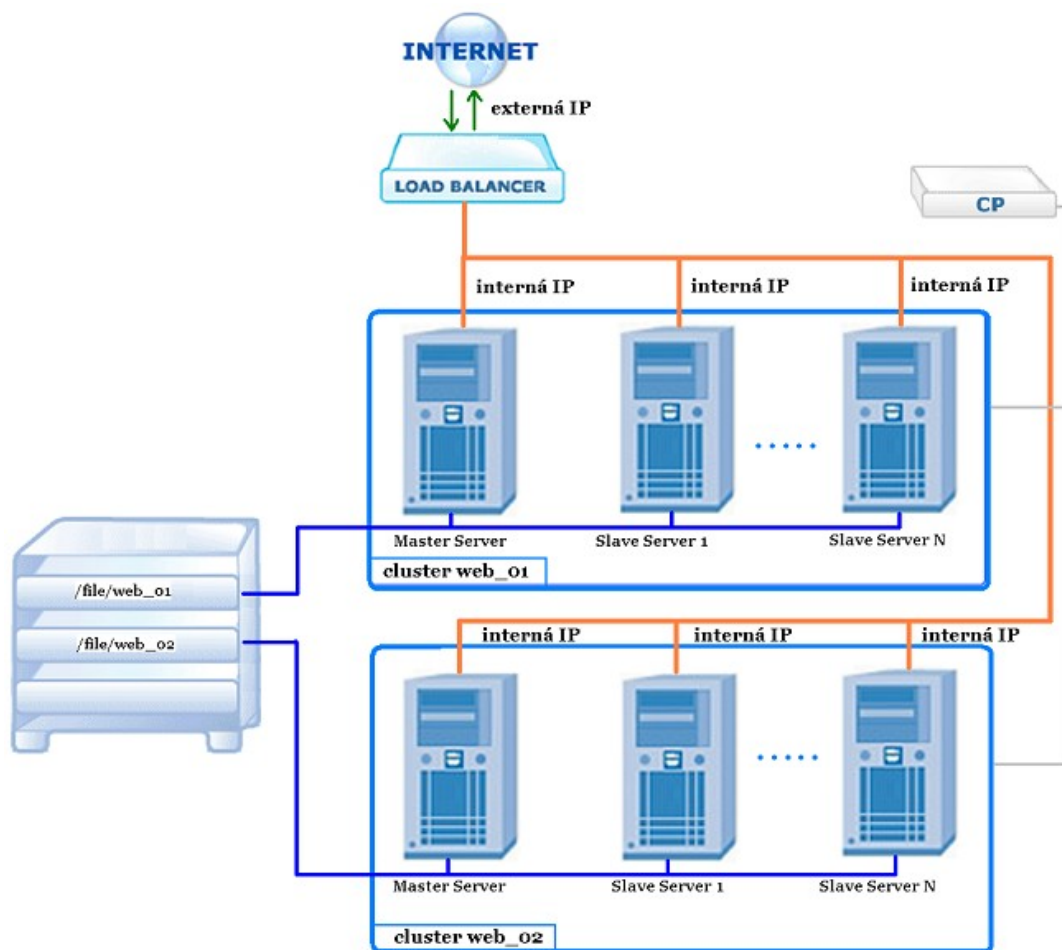
4.2. Výpočtový cluster

Výpočtový cluster (high – performance computing, HPC) slúži k zvýšeniu výpočtového výkonu viacerých počítačov, ktoré na tomto výpočte spolupracujú. Často sú použité počítače strednej cenovej hladiny prepojené vysokorýchlostnou počítačovou sieťou. Týmto

spôsobom vznikne vysokovýkonný celok, ktorý je mnohonásobne lacnejší ako jeden vysokovýkonný počítač. Príkladom a najznámejším clusterom tejto kategórie sú OpenMosix Cluster, Beowulf Cluster, Parallel Virtual Machine (PVD).

4.3. Cluster s rozložením zát'aže

Cluster s rozložením zát'aže (load ballancing, alebo scallable) zvyšuje možnú mieru zát'aže tým, že službu poskytuje niekoľko počítačov, ktoré majú rovnaký obsah (služba je poskytovaná paralelne). Rovnaký obsah je zaistený replikáciou obsahu medzi všetky prepojené počítače, alebo pomocou existujúceho špeciálneho, centrálného úložiska dát. Na obrázku 3 môžeme vidieť schému clusteru s rozloženou zát'ažou.



Obr. 3: Load balancing cluster.

Clustrovacie technológie hlbšie rozoberá Karl Kopper vo svojej knihe The Linux Enterprise Cluster [8].

5. Virtualizácia

Virtualizácia bude v tejto diplomovej práci veľmi často používaným pojmom a to z dôvodu nasadenia väčšieho počtu operačných systémov. Je veľmi dobrou pomôckou pre konsolidáciu serverov a prehľad celého prostredia, kedy administrátor má všetko poruke a taktiež okamžite. Ďalšou výhodou je čas strávený opravou hardwaru v prípade nožnej poruchy, kde dochádza k nedostupnosti potrebnej aplikácie. V neposlednom rade môžeme povedať, že virtualizácia šetrí náklady a nepriamo životné prostredie

Virtualizačné platformy:

- Plná virtualizácia
- Čiastočná virtualizácia
- Paravirtualizácia
- Virtualizácia na úrovni operačného systému
- Aplikčná virtualizácia
- Emulácia

5.1. Plná virtualizácia

Virtuálna platforma simuluje dostatočné množstvo hardwaru tak, aby umožnil oddelený beh neupraveného operačného systému (hosta) určeného pre rovnaký druh CPU. Najčastejšie je možný súčasný beh inštancií. Tento prístup objavil v roku 1966 u systému **CP-40** (predchodca VM od IBM). Príklady aplikácie podporujúce plnú virtualizáciu: **VirtualBox**, **Virtual PC**, **Vmware Workstation**, **Vmware Server**, **ESX server**, **QEMU**, **Mac-On-Linux**, **Win4BSD**, **z/VM**.

5.2. Čiastočná virtualizácia

Virtuálna platforma simuluje viac inštancií (nie však všetky) viacerým prostrediam hardwaru, na ktorom beží hostiteľ, predovšetkým adresovaného priestoru. Také prostredie podporuje zdieľanie zdrojov a izoláciu procesov, ale nedokáže oddeliť inštalácie hosťovaných operačných systémov. V tomto prípade však nie je možné hovoriť o virtuálnej platforme. Bol použitý v systéme **CTSS**. Väčšina ďalších systémov ako Microsoft Windows, alebo Linux a systémy z nižšie uvedených kategórií používajú túto techniku.

5.3. Paravirtualizácia

Virtuálna platforma nemusí nevyhnutne simulovať hardware, ale namiesto toho ponúka zvláštnu **API**, ktorá môže byť použitá len z upraveného operačného systému. Toto systémové volanie „hypervizor“ sa nazýva „hypercall“ v **Xen**, **Parallel Workstations**, **Enomalism**. Volanie je implementované hardwarovou inštrukciou DIAG (diagnose) v **CMS** od IBM. Príklady zahŕňajú **Win4lin 9x**, **logické domény zóny** od Sun a **z/VM**.

5.4. Virtualizácia na úrovni operačného systému

Virtualizuje sa fyzický server na úrovni operačného systému, čo umožňuje beh na viac izolovaných bezpečných virtuálnych serverov na jednom fyzickom stroji. Prostredie hostovaného operačného systému zdieľa jeden operačný systém s hostiteľským systémom – t.j. rovnaké jadro operačného systému je použité pre implementáciu hostovaného operačného systému. Aplikácie bežiacie v hostovanom prostredí ho však vnímajú ako samostatný systém. Medzi príklady patria **Linux-Vserver**, **Virtuozzo** (pre Linux alebo Windows), **OpenVZ**, **kontajnery Solarisu** a **FreeBSD Jail**.

5.5. Aplikačná virtualizácia

Serverové aplikácie bežia na danom stroji, používajú miestne zdroje, ale bežia vo zvláštnom virtuálnom stroji. To je rozdiel oproti tradičnému lokálnemu behu natívnych aplikácií, t.j. softwaru nainštalovanom priamo na systém. Takáto aplikácia beží v malom virtuálnom prostredí obsahujúcom komponenty nutné pre spustenie – napríklad položky registrov, súbory, premenné prostredie, prvky užívateľského rozhrania a globálne objekty. Toto virtuálne prostredie sa správa ako vrstva medzi aplikáciou a operačným systémom, ktorá zabráňuje konfliktom medzi operačným systémom a aplikáciou, alebo vzájomne medzi aplikáciami. Príklad zahŕňajú Java **Virtual Machine** od Sun **Softicity**, **Citrix**, **Vmware**, **Altiris**, **Portable Apps**, **Trigence**. Tento druh virtualizácie je zreteľne odlišný od všetkých predošlých. Delí ich len tenká hranica od prostredia virtuálnych prostredí ako je **Smalltalk**, **Forth**, **Tcl**.

5.6. Emulácia

Virtuálnu platformu simuluje celý hardware, dovoľuje beh neupraveného operačného systému host'a na celkom odlišnom procesore. Tento prístup je dlho používaný za účelom tvorby softwaru pre procesory, ktoré nie sú fyzicky dostupné. Príklady zahŕňajú **Bochs**, **PearPC**, Microsoft **Virtual PC** pre **PowerPC** (z osobnej skúsenosti táto emulácia nefunguje), **QEMU** bez akcelerácie a emulátor **Hercula**. Emulácia je implementovaná širokou škálou techník od **stavových automatov** až pod **dynamickú rekompiláciu** na **plne virtualizovaných** platformách.

S bližšími a podrobnými detailmi virtualizácie je možné sa zoznámiť [14].

6. Storage Area Network

Storage area network (zkratka SAN) je dedikovaná dátová sieť, ktorá slúži na pripojenie externých zariadení k serverom (diskové polia, páskové knižnice, zálohovacie zariadenia). SAN vznikla kvôli narastajúcim potrebám na zabezpečenie konsolidácie dát. Postupne však SAN začal nahrádzať DAS technológie, tým sa ukázali výhody SAN oproti DAS:

- fyzické oddelenie dát od serverov
- zdieľanie zdrojov (diskové zariadenia, zálohovacie zariadenia)
- vyššia dostupnosť
- redundantné cesty k zdrojom
- podpora clusterových riešení a architektúry
- podpora pre disaster recovery sites

6.1. Druhy sietí

Bežné SAN siete využívajú Fibre Channel protokol (SCSI), kde sa ako médium používa optický kábel.

- Fibre channel protokol (FCP), kde pomocou optického alebo metalického média sa mapujú SCSI s prenosovou rýchlosťou 1 Gbit/s, 2 Gbit/s, 4 Gbit/s, 8 Gbit/s, 10 Gbit/s
- FICON mapovanie pomocou Fibre Channel (mainframe servery)
- iSCSI, iFC, mapujú SCSI alebo FC pomocou TCP/IP
- HyperSCSI, mapujúce SCSI pre ethernet
- ATA pomocou Ethernet, mapovanie ATA cez Ethernet
- SCSI alebo TCP/IP mapovanie cez InfiBand

6.2. FC protokol

Fibre channel protokol definuje 3 rôzne topológie:

- Point to Point
- Arbitrated Loop
- Fabric

Samotná SAN sieť využíva hlavne Fabric topológiu. S ostatnými je možné sa stretnúť napríklad v koncových zariadeniach na prepojenie jednotlivých komponentov. V praxi sa Arbitrated Loops používa v niektorých diskových poliach pre prepojenie radičov a jednotlivých expanzií s diskami.

6.2.1. Základný kameň Fibre sietí

- Host Bus Adapter (HBA) – špeciálna karta, ktorá umožňuje jednotlivé servery pripojiť do SAN. Každá HBA (a ktorýkoľvek iný uzol v SAN) má unikátnu 64-bitovú WWN adresu (analógia ethernetovej MAC adresy), ktorá ju jednoznačne identifikuje v rámci SAN. Adresa sa skladá z prefixu výrobcu a samostatného čísla karty.
- Fabric Switch – zariadenie, ktoré podporuje jednotlivé prvky v rámci fabric SAN a zaisťuje ich komunikáciu (analógia ethernet switcha). Uzlom môže byť buď nejaký server (HBA), koncové zariadenie alebo iný FC switch. Každý switch obsahuje určitý počet portov pre pripojenie týchto zariadení. Vyrábajú sa ako malé 8 alebo 16 portové switche, rovnako ako aj veľké obsahujúce 256. Počet zariadení v celej fabric je limitovaný na 65535 prvkov.

Koncovým zariadením sú väčšinou diskové polia alebo iné zariadenia pre ukladanie dát, archiváciu alebo zálohovanie.

