Řešení vysoké dostupnosti pro síťové filesystémy v Linuxu

Martin Kravec

Máj 9, 2012

Prehlásenie

Prehlasujem, že som bakalársku prácu spracoval samostatne, a že som uviedol všetky použité premene a literatúru, z ktorej som čerpal.

Poďakovanie

Chcem poďakovať všetkým, ktorí ma podporovali a pomáhali mi, najmä svojim rodičom a starým rodičom za trpezlivosť a podporu pri štúdiu a Anke za motiváciu. V neposlednom rade ďakujem vedúcemu mojej práce, Ing. Lubošovi Pavlíčkovi za pomoc pri tvorbe práce a Tobimu za cenné rady, ktoré mi dal.

Abstrakt

Cieľom tejto práce je oboznámenie čitateľov s problematikou vysokej dostupnosti a poskytnutie základného prehľadu o voľne dostupných technológiách, pomocou ktorých sa dá dosiahnuť. Zameriavam sa predovšetkým na riešenie problémov vyplývajúcich z hardwarových chýb systémov. Dokument bude možné použiť na získanie základných vedomostí potrebných pre konfiguráciu vlastného riešenia.

V praktickej časti realizujem vysoko dostupné klastrové riešenie pomocou voľne dostupných technológií a zároveň porovnám výkonnosť rôznych súborových systémov na tejto platforme. Práca môže poslúžiť aj ako návod ako pomocou vlastnej konfigurácie otestovať a čo najvhodnejšie zvoliť súborový systém pre použitie v produkčnom prostredí.

Kľúčové slová

vysoká dostupnosť, pacemaker, drbd, linux, súborový systém

Abstract

The goal of this paper is to inform readers about high availability problematics and provide basic overview of free technologies you can use to achieve high availability. I focus mainly on solution of problems caused by hardware failures. Document can by used to gain basic knowledge needed for configuration of your own solution.

In practical part I will try to realize high available cluster using free technologies and compare performance of different filesystems on this platform. Document will provide instructions for choosing and testing appropriate filesystem for specific configuration that can be used in production environment.

Keywords

high availability, pacemaker, heartbeat, drbd, linux, filesystem

Obsah

Úvod

V dnešnej dobe sa kladie čoraz väčší dôraz na zabezpečenie nepretržitej dostupnosti služieb, ktoré sú pre firmu kľúčové. Niektoré systémy, ako napríklad DNS už vo svojom návrhu počítali s tým, že hardware má obmedzenú životnosť a dokážu fungovať aj pri poruche časti infraštruktúry. Avšak vzhľadom na náklady, čas a určenie mnohých programov nie všetky je možné vyvíjať spolu s prvkami, ktoré zabezpečia ich dostupnosť aj v prípade poruchy. Preto vznikli systémy, pomocou ktorých je možné automaticky riadiť migráciu procesov medzi jednotlivými počítačmi a tým dosiahnuť čo najkratšiu doby nedostupnosti systému.

V mojej práci sa práve týmto systémom venujem. Pokúsim sa objasniť čitateľom základné princípy, na ktorých dnešné riešenia fungujú. Cieľom práce je poskytnúť prehľad o možných riešeniach a detailne popísať a otestovať jedno z voľne dostupných riešení. Práca poslúži čitateľovi k oboznámeniu sa s problematikou a získa základný prehľad o