6.2.2. Komunikácia v rámci Fabric

Jeden z najdôležitejších mechanizmov v rámci fabric je tzv. zóning (analógia v sieti Ethernet je protokol 802.1q - VLANy), pomocou ktorého môže správca SAN vydefinovať zóny zariadenia, ktoré spolu môžu komunikovať. Napríklad, ak má infraštruktúra diskové pole, páskovú knižnicu, 5 serverov a jeden zálohovací server, môže byť zónovanie navrhnuté tak, aby všetky servery boli schopné komunikácie s diskovým poľom a len zálohovací server bude mať nadefinovanú zónu pre ďalšiu komunikáciu s páskovou knižnicou. Zóning je propagovaný v celej SAN a stačí len vydefinovať príslušnú zónu na ktoromkoľvek fabric switchi. Používajú sa dva spôsoby:

- definícia zón nad jednotlivými fyzickými portami switchov
- definícia zón nad WWN adresami

V praxi je lepšie používať zóny nad WWN, tým predídeme problémom s komunikáciou pri prepojení zariadenia z jedného portu do iného. Port zóning sa používal skôr, keď WWN zóning ešte nebol podporovaný. Použitie zóningu je vo väčšine prípadov nutné.

6.3. LUN masking

LUN masking zaisťuje ďalšiu granularitu SAN. Zóning zaisťuje rozdelenie fabric na nódy, ktoré spolu môžu komunikovať. To však pre väčšinu zariadení nie je dostatočné. Pokiaľ máme diskové pole s dvoma RAID kontrolermi, zóningom len definujeme, ktoré servery s týmito kontrolermi môžu komunikovať. Na diskovom poli môžu byť definované desiatky a desiatky logických diskov a aj tie je potrebné priradiť konkrétnemu serveru. Preto sa na diskových poliach definuje tzv. mapovanie. Každý virtuálny disk dostane priradené tzv. Logical Unit Number (LUN) a ten sa potom mapuje na konkrétnu WWN adresu v SAN. Kombinácia zóningu a mapovania zaisťuje, že všetky servery vidia diskové pole, ale každý server vidí len disky, ktoré mu patria.

6.4. Zdieľanie médií

V rámci SAN je možné zdieľať jednotlivé zariadenia, či sú to napríklad virtuálne logické disky, alebo páskové knižnice. To je potrebné pri budovaní clusterov, filesystémov na ktorých sa vyžaduje paralelná operácia alebo LAN-less, či Server-less zálohovanie, kedy sa dáta zálohujú cez SAN sieť a významne sa tým šetrí prevádzka na LAN sieti.

SAN vlastne umožňuje spájanie skladových ostrovov použitím vysokorýchlostnej siete. Operačný systém vidí zariadenie v SAN ako lokálne pripojené zariadenie. V prípade logických diskov vypublikovaných na diskovom poli, vidí radu lokálnych diskov, na ktorých môže byť nakonfigurovaný niektorý z filesystémov, ktorý daný operačný systém podporuje, alebo niektorý z diskov môže byť priamo použitý napríklad databázovým softwarom (Oracle DB na raw volumách, alebo ASM). Jeden z problémov prístupu k dátam v SAN je ten, že SAN rieši zdieľanie logických diskov medzi jednotlivými nódmi, operačný systém väčšinou nie. To hlavne preto lebo súčasný design väčšiny filesystémov neumožňuje paralelné prístupy. Pokiaľ je táto funkcionality požadovaná, je riešenie pokročilými súborovými systémami ako napríklad GPFS (Linux, AIX), VMFS (VmWare). Navzdory týmto problémom SAN pomáha k lepšej konsolidácii dát a tiež k ich efektívnejšiemu a rýchlejšiemu prístupu ako je to pri použití technológie NAS.

6.5. Virtualizovaný úložný priestor

Pre ukladanie diskov virtuálnych serverov používa ESX špeciálny filesystém VMFS, ktorý ma oproti iným filesystémom určité špecifiká. Je navrhnutý tak, aby bola minimalizovaná réžia pri správe objektov filesystému a optimalizovaný prístup z jednotlivých virtuálnych serverov. Hlavnou výhodou je možnosť paralelného prístupu z viacerých operačných systémov a tým aj možnosť tzv. hot migrácie.

7. Monitorovacie nástroje pre Linux

Pre neustále zlepšovanie a optimalizovanie zabezpečenia systému je nutné vykonávať komplexné monitorovanie aktivít a javov, či už sieťových, systémových alebo hardvérových. Dodržiavanie základných princípov teórie bezpečnostných bariér síce pomáha pri zvyšovaní miery zabezpečenia systému, no intenzívny pokus o útok môžeme včas zastaviť len v prípade, že o ňom vieme. Sledovať možno akúkoľvek časť systému, a v každom systéme sú požiadavky individuálne. V tejto časti sa zameriame na oboznámenie pojmu monitorovanie počítačového systému. Monitoring je možné rozdeliť na jednotlivé časti.

Monitorovanie počítačového systému:

- monitorovanie hardvérových súčastí systému
- monitorovanie vytťaženia CPU
- monitorovanie záťaže sieťových rozhraní
- monitorovanie využitia pamäte RAM
- monitorovanie využitia kapacity pevných diskov
- monitorovanie teplôt jednotlivých hardvérových súčastí
- monitorovanie softvérových súčastí systému
- monitorovanie sieťových aktivít systému
- monitorovanie integrity súborového systému
- monitorovanie spustených procesov
- monitorovanie systémových log súborov
- monitorovanie útokov a pokusov o prienik

Toto rozdelenie určite nie je úplné a hranice medzi jeho jednotlivými časťami nie sú vždy jasné. Už v úvode som spomínal, že každý systém je individuálny, a tak k nemu treba pristupovať aj v oblasti monitorovania. Na niektorých počítačoch je smerodajné vytťaženie CPU, na iných zas využitie kapacít sieťových rozhraní a dokonca sa nájdu aj systémy, na ktorých je nutné monitorovať „exotickejšie“ prvky, ako napríklad počet prístupov používateľov za určité časové obdobie. V unixových systémoch je implementácia aj takýchto nevšedných požiadaviek možná a ľahko realizovateľná, pretože sú pre ne voľne dostupné kvalitné a vysoko konfigurovateľné nástroje.

7.1. Monitorovanie hardvérových súčastí systému

Pri monitorovaní hardvérových súčastí systému sa často využíva aplikačný protokol SNMP (Simple Network Management Protocol), ktorý je primárne určený na monitorovanie stavu zariadení pripojených k počítačovej sieti. Podstata komunikácie prostredníctvom tohto protokolu je veľmi jednoduchá. Klient (subagent) si vyžiada od servera (master agent) informácie o nejakom konkrétnom objekte, ktorý server pozná a vie o ňom poskytnúť informácie (napríklad vytťaženie CPU). Server zašle klientovi v odpovedi aktuálnu hodnotu prislúchajúcu danému objektu. Už pri návrhu tohto protokolu sa počítalo s nutnosťou jeho budúceho rozširovania o podporu nových objektov, a preto je v ňom

implementovaný špeciálny typ databázy zvanej MIB (Management Information Base). Ak chce výrobca hardvéru pre svoj produkt zaviesť podporu do rôznych agentov alebo subagentov, stačí ak vydá rozšírenie MIB. Napriek tomu, že protokol SNMP už nie je práve najmladší, používa sa vďaka kvalitnému návrhu dodnes. Výraznými zmenami prešiel akurát spôsob autentifikácie používateľov, následkom čoho sú dnes známe tri verzie tohto protokolu SNMPv1, SNMPv2c a SNMPv3. Protokol SNMP sa dá využiť nielen na získavanie informácií o jednotlivých definovaných objektoch, ale aj na nastavovanie konkrétnych hodnôt pre tieto objekty. Tak môže administrátor napríklad vzdialene spravovať smerovaciu tabuľku alebo ARP záznamy na aktívnych sieťových prvkoch. Výhodou oproti webovým a telnetovým správcovským rozhraniám je možnosť správu protokolom SNMP skriptovať a teda kompletne automatizovať.

K protokolu SNMP existujú aj alternatívne technológie, ktoré sa snažia vytvoriť podobné rozhranie pre získavanie informácií o systéme. Sú to však väčšinou len pokusy, ktorým chýba sieťový model, preto dokážu pracovať iba na konkrétnom hoste. To znemožňuje ich nasadenie vo väčších sieťach, kde sa na monitorovanie systémov vyhradzuje osobitný počítačový systém získavajúci údaje prostredníctvom siete. Pri popise možností monitorovania hardvérových súčastí systému však nesmie byť vynechaný projekt `lm_sensors`, ktorý zavádza do linuxového jadra podporu pre zariadenia umiestnené na zberniciach I2C (Inter-Integrated Circuit) a SMBus System.

Management Bus slúžiacich v osobných počítačoch najmä na sprostredkovanie informácií z diagnostických senzorov ako je napríklad snímač teploty CPU. Pre serverové systémy je prítomnosť takéhoto softvéru nevyhnutnosťou, pretože znalosť aktuálneho „zdravotného stavu“ počítača pomáha ľahšie identifikovať aktuálne resp. predpovedať budúce problémy.

7.2. Monitorovanie sieťových aktivít systému

Samotný proces monitorovania sieťových aktivít systému by sa dal rozdeliť na monitorovanie využitia sieťových kapacít a na monitorovanie aktuálnych spojení a otvorených sieťových portov. Na monitorovanie využitia sieťových kapacít alebo inak povedané vyťaženia dostupnej linky sa používajú dva druhy nástrojov, ktoré majú mierne odlišný princíp činnosti. Nástroje z prvej skupiny využívajú na získanie aktuálnych hodnôt protokol SNMP. Zozbierané dáta následne do grafickej podoby spracováva nejaký vizualizačný softvér. Nástroje z druhej skupiny sú väčšinou založené na knižnici `libpcap`, ktorá je vo veľkej miere využívaná napríklad analyzátormi sieťových protokolov. Tieto produkty prepnú sieťové rozhranie do promiskuitného režimu a analyzujú všetky prichádzajúce pakety. Ich výhodou oproti riešeniam založeným na protokole SNMP je, že ponúkajú množstvo nastavení. Dajú sa nimi monitorovať napríklad iba určité rozsahy adries alebo vybrané protokoly a väčšinou obsahujú aj modul pre vizualizáciu dát.

Monitorovanie aktívnych spojení a otvorených sieťových portov plní v procese zabezpečenia a monitorovania operačného systému veľmi dôležitú úlohu. Ak by bol systém kompromitovaný a bol by na ňom spustený nejaký backdoor, alebo ak by sa útočník

pripájal pomocou tunelovaných spojení, tak monitorovanie aktívnych spojení a otvorených sieťových portov je prakticky jediná možnosť ako ho odhaliť. Na kritických systémoch je potrebné okrem reštriktívnych nastavení firewallu vykonať niekoľkokrát za deň monitorovanie spomínaných záležitostí. Bez problémov sa na to dajú použiť štandardné nástroje ako netstat alebo nmap, ktoré sú dostupné vo väčšine linuxových distribúcií.

7.3. Monitorovanie integrity súborového systému

Neodmysliteľnou súčasťou detekcie prienikov v operačných systémoch je sledovanie zmien dôležitých systémových súborov. Nie je nič nezvyčajné, ak v snahe vyťažiť maximum z úspešného prieniku, zamení útočník niektoré systémové programy za svoje upravené verzie. Ak zamení napríklad SSH klienta, môže získať prístupové údaje k iným systémom, na ktoré sa používatelia napadnutého systému vzdialene prihlasujú. Preto sa používajú techniky kontrolujúce integritu jednotlivých súborov pomocou hašovacích algoritmov ako napríklad MD5 (Message-Digest Algorithm 5) alebo SHA-1 (Secure Hash Algorithm). Pri prvom spustení nástroja pre kontrolu integrity súborového systému sa vytvorí databáza kontrolných súčtov definovaných súborov a je potrebné ju uložiť na médium chránené proti zápisu, aby ju útočník nemohol pozmeniť. Táto databáza je používaná ako etalón a všetky neskôr vypočítané kontrolné súčty sú proti nej porovnávané. Samozrejme v prípade aktualizácie dôležitých komponentov systému je potrebné referenčnú databázu aktualizovať.

7.4. Monitorovanie spustených procesov

Je dobré, ak bezpečnostná politika definuje pre každý systém zoznam procesov, ktoré na ňom môžu bežať. Procesy bežiace na systéme by sa mali nepretržite a v nepravidelných intervaloch porovnávať s týmto zoznamom. O každom výskyte neznámeho procesu by mal byť informovaný správca systému. Neustále monitorovanie bežiacich procesov však nemá charakter iba bezpečnostný, ale zabezpečuje aj opätovné spustenie dôležitých procesov v prípade ich pádu. Softvér vykonávajúci túto úlohu musí byť schopný o všetkých vykonaných akciách informovať správcu systému poprípade ich zaznamenávať do log súborov operačného systému.

7.5. Monitorovanie systémových log súborov

Vo svete unixových systémov takmer všetky procesy zapisujú informácie o svojej činnosti do log súborov. V skutočnosti ich tam nezapisujú priamo, ale prostredníctvom špecializovaného démona zvaného syslog. Tieto súbory sa zvyčajne nachádzajú v adresári /var/log, ktorý by mal byť čitateľný iba pre správcu systému, pretože môže obsahovať citlivé informácie o systéme alebo jeho užívateľoch. Ak počas činnosti niektorého z procesov nastane chyba, nezobrazí sa priamo na monitore, ale odovzdá sa jej popis démonovi syslog, ktorý ho zapíše do log súborov. Správca systému z nich následne môže v prípade problémov zistiť, čo bolo príčinou chýb. Tieto súbory je však potrebné kontrolovať pravidelne, pretože môžu obsahovať dôležité informácie o stave hardvéru, konfiguračných chybách, pokusoch o prienik a mnohých ďalších nemenej dôležitých udalostiach. Snáď

každému bežnému človeku sa po niekoľkých pokusoch o manuálne kontrolovanie týchto súborov zrodí v hlave myšlienka, že táto činnosť by mala byť automatizovaná, pretože je časovo náročná a príliš monotónna. Existuje viacero nástrojov, ktoré slúžia na automatizovanú kontrolu log súborov. Ich činnosť je v podstate rovnaká. Vyberú z log súborov iba riadky obsahujúce vybrané slová alebo riadky zodpovedajúce preddefinovaným regulárnym výrazom a odošlú ich správcovi systému.

V organizáciách, ktoré vlastnia viac ako dva unixové servery je vhodné zvážiť nasadenie dedikovaného logovacieho servera. Log súbory na ostatných serveroch sa tak môžu nechať rotovať nástrojom ako je napríklad logrotate a ich archiváciu stačí vykonávať iba na logovacím serveri. Ak by sa útočníkovi podarilo napadnúť niektorý zo systémov, môže síce zmazať záznamy na lokálnom disku, no nezmaže ich z logovacieho servera. Pri takom riešení však treba vyriešiť problém synchronizácie času medzi jednotlivými systémami, pretože časový posun medzi nimi môže znemožniť spätnú analýzu prienikov alebo iných mimoriadnych situácií. Na synchronizáciu času jednotlivých počítačových systémov sa zvyčajne používa protokol NTP (Network Time Protocol), ktorý je implementovaný aj v mnohých hardvérových produktoch synchronizovaných pomocou GPS (Global Positioning System).

7.6. Monitorovanie útokov a pokusov o prienik

Systémy na detekciu pokusov o prienik (ďalej len IDS) sú schopné zachytiť mnohé druhy zlomyseľných aktivít na sieti i v samotnom operačnom systéme, ktoré nemôžu byť zachytené bežnými firewallovými systémami. Okrem bežných sieťových útokov, ktoré môžu byť čiastočne blokované aj firewallom, dokáže IDS vďaka operovaniu na aplikačnej vrstve zachytiť aj prechádzajúce vírusy, trójske kone či internetové červy.

IDS môžu pracovať s bázou znalostí o prienikoch a porovnávať s ňou vyskytujúce sa udalosti, alebo môžu detekovať odchýlky od definovaného normálu. Ak sa ich činnosť opiera o bazu znalostí, dokážu rozoznávať len známe útoky, ktoré sú v databáze resp. podobné útoky, ktoré svojimi hlavnými prvkami pripomínajú už známe útoky. Kvalita týchto IDS závisí od kvality bázy vzoriek. Naproti tomu, IDS založené na vyhľadávaní odchýlok od definovaného normálu sa dokážu učiť a tak zlepšovať svoj výkon.

Podľa poľa pôsobenia môžeme IDS rozdeliť na HIDS (Host-based Intrusion Detection System) teda hostové systémy na detekciu prieniku a NIDS (Network Intrusion Detection System) čiže sieťové systémy na detekciu prieniku. Úlohou HIDS je monitorovať systémové volania, log súbory, modifikácie súborového systému a iné aktivity v konkrétnom operačnom systéme. NIDS sú zase často inštalované na prístupové body lokálnej siete, kde analyzujú sieťovú prevádzku celej siete. Existujú však aj tzv. aktívne IDS, ktoré fungujú ako adaptívne firewally a dokážu podozrivým IP adresám zabrániť v prístupe ku konkrétnemu systému alebo dokonca k celej lokálnej sieti.

8. Tomcat

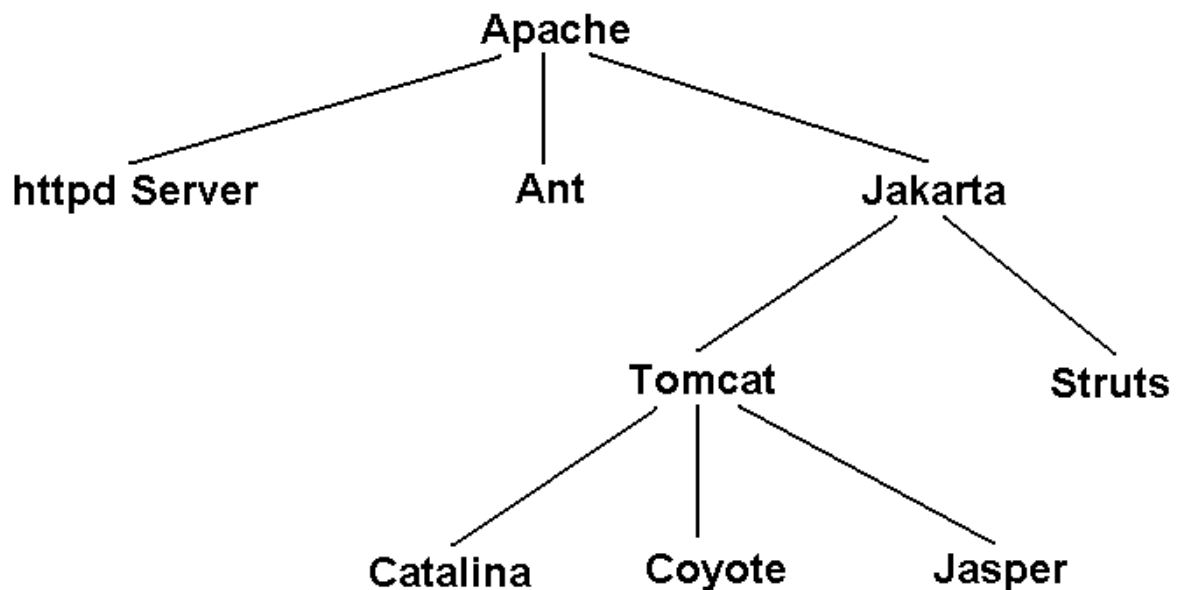
Programy potrebujú prostredie, v ktorom sú spustené v rámci hostiteľského počítača. V niektorých prípadoch operačný systém dokáže poskytnúť potrebné prostredie, ale niekedy je potrebné zvoliť viac sofistikovaný „kontajner“. Tomcat je kontajnerom, ktorý poskytuje prostredie pre Java code využívajúci web server.

Tomcat je servlet kontajner pre Java Servlety a JavaServer Pages. Poskytuje Java Virtual Machine a združuje základné prvky, ktoré ponúkajú kompletný Java Runtime Environment. Taktiež poskytuje web server software, ktorý vytvorí prostredie dostupné pre Web a možnosti konfigurácie ovládacích nástrojov, ktorých konfigurácia je uložená prehľadne v XML formáte.

Tomcat umožňuje pre vývojárov výhodu v implementovanom kontajnery pre servlety a JSP, kde výsledný produkt vývoja je kompatibilný s inými kontajnermi zahrnutými do štandardnej sady.

Tomcat sám o sebe obsahuje niekoľko elementov ako Catalina, Coyote a Jasper.

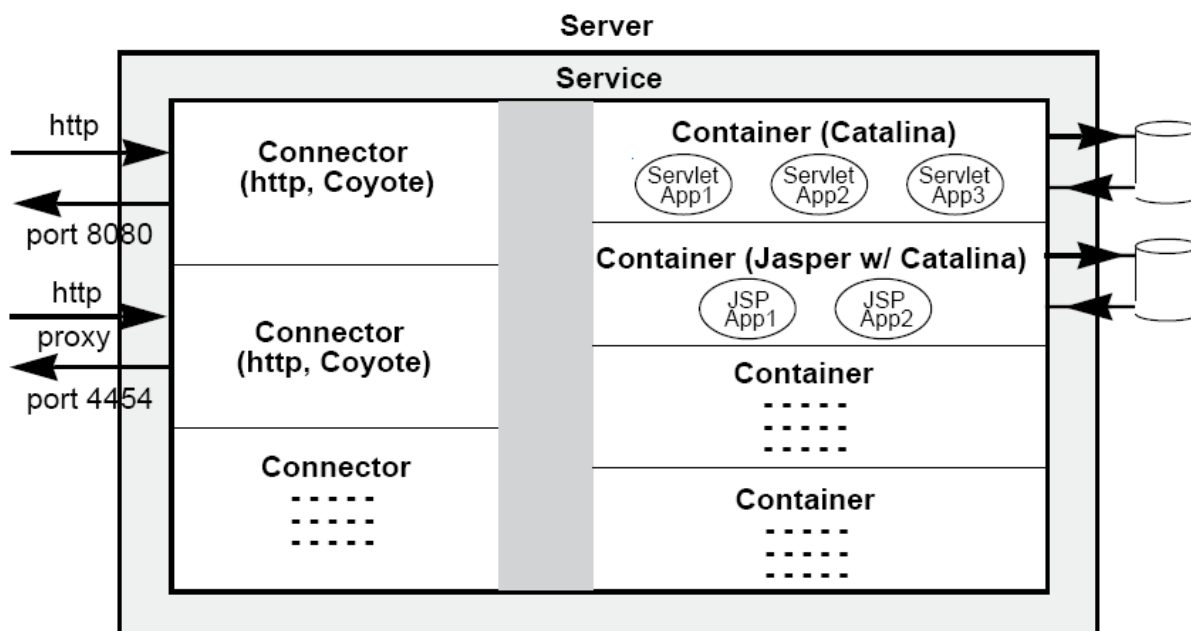
Catalina je Servlet kontajner Tomcatu. Coyote je web kontajner a Jasper je JSP nástroj, ktorý tomcat používa.



Obr. 4 : Štruktúra projektu apache.

8.1 Štruktúra Tomcat

Tomcat poskytuje služby ako pre windows, linux kde očakáva pripojenie na porte 8080. Jedna inštancia tomcatu môže poskytovať niekoľko služieb. Každá služba od tomcatu bude obsahovať najmenej jeden connector a kontajner v ktorom práve Catalina poskytuje službu.



Obr. 5: Štruktúra Tomcat.

Server, služba, connector, kontajner a engine sa dajú veľmi ľahko konfigurovať. Štandardná konfigurácia aplikácie môže prevziať základnú konfiguráciu, ak je to potrebné.

Tomcat Manager je užitočná aplikácia, ktorá je poskytnutá v jednom z tomcat kontajnerov a využíva sa pre kontrolu vyťaženia jednotlivých aplikácií.

Detailným rozborom tomcatu sa zaoberá [Jon Eaves](#) a [Warner Godfrey](#) vo svojej knihe Apache Tomcat Bible [4].

9. Použitie programu rsync

Program rsync bude v tejto práci použitý ako program pre zálohovanie systémových a aplikačných dát. Program rsync je unixový príkaz, ktorý synchronizuje rôzne súbory a adresáre po sieti. Môže poskytovať služby ako pre klienta, ktorý zálohuje svoje dáta na server, tak aj pre server, ktorý ukladá dáta od klienta. Využíva algoritmus minimalizujúci množstvo prenesených dát po sieti a počet spojení.

rsync – klient

- program môže byť využívaný lokálne
- pomocou rsync sa môže pripojiť k vzdialenému serveru
- využíva pripojenie cez ssh protokol
- dokáže listovať moduly zo vzdialeného servera

rsync – server

Aplikácia rsyncd, rsync server, používa TCP port 873. Môže byť spustený pomocou inetd alebo ako samostatne bežiaci proces.

10. Návrh riešenia

Vytvorenie funkčného a stabilného serverového prostredia pre callback systém, zahŕňa celý rad vedomostí týkajúcich sa mnohých oblastí. Spojenie týchto oblastí musí tvoriť jeden funkčný celok. Návrh riešenia pozostáva z nasledujúcich aplikácií:

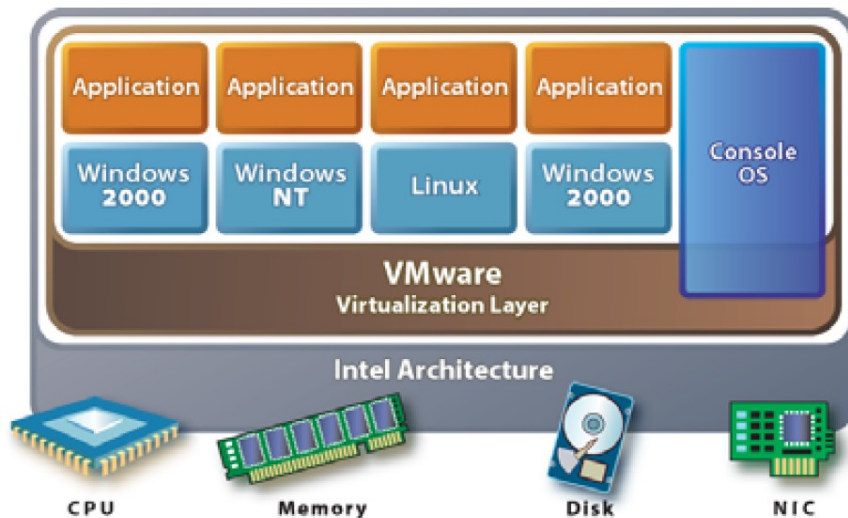
- ESX virtualizácia, bude pre nás poskytovať pohodlné ovládanie a uľahčí nám komunikáciu medzi hosťujúcim operačným systémom a fyzickým zariadením.
- Vysoká dostupnosť aplikácií v linuxovom operačnom systéme bude zaistená balíčkom heartbeat 2, ktorý nám poskytne možnosť vytvoriť High Availability cluster a Load balancing a dosiahnuť tak požadovaný zámer.
- mySQL je databázový produkt, ktorý bude clustrovaný a zabezpečí sa tým vysoká dostupnosť databáze.
- Tomcat poskytne servlet kontajnery pre Java servlety.
- Apache webserver umožní využitie a prístup do databázy pre zákaznícke účty. Apache webserver je priložený ako open source k obom operačným systémom, takže sa nemusíme starať o jeho licencovanie a ďalšie využitie. Je vhodné však použiť novšie verzie, ktoré sú na stránkach výrobcu k dispozícii.
- Návrh úložného priestoru pre operačný systém a dátové aplikácie prináša pri vhodnom návrhu dostupnosť dát aj v prípade hardwarových výpadkov.
- Monitorovací systém je jedna z dôležitých častí pre pokojný spánok správcu systémov. Využíva sa na monitorovanie stavu operačného systému a taktiež zaťaženia použitých aplikácií.
- Zálohovanie dát je potrebné zabezpečiť zálohovanie operačného systému a aplikácií a predísť tak omylom a chybám vlastníkov jednotlivých použitých aplikácií. Súbor budú zálohované podľa potreby s potrebnou históriou.
- Hlavnou časťou každej vytvorenej logickej partície bude operačný systém. Výber operačného systému je ľubovoľný. My sa zameriame na použitie Linuxového operačného systému a to RedHat Enterprise Linux 5 server, alebo Suse Enterprise 10 SP2. Sú to podobné operačné systémy a v konečnom dôsledku nie je rozdiel, ktorý systém použijeme a na ktorý sa budeme orientovať. Oba operačné systémy sú podporované spoločnosťami a za menší poplatok je možné ich použiť v produkčnej sfére.

Zoznam použitých produktov:

Operačný systém:	Red Hat Enterprise Linux 5 server alebo Suse Enterprise 10
Java Application Server:	Tomcat
JVM:	JVM
Database:	MySQL
MySQL Connector:	Connector/j
Apache webserver	
Mod_python	
Jvm 1.5.0_10	
Cheetah template	
sqlAlchemy	
PIL	

11. VMware ESX Server

VMware ESX Server je serverový systém vytvorený pre spoločnosti využívajúce virtualizačné technológie. ESX server je súčasťou skupiny VMware infraštruktúry, ktorá poskytuje pohodlné ovládanie a spoľahlivosť služieb. Pre vybudovanie popísaného prostredia je potrebná forma perzistentného uloženia, napr. koncepcia diskového priestoru, pre uloženie virtuálneho jadra a podpory súborov. Možnosť predstaviť si ako funguje virtualizované prostredie je uvedená na obrázku 6.



Obr. 6 : Koncepcia virtualizácie.

11.1. CPU virtualizácia

Pojmom CPU virtualizácia sa v prostredí informatiky označujú postupy a techniky, ktoré umožňujú k dostupným zdrojom pristupovať inak, než ako fyzicky existujúcim. Virtualizované prostredie je prispôbené potrebám užívateľa a to skrytím nepodstatných detailov a jednoduchým ovládaním (hardwarové prostriedky). Virtualizovať možno na rôznych úrovniach, od celého virtuálneho stroja, až po jeho jednotlivé hardwarové komponenty (virtuálne procesory, pamäť, disk, atď.), prípadne len softwarové prostredie (virtualizácia operačného systému).

Každá virtuálna partícia, ktorá beží na svojom vlastnom procesore alebo časti procesoru, je plne izolovaná od ostatných virtuálnych partícií so svojim vlastným registrom a ďalšími možnosťami riadenia. Inštrukcie sú priamo vykonávané na fyzickej vrstve CPU.

11.2. Pamäťová virtualizácia

Veľkosť pridelenej pamäte je použiteľná na každom virtuálnom stroji, ale veľkosť fyzickej pamäte nemusí mať nič spoločné s pridelenou virtuálnou pamäťou. Namiesto

spoločnej fyzickej pamäte sú stránky mapované efektívne pre každý virtuálny stroj. Niektoré časti fyzickej pamäte môžu byť mapované pre virtuálny stroj ako zdieľané pamäťové stránky, alebo ako pamäťové stránky, ktoré nie sú mapované. Riadenie virtuálnej pamäte je uskutočnené ESX serverom bez toho aby host'ovský operačný systém o tom vedel a taktiež bez interferencie s jeho pamäťovým subsystémom.

11.3. Disková virtualizácia

Disková podpora zariadení v ESX servery spočíva len v podpore niektorých vybraných technológií. Každý virtuálny disk je reprezentovaný ako SCSI disk pripojený do SCSI adaptéra. Zariadenie využíva storage controller host'ovského operačného systému aj napriek širokej možnosti použitia SCSI, RAID a Fibre Channel adaptérov, ktoré môžu byť využívané systémom. Tento spôsob vytvára virtuálny stroj viac robustný a prehľadnejší. Eliminuje obavy o stabilitu ovládačov, ktorá môže byť nainštalovaná na host'ovskom operačnom systéme. ESX Server môže byť použitý efektívne so storage area network (SAN). ESX server podporuje Qlogic a Emulex adaptér, ktoré dovoľujú ESX Serveru pripojiť sa do SAN a vidieť tak na diskové polia SANu.

11.4. Sieťová virtualizácia

Počas definície virtuálnych prostriedkov môžeme pre host'ovský operačný systém nadefinovať až štyri virtuálne sieťové karty. Každá virtuálna sieťová karta má svoju vlastnú MAC adresu a môže mať svoju vlastnú IP adresu. Virtuálne sieťové zariadenie je pripojené do virtuálneho switcha. Každý virtuálny switch je nakonfigurovaný výhradne ako virtuálna sieť bez pripojenia do fyzického LAN. Taktiež môže vytvárať bridge do fyzickej LAN siete alebo viacero fyzických sieťových portov na fyzickom servery.

12. Heartbeat

Heartbeat je voľne šíriteľný clustrovací produkt pre Linux. Heartbeat zabezpečuje vysokú dostupnosť a ovládateľnosť veľmi kritických zdrojov vrátane dát, aplikácií a servisu. Tento súhrn balíčkov podporuje failover, failback a migrácie v závislosti nastavenia clustrovaných prostriedkov.

Črty heartbeat:

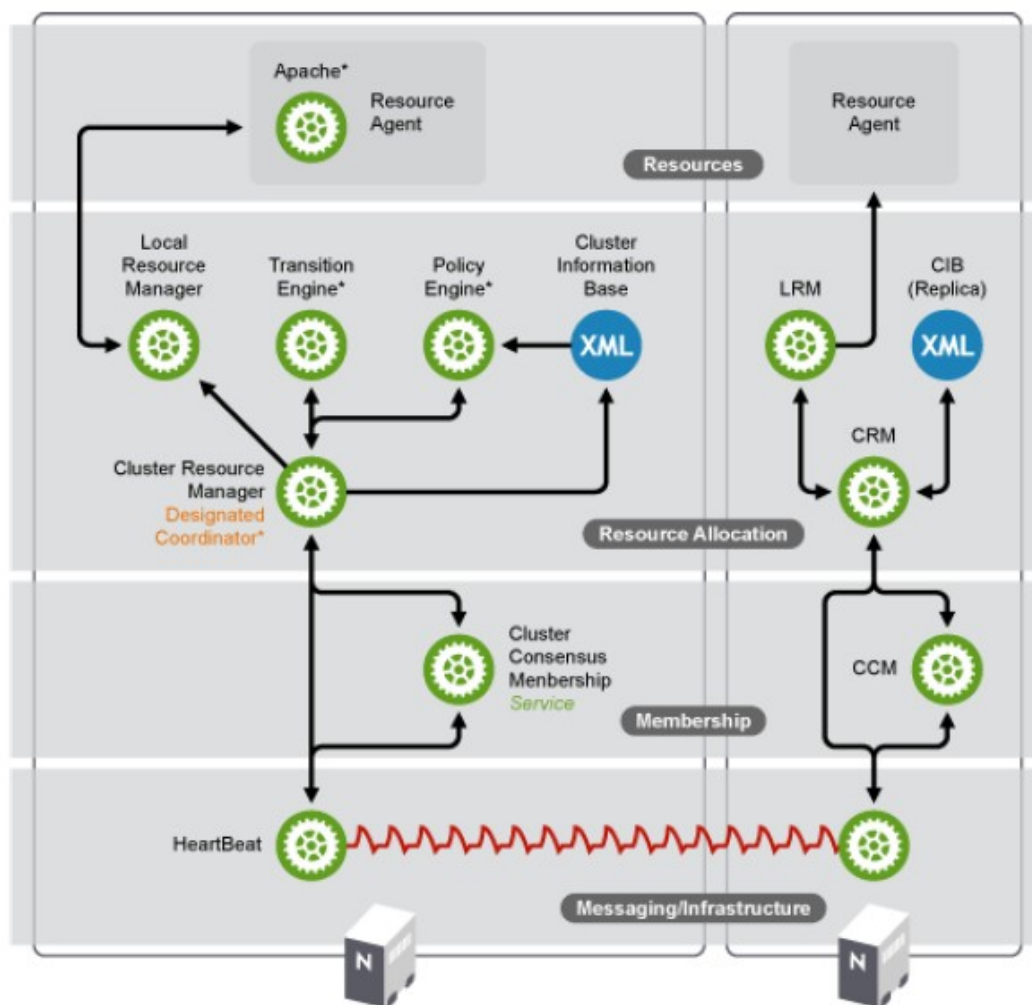
- podpora Fibre Channel alebo iSCSI
- viacnásobné clustery pozostávajúce až zo 16 linuxových serverov
- administrácia single point cez grafické prostredie alebo command nástrojom
- možnosť prispôbiť cluster aplikácii a hardwarovej infraštruktúre
- dynamické pridelovanie a znovu pridelovanie úložného priestoru podľa potreby
- podpora zdieľaného diskového priestoru
- podpora clustrovacieho súborového systému OCFS 2
- podpora pre cluster LVM ako EVMS

12.1. Architektúra Heartbeat

Architektúra heartbeat sa skladá z niekoľkých vrstiev:

- vrstva komunikačná a vrstva infraštruktúry
- členská vrstva
- vrstva pridelovania zdrojov
- vrstva zdrojov

Popis architektúry heartbeat je uvedený na obrázku 7.



Obr. 7: Architektúra Heartbeat.

Vrstva komunikačná a infraštruktúry

Prvá vrstva nazýva sa aj Heartbeat vrstva. Na tejto vrstve prebieha komunikácia medzi servermi v podobe „som tu a žijem“ rovnako ako ďalšie informácie.

Členská vrstva

Je zodpovedná za počítanie členstva serverov v prostredí clusteru a synchronizáciu výsledku všetkých členov tejto skupiny. To je spracovávané na základe informácií z Heartbeat vrstvy.

Vrstva pridelovania zdrojov

Pridelovanie zdrojov je viac komplexná činnosť a skladá sa z nasledujúcich komponentov:

- Cluster manager - každá vykonaná činnosť na vrstve pridelovania zdrojov sa uskutoční na tejto vrstve. Ak nejaký ďalší komponent z tejto vrstvy, alebo nejaký iný komponent z vyššej vrstvy potrebuje komunikovať, tak komunikuje cez Cluster resource managera.
- Cluster information base, alebo CIB, je ako databáza XML, kde je obsiahnutá kompletná clusterová konfigurácia a charakter clusteru. Obsahuje taktiež primárny CIB v clusterovom prostredí, ktorý je riadený DC (Designated coordinator). Všetci ostatní členovia majú uloženú len kópiu CIB. Policy Engine (PE) a Transition Engine (TE) sú používané pri zabezpečovaní služieb, kedy je potrebná akákoľvek zmena týkajúca sa clusteru.
- Local Resource Manager (LRM), tiež nazývaný ako agent v zastúpení CRM. Ten môže uskutočňovať start, stop, monitorovacie operácie a správy CRM. LRM je autoritatívny zdroj pre všetky súvisiace informácie na jeho lokálnom serveri.

Vrstva zdrojov

Táto posledná vrstva sa skladá z jedného alebo viacerých zdrojových agentov (Resource Agents - RA). Resource agent je program, najčastejšie shell script, ktorý je napísaný na štartovanie, zastavovanie a monitorovanie určitých druhov služieb. Základnými Resource agentmi sú LSB init scripty. Heartbeat tiež podporuje oveľa flexibilnejší Open Clustering Framework Resource Agent API.

12.2. Inštalácia a nastavenie

Heartbeat cluster môže byť nainštalovaný a používaný pomocou programu YaST. V tejto časti sa budeme zaoberať inštaláciou, nastavením a konfiguráciou Heartbeat clusteru. Podmienkou fungovania clusteru je, že na všetkých uzloch musí byť implementovaný potrebný program. Po nainštalovaní Heartbeat, je potrebné vytvoriť a nakonfigurovať cluster prostriedky. Ďalej je potrebné vytvoriť file system pre zdieľaný disk (SAN), ktorý bude konfigurovaný pomocou Heartbeat.

Príprava

1. Konfigurácia hostname je základ, pri ktorom každý člen clusteru je schopný vyhľadať člena podľa mena. Ak nie je člen clusteru dostupný cez hostname, interná komunikácia zlyhá. V tomto prípade je potrebné zmeniť */etc/hosts* súbor.

```
sql01:~ # cat /etc/hosts
```

```
127.0.0.1    localhost
fe00::0      ipv6-localnet
```

```
ff00::0      ipv6-mcastprefix
ff02::1      ipv6-allnodes
ff02::2      ipv6-allrouters
ff02::3      ipv6-allhosts
127.0.0.2    sql01
192.168.0.80 sql01
192.168.0.21 tsm01
192.168.0.60 central01
192.168.0.62 sql01_test
192.168.0.81 sql02
sql01:~ #
```

2. Konfigurácia a synchronizácia času. Každý člen clusteru bude používať server mimo vnútornej siete ako zdroj synchronizácie času.

YaST -> Network Services -> NTP Client

Je potrebné nastaviť **NTP** démona počas štartu systému a ďalej je dôležité použiť IP adresu serveru pre synchronizáciu času. Toto je potrebné uskutočniť na všetkých serveroch. Ak chceme zistiť či nám požadovaná synchronizácia funguje použijeme príkaz ***ntpd -p***

12.3. Inštalácia MySQL

Pre plnohodnotné fungovanie clusteru MySql je potrebné nainštalovať túto aplikáciu. Možností inštalácie je viac. Treba zvážiť najmä funkčnosť a variabilitu s rozšírením do budúcnosti.

Možnosti inštalácie MySql:

- použitím YAST
- RPM inštalácia
- Inštalácia binárnych súborov
- Inštalácia zo zdrojového kódu

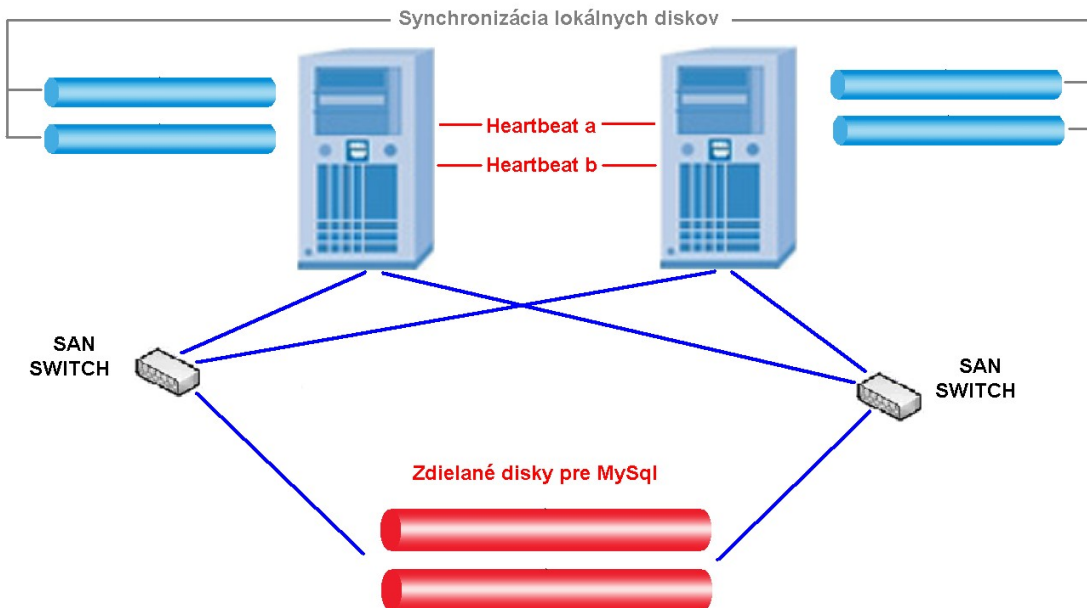
Orientácia na zdieľaný disk:

- kompiláciou do rootvg a dát do dátovej volume group (potrebná synchronizácia dát, nedostupná databáza počas údržby systému)
- kompiláciou zdrojového kódu a dát do dátovej volume group (obr. 8)

Postup inštalácie:

1. Vytvorenie pravidla pre užívateľa „mysql“ pod ktorým bude MySql bežať

- ```
groupadd mysql
useradd -g mysql mysql
```
2. Stiahnuť súbor mysql-6.0.10-alpha.tar.gz
  3. `# tar -xzf mysql-6.0.10-alpha.tar.gz`
  4. `# cd mysql-6.0.10-alpha`
  5. `# ./configure --prefix= /mysql/bin/usr/local/mysql-alpha --with-charset=utf8 --with-collation=utf8_general_ci`
  6. `# make`
  7. `# make install`
  8. `# ln -s /mysql/bin/usr/local/mysql-alpha/ /mysql/bin/start`
  9. Vytvorenie my.cnf súboru
  10. `# cp /mysql/bin/usr/local/mysql-alpha/share/mysql/my-small.cnf /etc/my.cnf`  
`# chown root /etc/my.cnf`  
`# chgrp root /etc/my.cnf`  
`# chmod 644 /etc/my.cnf`
  11. vloženie vytvoreného užívateľa do /etc/my.cnf „user = mysql“



Obr. 8: Zdieľaný úložný priestor pre dátové volume groupy.

## 12.4. Inštalácia Heartbeat

Pre funkciu heartbeat 2 je potrebné nainštalovať nasledujúce balíčky v našom prípade sú to:

```
sql01:/ # rpm -qa | grep heartbeat
heartbeat-pils-2.0.5-7.10
heartbeat-stonith-2.0.5-7.10
heartbeat-ldirectord-2.0.5-7.10
heartbeat-2.0.5-7.10
yast2-heartbeat-2.13.10-1.3
heartbeat-cmpi-2.0.5-7.10
```

## 12.5. Implementácia Heartbeat

1. Spustiť **YAST** -> **High Availability**, alebo **YAST2 heartbeat**, ktorý naštartuje heartbeat modul. Vytvoríme ním nový cluster a pridáme uzol do clustrovaného prostredia.
2. Pridať nový uzol do prostredia tým, že zadefinujeme **hostname**.
3. Určiť definíciu overovacích kľúčov, vybrať CRC metódu.
4. V ďalšom kroku je potrebné spresniť metódu, ktorá bude použitá pre internú komunikáciu medzi jednotlivými uzlami v prostredí clusteru. Táto konfigurácia je zapísaná v súbore **/etc/ha.d/ha.cf**  
V tejto časti sa definuje, akým spôsobom budú jednotlivé uzly v clustery komunikovať, posilať si správu. Pre lepšie zaistenie spätnej kontroly je vhodné použiť najmenej dve média cez ktoré preteká heartbeat.
5. Definovať spustenie clusteru. Je možné vybrať z možností, že sa Heartbeat spustí pri bootovaní systému, alebo spustením ručne pomocou príkazu **rheartbeat start**. Spustiť Heartbeat na inom serveri v clusteri je možné použitím príkazu **chkconfig heartbeat on**.
6. Konfigurovať Heartbeat na ostatných serveroch v clustery je vhodné pomocou príkazu **/usr/lib/heartbeat/ha\_propagate**. Na 64 bitovom systéme je príkaz **ha\_propagate** uložený v adresári **/usr/lib64/heartbeat/**. Tento príkaz skopíruje konfiguráciu na všetky uzly v clustery.
7. **rheartbeat start** naštartuje Heartbeat na každom uzle, kde sa nachádza kópia konfiguračného súboru.

## 12.6. Konfigurácia STONITH a Cluster Resource

STONITH je servis, ktorý využíva Heartbeat na ochranu zdieľaných dát. Heartbeat je schopný kontrolovať sieťovú komunikáciu a predchádza nožnej chybe jedného z uzlov a zabráňuje tak poškodeniu zdieľaných dát.

1. Štart HA management klienta a prihlásenie sa na cluster
2. Klikneme na STONITH v oblasti linux-ha -> konfigurácia

3. Aktivujeme STONITH označením pol'a „Enabled“
4. Vyberieme zdroj pridaním nových položiek
5. Vyberieme „Native as the item type“
6. Špecifikujeme zdroj ID pre STONITH resource
7. V časti „Type section“ vyberieme STONITH zariadenie, ktoré sa zhoduje s našou sieťou. Po pridaní STONITH resource typu a definícii ktoré typy môžu byť pridané do parametrov, je potrebné zadať hodnotu „Value heading“.
8. Označíme „clone“ možnosť, ktorá umožňuje zdrojom simulovať spustenie na viacerých uzloch v clusteri.
9. V časti „clone\_node\_max“ je potrebné zadať číslo inštancie STONITH, ktoré budú bežať na zvolenom uzle clusteru. Hodnota nastavenia by mala byť 1.
10. V časti „clone\_max“ je potrebné zadať počet uzlov v clusteri, kde bude bežať STONITH služba.
11. Zvolíme „Clone“ alebo „Master/Slave ID“.
12. Pridáme STONITH resource do clusteru, zobrazia „Resources“ v ľavej časti panelu.
13. Označíme zdroje na ľavej strane a naštartujeme STONITH zdroje.

Po vytvorení „STONITH resource“ musíme ešte vytvoriť lokálne obmedzenia. Lokálne obmedzenia je potrebné nadefinovať kvôli uzlom v clusteri na ktorých má bežať STONITH.

Spustením STONITH sa v clusteri vytvoria nadefinované lokálne obmedzenia.

## 12.7. Konfigurácia Resource

V tejto časti sa zameriame na konfiguráciu a nastavenie „resource“ pre cluster. Je využívaný pri vytváraní obrazu clustrovaných zdrojov a pri migrácii na iný uzol clusteru. Taktiež nám pomôže pri testovaní Heartbeat clusteru a zaručí nám správnosť funkcie celého clusteru. Nakonfigurujeme resource IP adresy a potom otestujeme jej správne fungovanie.

1. Spustenie a prihlásenie sa na HA management klienta.
2. V časti resource pridáme nový zdroj.
3. Použijeme Native ako resource.
4. Zadáme meno resource (ipaddress1).
5. Špecifikujeme typ resource IPAddr (OCF Resource Agent).
6. V Parameters sekcii, kde bola pridaná IP address source je potrebné zmeniť hodnotu „Value“.
7. Pridáme IP adresu na definovanie IP adresy cluster resource.
8. Pridáme parameter, kde budeme špecifikovať meno „nic“ a hodnotu „eth0“
9. Priradíme resource do clusteru
10. Vyberieme resource a naštartujeme ho.

## 12.8. Proces migrácie na iný uzol clusteru

Migráciu novovytvoreného resource na iný uzol v clusteri uskutočníme pomocou príkazov uvedených v tejto časti.

*crm\_resource -M -r resource\_name -H hostname*

Definujeme pri migrácii zdrojov ipaddress1 na uzol v clustery sql02:

*crm\_resource -M -r ipaddress1 -H sql02*

Použitím HA Managent Client pre migráciu: Resources -> Migrate resource

## 12.9. Ručná konfigurácia clusterovaných zdrojov

Zdroje sú definované ako typ služby, ktoré server poskytuje. Zdroje sú zahrnuté v Heartbeate a môžu byť kontrolované pomocou RA (Resource Agentov), LSB skriptov, OCF skriptov a ďalších dostupných skriptov. Všetky zdroje sú nakonfigurované v CIB (Cluster Information Base) „resource“ časti.

Pridať akýkoľvek zdroj do aktuálnej konfigurácie clusteru je možné pomocou XML súboru v určenom poradí pre tento zdroj. Uvedieme príklad - pridanie IP adresy 10.0.0.90 do clusteru:

```
<primitive id="ip_1" - 1
class="ocf" - 2
type="IPaddr" - 2a
provider="heartbeat" > - 2b
<instance_attributes>
<attributes> - 3
<nvpair name="ip" value="10.10.0.1"/> - 4
</attributes>
</instance_attributes>
</primitive>
```

1. Hodnota atribútu *id* pre *primitive* tag je ľubovoľná. Ako všetky ID v XML aj táto musí mať jedinečnú hodnotu pre systém.
2. Nasledujú tri atribúty, *class*, *type* a *provider*. Určuje presný script, ktorý je použitý pre *primitive*. Pre uvedenú IP je vhodné uložiť skript do nasledujúcej cesty: */usr/lib/ocf/resource.d/heartbeat/IPaddr*.
3. Všetky atribúty pre resource agenta sú vložené v zozname *nvpair* tag. Tento by sa nemal zamieňať s XML atribútmi, ktoré sú pridané! napríklad *primitive* tag.
4. V príklade je uvedený RA atribút ip, ktorý je použitý pre 10.0.0.90.

Ak konfiguruje zdroj pomocou Heartbeatu, tak spustenie zdrojov by nemalo byť inicializované z init súboru. Heartbeat by mal byť zodpovedný za štart a stop všetkých služieb.

Pridaním IPaddr do clusteru, je vhodnejšie najprv uložiť konfiguráciu do súboru s názvom ip\_1.xml a pridať potom obsah tohto súboru do clusteru pomocou príkazu:

### ***cibadmin -o resources -C -x ip\_1.xml***

ak pridanie tohto zdroja prebehlo úspešne, nový zdroj si môžeme pozrieť príkazom `crm_mon`, ktorý je spustený na uzle clusteru.

## **12.10. Konfigurácia a riadenie clusterovaných zdrojov**

Zdroje clusteru musia byť vytvorené pre každý zdroj alebo aplikáciu bežiacu a poskytujúcu službu na servery. Clustrovanými zdrojmi môžu byť: Web stránka, e-mail, databáza, file systém, virtuálny stroj, a mnoho ďalších serverových aplikácií alebo služieb, ktoré chceme urobiť dostupnými pre užívateľov.

Na to je možné využiť grafické rozhranie HA management Clienta, alebo ***cibadmin*** príkazový riadok. V tejto časti sa budeme zaoberať:

- vytváraním zdrojov
- konfiguráciou obmedzení
- špecifikáciou failover a failback uzlov
- konfiguráciou zdrojov monitoringu
- štartom a zmazávaním zdrojov
- konfiguráciou zdrojovej skupiny
- klonovaním zdrojov
- manuálnou migráciou zdrojov

Po inštalácii a vytvorení linuxového užívateľa „hacluster“ je potrebné nastaviť heslo, kvôli dostupnosti HA Mangement Clienta (***passwd hacluster***).

## **12.11. Vytvorenie clusterovaných zdrojov**

Vytvorenie zdrojov v clusteri:

1. Zvolíme voľbu „resource“ -> „add new Item“
2. Vyberieme zdroj ktorý má byť vytvorený. Kategórie zdrojov sú nasledujúce:
  - Navite
  - Group
  - Location (Constraint)
  - Order (Constraint)
  - Colocation (Constraint)
3. Definovať názov „Resource ID“
4. Zvoliť „Resource Type“
5. Vybrať „Resource Type“ môžeme ešte zvoliť ďalšiu premennú a to „Parameters“, kde ďalej definujeme hodnotu „Value“
6. Potvrdíme zvolené parametre a pridáme zdroj do clusteru.

### 12.12. Definícia Failover uzlov

Podstatou Failover clusteru je, že zdroje budú automaticky prepnuté na druhý uzol v dôsledku softwarovej alebo hardwarovej chyby. Ak používame viac uzlov v clusteri, potom pri preberaní zdrojov určitým uzlom je zodpovedný Heartbeat software. Ak chceme určiť, ktorý uzol bude preberať zdroje, musíme potom nadefinovať lokálne obmedzenia pre daný zdroj.

V HA management Clientovi definujeme:

1. Vybrať požadovaný zdroj alebo skupinu a pridáme novú podmienku
2. Vybrať „Location“ ako zdrojový typ
3. Špecifikovať ID pre novovytvorenú podmienku
4. Pridať podmienku do definícií, ktoré budú automaticky nastavené na nulu, aby sme sa vyhli nežiaducemu efektu, ktorý je spôsobený nekompletným pravidlom. Obmedzenia s vyššou hodnotou sú uprednostňované pred obmedzeniami s nižšou hodnotou. Vytvorením ďalších lokálnych obmedzení s odlišnou hodnotou pre daný zdroj môžeme využiť uzol v clusteri, ktorý bude preberať zdroje pri jeho nedostupnosti. Ak potrebujeme špecifikovať hodnotu obmedzenia, je možné ju meniť počas funkcie clusteru.
5. Na ľavej strane HA management clientovi je možné vybrať zdroj a určiť obmedzenia „Add Expression“
6. Definujeme hodnoty *#uname* pre „Attribute“, *eq* pre Operation a meno failover serveru ako Value.
7. Uložiť zmeny
8. Ak by sme sa rozhodli špecifikovať ďalší zdroj pre failover cluster, vytvoríme ďalšie lokálne obmedzenia pomocou krokov 1 až 8.

### 12.13. Definícia zdrojov Failback uzlov

Ďalšou definíciou a možnosťou pri vytváraní clustrovaného prostredia, je určenie uzlov v clusteri a odpoveď na otázku, čo sa bude diať, ak uzol ktorý mal poruchu sa znova prihlási do clusteru a je pripravený poskytovať služby. Jedna z možností je, že zdroj bude automaticky pripravený na prevzatie jeho originálnym uzlom, ak bude uzol znova online a registrovaný v clusteri. Ak chceme ochrániť zdroj pred opätovným prevzatím do uzla kde predtým bežal, alebo ak chceme špecifikovať rozdielny uzol pre zdroj, pre failback, musíme zmeniť jeho zdrojovú „stickiness“ hodnotu. Túto hodnotu môžeme špecifikovať keď vytvárame zdroj.

Uvedenú funkciu budeme aplikovať pomocou HA Management Clienta.

1. Zvolíme požadovaný zdroj, alebo grupu. Definovať „Attributes“ a potom pridáme požadovaný atribút.
2. V časti „Name“ je potrebné definovať „resource\_stickiness“.
3. V časti „Value“ špecifikujeme hodnotu medzi -100 a 100

Vysvetlenie hodnôt „stickiness“:

Hodnota nula

- je považovaná za defaultnú hodnotu. Zdroj bude umiestnený optimálne v systéme. To je chápané tak, že je zdroj premiestnený na uzol kde sú najlepšie podmienky pre jeho funkciu. Berie sa do úvahy vyťaženie uzla. Táto hodnota je najlepšou voľbou pre automatický „failback“ kam môže byť zdroj premiestnený (na uzol, kde predtým nebol aktívny, alebo tam kde sa nenachádza ani jeden aktívny zdroj).

Hodnota väčšia ako nula

- zdroj môže byť nadefinovaný tak, že ostane na jeho pôvodnej lokalizácii, ale môže byť presunutý ak je viac vhodných, alebo výhodnejších uzlov k dispozícii. Vyššie hodnoty indikujú silnejšie predpoklady nato, aby zdroj ostal tam kde sa nachádza.

Hodnota nižšia ako nula

- pre zdroj je preferovaná zmena doterajšej lokalizácie. Nižšie hodnoty indikujú silnejší predpoklad na presun zdroja.

Hodnota je sto

- zdroj je nadefinovaný tak, že bude vždy na jeho pôvodnej lokalizácii, pokiaľ nebude ukončené jeho trvanie na uzle v clusteri (jedným z dôvodov môže byť vypnutie uzla, zmena ako rezervný uzol, alebo konfiguračná zmena). Toto nastavenie je definované ako vypnutie automatického „failback“. Zdroje môžu byť presunuté na ďalší uzol, ako to bolo popísané u predchádzajúcich hodnôt.

Hodnota je mínus sto

- zdroje budú vždy presunuté z jeho súčasnej lokalizácie.

Dobrou pomôckou v tejto časti je príkaz *crm\_failcount*.

## 12.14. Konfigurácia monitorovania

Nakoľko Heartbeat môže detekovať poruchu na jednotlivých uzloch clusteru, má tiež schopnosť detekovať poruchu zdroja na jednotlivých uzloch clusteru. Ak si chceme byť istý, že zdroje bežia, musíme nakonfigurovať monitoring pre jednotlivé zdroje. Konfigurácia monitoringu sa skladá z nastavenia „timeout“, štart, hodnoty a intervalu. Interval definuje čas ako často má Heartbeat zistiť status zdrojov.

Konfigurácia monitoringu:

1. Vyberieme zdroj, pre ktorý chceme definovať monitoring v časti „Operations“
2. Zvolíme názov definície pre „Monitor“
3. Pridáme požadované hodnoty „Interval“, „Timeout“ a „Start Delay“
4. Spustíme monitoring

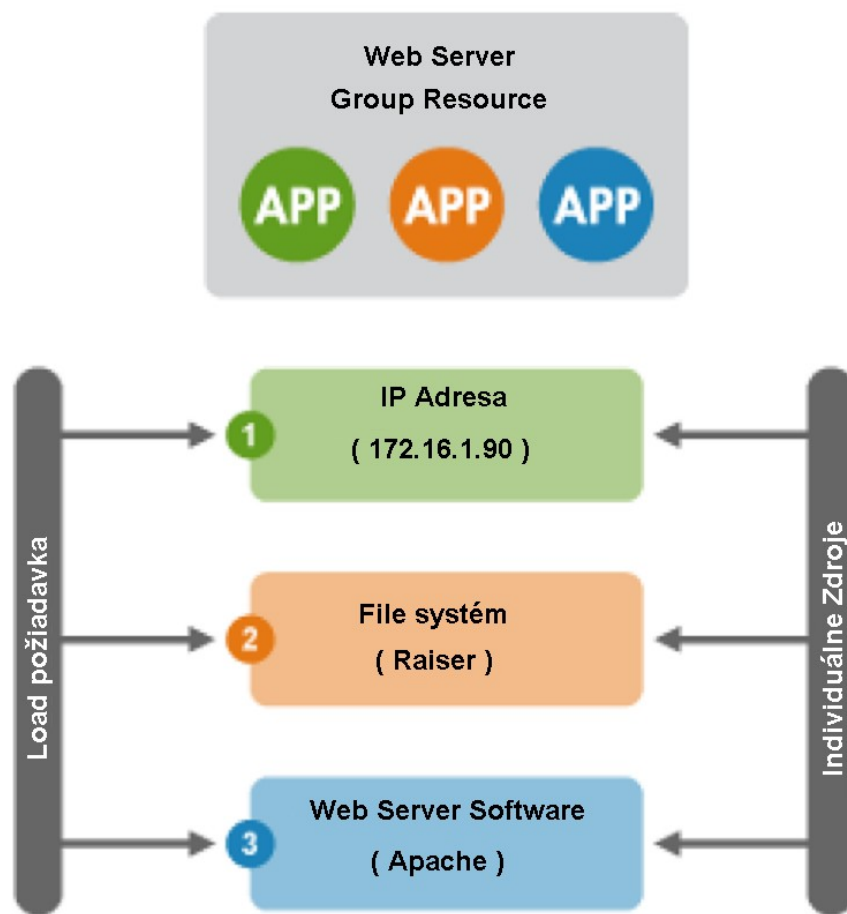
V prípade, že nenakonfigurujeme monitoring pre zdroje, tak zdroj nebude komunikovať a po jeho naštartovaní v clusteri bude vždy zobrazený ako riadne bežiaci.

Akcie, ktoré môžu nastať pri správne nakonfigurovanom monitoringu počas zlyhania zdroja :

- Log súbor začne generovať správy
- Zlyhanie bude zobrazené v hb\_gui a crm\_mon nástroji a CIB sekcii
- Cluster sa bude snažiť vykonať obnovenie vrátane zastaveného zdroja a reštartovať zlyhaný zdroj lokálne alebo na druhom núde

### 12.15. Konfigurácia Heartbeat Cluster Resource Group

Niektoré clustrované zdroje sú závislé od ostatných komponentov alebo zdrojov. Je dôležité, aby každý komponent alebo zdroj bol naštartovaný v špecifickej požiadavke. Podmienkou je, že musia ležať dohromady na tom istom servery. Príkladom je Web server, ktorý vyžaduje IP adresu a file systém. V tomto prípade je každý z komponentov clustrovaný zdroj, ktorý je kombinovaný v clusteri resource grupe. Resource grupa by mohla potom bežať na servery, alebo niekoľkých serveroch. V prípade softwarového alebo hardwarového zlyhania, zlyhaná resource grupa prejde na iný server ako jedna individuálna resource grupa.



Obr. 9: Ukážka kombinácie zdrojov v resource grupe

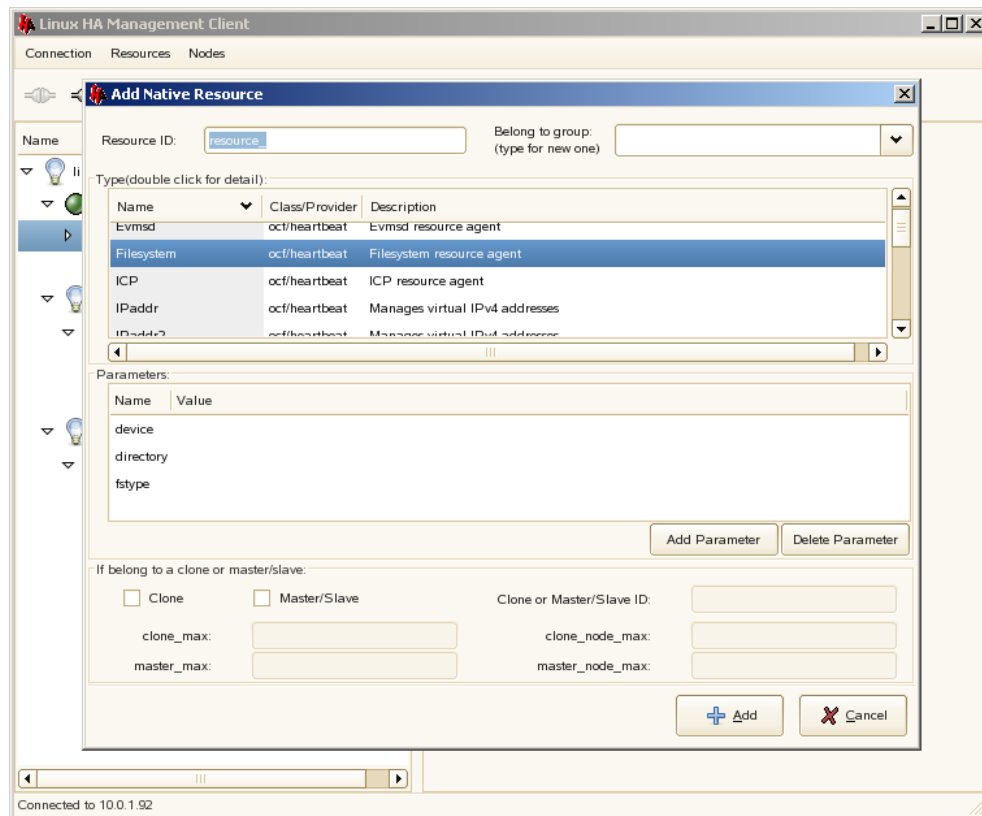
Konfigurácia resource grupy:

1. Pri konfigurácii použijeme HA Management Clienta.
2. Vyberieme záložku Resource a pridáme novú úlohu.

3. Definícia Grupy ako resource typ.
4. Špecifikácia ID pre grupu a hodnoty pre „Ordered“ a „Collocated“. Zvolíme hodnotu TRUE. Hodnota „Ordered“ označuje či zdroje v grupe budú zaťažené na požiadavku. Hodnota „Collocated“ špecifikuje, či zdroje v grupe budú bežať na tom istom serveri.
5. Zvoliť meno zdroja (ID) pre IP adresu zdroja časti grupy.
6. V sekcii „Type“ vybrať Ipaddr1 (OCF Resource Agent) ako resource type.
7. V sekcii Parameters zadať IP adresu zdroja.
8. Pridať IP adresu pre IP adresu clustrovaného zdroja.
9. Pridať parameter a vybrať hodnotu „nic“ a hodnotu „eth1“
10. Pridať zdroj do grupy.

Pridanie File systému do Resource grupy:

1. Pri konfigurácii použijeme HA Management Clienta (Obr. 10).
2. Vyberieme z menu položku „Native“
3. Špecifikovať názov zdroja (ID) pre súborový systém a pridať ho do grupy, ktorú sme vytvorili v predchádzajúcom kroku.
4. V sekcii „Type“ vyberieme „Filesystem“ ako resource typ.
5. V položke „Parameters“ definujeme tri hodnoty:
  - device
  - directory
  - fstype
6. Vyplníme typ device, directory a typ FS.



Obr. 10: HA management Client.

## 12.16. Konfigurácia a Heartbeat klonovanie zdrojov

Ak chceme aby niektorý zo zdrojov bežal súčasne na viacerých uzloch v clusteri, musíme nakonfigurovať požadovaný klon. Môžeme klonovať všetky zdroje, ktoré sa nachádzajú v databáze ResourceAgent. Klony zdrojov môžu byť konfigurované v závislosti od toho, na ktorom uzle majú byť spustené.

Rozlišujeme tri typy klonovania zdrojov:

- Anonymous Clones: Sú to jednoduché typy klonov, ich správanie je identické všade, kde práve bežia.
- Globally Unique Clones: Tieto zdroje sú rozdielne entity. Inštancia jedného klonu beží na jednom núde. Nie je však rovnocenná ďalšej inštancii na inom uzle.
- Stateful Clones: Aktívne inštancie sú rozdelené do dvoch stavov - aktívneho a pasívneho. Nesú označenie ako primárny a sekundárny, alebo master a slave. Stateful klon, môže byť jeden typu anonymous, alebo globally.

Vytvorenie klonovaných zdrojov:

1. Vytvoríme požadovaný zdroj.

2. Označiť „Clone“ záložku. Tak sa aktivuje zdroj ktorý beží súčasne na viacerých uzloch.
3. Definovať „clone\_node\_max“, vložiť počet inštancií zdrojov, ktoré budú bežať na danom uzle.
4. Definovať „clone\_max“, vložiť počet uzlov na ktorých bude zdroj bežať.
5. Pridáme klonovaný zdroj do clusteru.

### **12.17. Migrácia clustrovaných zdrojov**

Zdroje sú konfigurované tak, aby pri akejkoľvek zlyhaní serveru boli automaticky presunuté na iný uzol v clusteri. Taktiež je však možné manuálne migrovať zdroje na rozličné uzly použitím príkazového riadku:

***crm\_resource -M -r resource -H hostname -f***

***crm\_resource -M -r IP\_17 -H sql02***

Príkaz `crm_resource` si vysvetlíme v nasledujúcej časti.

Výpis z príkazu **crm\_mon**, ktorý nám vypíše nadefinované zdroje a resource grupy pre funkčný cluser. V prílohe č. 1 je zdrojový kód, ktorým je tiež možné vytvoriť cluster.

**[root@sql01 ~]# crm\_mon -l**

=====

**Last updated: Sat May 9 12:09:54 2009**

**Current DC: sql02 (223820e4-b067-4c62-b7b1-54d3e0217478)**

**2 Nodes configured.**

**3 Resources configured.**

=====

**Node: sql02 (223820e4-b067-4c62-b7b1-54d3e0217478): standby**

**Node: sql01 (ca0b9a69-74fd-4ddc-8235-8080a61b54f2): online**

**Resource Group: mysql**

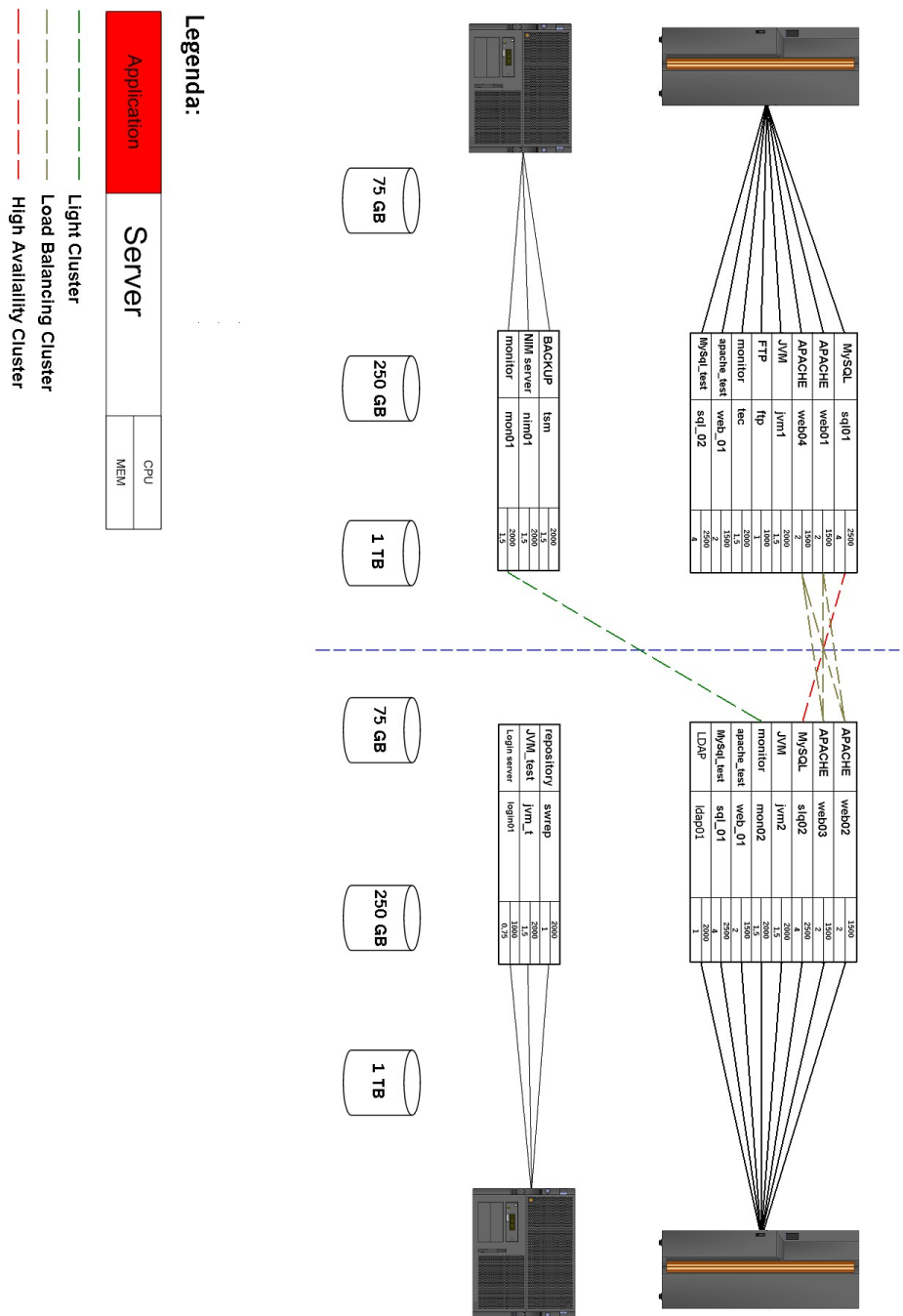
|                      |                                     |                      |
|----------------------|-------------------------------------|----------------------|
| <b>LVM_sqlvg</b>     | <b>(ocf::heartbeat:LVM):</b>        | <b>Started sql01</b> |
| <b>mysql_fs</b>      | <b>(ocf::heartbeat:Filesystem):</b> | <b>Started sql01</b> |
| <b>mysql_fake_fs</b> | <b>(ocf::heartbeat:Filesystem):</b> | <b>Started sql01</b> |
| <b>postgresql_fs</b> | <b>(ocf::heartbeat:Filesystem):</b> | <b>Started sql01</b> |
| <b>IP_17</b>         | <b>(ocf::heartbeat:IPaddr):</b>     | <b>Started sql01</b> |
| <b>mysqld</b>        | <b>(ocf::heartbeat:mysql):</b>      | <b>Started sql01</b> |
| <b>mysql_backup</b>  | <b>(ocf::heartbeat:TSMclient):</b>  | <b>Started sql01</b> |

**Resource Group: ldap**

|                    |                                     |                      |
|--------------------|-------------------------------------|----------------------|
| <b>LVM_ldapvg</b>  | <b>(ocf::heartbeat:LVM):</b>        | <b>Started sql01</b> |
| <b>ldap_fs</b>     | <b>(ocf::heartbeat:Filesystem):</b> | <b>Started sql01</b> |
| <b>IP_18</b>       | <b>(ocf::heartbeat:IPaddr):</b>     | <b>Started sql01</b> |
| <b>slapd</b>       | <b>(lsb:ldap):</b>                  | <b>Started sql01</b> |
| <b>ldap_backup</b> | <b>(ocf::heartbeat:TSMclient):</b>  | <b>Started sql01</b> |

**Resource Group: transcode**

|                        |                                     |                      |
|------------------------|-------------------------------------|----------------------|
| <b>LVM_transcodevg</b> | <b>(ocf::heartbeat:LVM):</b>        | <b>Started sql01</b> |
| <b>transcode_fs</b>    | <b>(ocf::heartbeat:Filesystem):</b> | <b>Started sql01</b> |
| <b>IP_32</b>           | <b>(ocf::heartbeat:IPaddr):</b>     | <b>Started sql01</b> |
| <b>avservice</b>       | <b>(ocf::heartbeat:tomcat_int):</b> | <b>Started sql01</b> |



Obr. 11: Callback environment.



### 13.Vysvetlenie príkazov pri práci s Heartbeat

***cibadmin*** – administruje Heartbeat Cluster Information Base

Zápis

```
cibadmin (--cib_query|-Q) -[Vrwlsmfbp] [-i xml-object-id|-o xml-object-type] [-t t-flag-whatever] [-h hostname]
```

Popis

Cibadmin je príkaz, pomocou ktorého je možné pružne narábať s Heartbeat CIB. Používa sa na výpis všetkých častí CIB, aktualizácií, úprav a mazaniu CIB a administratívnym úpravám CIB operácií. Cibadmin pracuje s XML stromovou štruktúrou CIB, prevažne bez zohľadnenia uskutočnených aktualizácií. Cibadmin by mal byť prednostne použitý na ručnú editáciu cib.xml a to hlavne ak je cluster aktívny.

***crmadmin*** – kontroluje Cluster Resource Manager

Zápis

```
crmadmin [-V|-q] [-i|-d|-K|-S|-E] node
```

Popis

Crmadmin bol pôvodne vyvinutý na kontrolu väčšiny operácií crm daemonu. Najväčšia časť jeho funkčnosti bola ovplyvnená vývojom iných nástrojov ako crm\_attribute a crm\_resource. Ďalšia možnosť použitia tohto príkazu je testovanie a zisťovanie statusu ***cmrd*** procesov.

***crm\_attribute*** – narába s atribútmi v CIB

Zápis

***crm\_attribute [options]***

Popis

Crm\_attribute – tento príkaz pracuje s parametrami uzlu a konfiguráciou clusteru, ktoré sú použité v CIB.

**crm\_diff** – identifikuje zmeny v konfigurácii clusteru a aplikuje balíčky do konfigurácie súborov.

Zápis

**crm\_diff [-?|-V] [-o filename] [-O string] [-p filename] [-n filename] [-N string]**

Popis

Crm\_diff príkaz asistuje pri vytváraní a aplikovaní XML balíčkov. To môže slúžiť ako vizuálna pomôcka pri zmenách medzi dvoma verziami konfigurácie clusteru. Tieto zmeny môžu byť použité neskôr pri príkaze **cibadmin**.

**crm\_failcount** – obsluhuje failcount na zadané zdroje

Zápis

**crm\_failcount [-?|-V] -D -u|-U node -r resource**  
**crm\_failcount [-?|-V] -G -u|-U node -r resource**  
**crm\_failcount [-?|-V] -v string -u|-U node -r resource**

Popis

Heartbeat umožňuje dômyselný prechod zdroja na iný uzol v prípade zlyhania na aktuálnom uzle. Zdroj využije zistenia resource\_stickiness a posúdi nutnosť prehodenia zdroja na iný uzol. Podobne tak využíva resource\_failure\_stickiness, ktorý určí hranicu, kedy sa zdroj prepne na druhý uzol. Failcount slúži na monitorovanie počtu zlyhaní zdroja a ak číslo prekročí dané preferencie, tak zdroj preberie druhý uzol do vtedy, pokiaľ nepríde k mazaniu histórie chýb. Crm\_failcount zisťuje počet chýb daného zdroja a je používaný k resetovaniu zistenej hodnoty.

**crm\_master** – určuje, ktorá instancia zdroja prebieha na primárnom uzle

Zápis

**crm\_master [-V|-Q] -D [-l lifetime]**  
**crm\_master [-V|-Q] -G [-l lifetime]**  
**crm\_master [-V|-Q] -v string [-l string]**

## Popis

Crm\_master je používaný pri vykonávaní skriptov, ktoré sa nachádzajú v resource agent a to na primárnom uzle. Nemal by sa používať z príkazového riadku. Slúži ako pomocný nástroj pre resource agentov. RA používa crm\_master na vykonanie jednotlivých podnetov primárneho uzla, alebo na zmazanie volieb. Volanie crm\_master má podobnú syntax ako crm\_attribute príkaz.

**crm\_mom** – monitoruje status clusteru

## Zápis

**crm\_mon [-V] -d -pfilename -h filename**

**crm\_mon [-V] [-l|-n|-r] -h filename**

**crm\_mon [-V] [-n|-r] -X filename**

## Popis

Crm\_mon umožňuje monitorovanie statusu a konfigurácie clusteru. Výstup obsahuje počet uzlov, uname, uuid, status, nastavenie zdroja a aktuálny stav clusteru. Výstup crm\_mon sa môže zobrazovať na terminále alebo do HTML súboru.

**crm\_resource** – ovplyvňuje Cluster Resource Manager

## Zápis

**crm\_resource [-?|-V|-S] -L|-Q|-W|-D|-C|-P|-p [options]**

## Popis

Crm\_resource príkaz uskutočňuje rôzne operácie súvisiace so zdrojom clusteru. Môže upravovať definície, konfigurácie clusteru, spustiť a zastaviť zdroj, a tiež zmazať a migrovať zdroj medzi nódami.

**crm\_standby** – určuje na ktorom uzle budú spustené zdroje

## Zápis

**crm\_standby [-?|-V] -D -u|-U node -r resource**

**crm\_standby [-?|-V] -G -u|-U node -r resource**

**crm\_standby [-?|-V] -v string -u|-U node -r resource [-l string]**

## Popis

Cmr\_standby príkaz pracuje s uzlami, ktoré sa nachádzajú v statuse standby. Uzol v standby núde nie je schopný vlastniť žiadny zdroj, ktorý by mohol byť naň presunutý. Standby mód môže byť nápomocný pri vykonávaní údržby a taktiež updatu kernela. Zmazanie standby atribútu z uzla môže nastať až v prípade, kedy je uzol plne aktívnym členom clusteru.

***crm\_uuid*** – vypíše UUID uzlu

## Zápis

***crm\_uuid [-w|-r]***

## Popis

UUID sa používa na identifikáciu uzlov v clusteru. Každé UUID musí mať vždy jedinečné identifikačné číslo. Príkaz ***crm\_uuid*** zobrazí a upravuje UUID uzlu, ktorý sa nachádza v clusteru. V prípade, že je Heartbeat prvý krát naštartovaný, tak si vytvorí UUID, ktoré je uložené v súbore ***/var/lib/heartbeat/hb\_uuid***. S týmto súborom pracuje príkaz ***crm\_uuid***.

***crm\_verify*** – kontroluje konzistenciu CIB

## Zápis

***crm\_verify [-V] -x file***  
***crm\_verify [-V] -X string***  
***crm\_verify [-V] -L|-p***  
***crm\_verify [-?]***

## Popis

Cmr\_verify kontroluje konfiguračnú databázu (CIB) kvôli konzistencii údajov. Taktiež môže byť použitý na kontrolu obsahu súboru, alebo na pripojenie uzla do funkčného clusteru. Podáva správy o dvoch typoch problémov - chyby a varovania. Hlásené chyby musia byť opravené predtým, ako Heartbeat začne pracovať. Cmr\_verify asistuje pri vytváraní nových modifikačných konfigurácií.

## 14. Záver

Cieľom práce bolo navrhnuť kompaktné prostredie pre Callback systém, kde bol kladený dôraz na použitie technológie vysokodostupného prostredia pre MySQL. Výsledkom je navrhnutý funkčný systém. Ďalším krokom bude simulácia a testovanie aplikácií vo vývojárskom prostredí. V ňom sa ešte doladia a odstránia chyby aplikácií a možné nedostatky pomocou simulovania reálneho prostredia. Po testovacej fáze sa predpokláda nasadenie do produkcie.

Počas vytvárania tohto prostredia boli zistené chyby pri aplikácii Heartbeat, ktoré sa neskôr odstránili inštaláciou nových balíčkov. Taktiež použitie operačných systémov SLES 10 SP2 alebo Redhat 5.2 nebolo najvhodnejšou voľbou, nakoľko tieto operačné systémy nie sú poskytované zdarma. Ďalším problémom, ktorý sa vyskytol, bolo vytvorenie správneho zdieľaného diskového priestoru pre jednotlivé aplikácie. To sa vyriešilo použitím LVM a vmkfstools, kde bol vytvorený „thick“ diskový formát.

Výhodou celého prostredia je rozdelenie jednotlivých aplikácií na samostatné operačné systémy, čo je obrovským prínosom pre funkčnosť celého prostredia. Tým pádom sa odstránil problém s aktualizáciami, zálohovaním, obnovovaním jednotlivých systémov spolu s aplikáciami, čo minimalizuje čas nedostupnosti služby a v kombinácii s použitím heartbeatu vyrieši tento problém.

Jedinou nevýhodou je veľmi vysoká cena virtualizačného nástroja, ktorej vývoj je nákladný, čo má dopad na celkovú cenu navrhnutého prostredia. Zo strany vmware je poskytnutý servis pre aktualizácie a riešenie prípadných problémov. V konečnom dôsledku nemožno poprieť, že najkvalitnejšou zo všetkých poskytovaných virtualizačných platforiem na x86 architektúre je platforma ESX 3.5. Počiatočná investícia do virtualizačných nástrojov je mnohonásobným prínosom v odstránení nedostupnosti služieb, ušetrených nákladov pri kúpe samostatných serverov, nákladov na prevádzku, ale aj, odhliadnuc od obchodných stratégií, šetrenie životného prostredia. Na druhej strane v dnešnej dobe už nie je možné použiť pri takomto komplexnom systéme aplikáciu, ktorá by bola zadarmo.

V poslednom rade bolo poukázané na to, že nie len najlepšie služby, ovládateľnosť a vysokú dostupnosť musí poskytovať len veľmi nákladná kombinácia aplikácií a operačného systému ako AIX s DB2, alebo Solaris v spojení s Oracle, atď.

Čo sa týka budúcnosti je isté, že virtualizačné platformy budú nasadzované do praxe. Určité problémy sa môžu vyskytnúť v myšlienke nasadzovania jednoúčelového operačného systému a aplikácie.

## **Literatúra**

- [1] Backer, R., Massiglia P.: Storage area network essentials: a complete guide to understanding and implementing SANs, John Wiley and Sons, 2001
- [2] Bookman, Ch.: Linux Clustering Building and Maintaining Linux Cluster. New Riders Publishing, 2003
- [3] Bourke, T.: Server Load Balancing. O'Reilly & Associate, Second Edition, 2004
- [4] Eaves, J., Godfrey, W.: Apache Tomcat Bible, Wiley-India, 2006
- [5] Evan, M., Hal, S.: Blueprints for High Availability. Wiley Publishing, Second Edition, 2005
- [6] Gerner, J., Naramore, E., Warden, M.: Linux, Apache, MySQL and PHP5 Web Development. Wiley Publishing, 2006
- [7] Gropp, W., Lusk, E.: Beowulf Cluster Computing with Linux, MIT Press, 2001
- [8] Kopper, K.: The Linux Enterprise Cluster—Build a Highly Available Cluster with commodity Hardware and Free Software. No Starch Press, San Francisco, 2005
- [9] Muller, A., Wilson S., Happe D.: Virtualization with VMware ESX Server, Syngress, 2005
- [10] Nusairat, J.: Beginning JBoss Seam From Novice to Professional, Apress, New York, 2007
- [11] Oglesby, R., Herold, S.: VMware ESX Server Advanced Technical Design Guide. Brian Madden, Mattoon, IL., 2005
- [12] Petersen R.: Linux: the complete reference, McGraw-Hill Professional, 2007
- [13] Ward, B.: How Linux works: what every superuser should know, No Starch Press, 2004
- [14] Wolf, Ch., Halter, E.: Virtualization: from the desktop to the enterprise, Apress, 2005

## **Zoznam príloh**

Príloha č.1    CD – zdrojový kód cib.xml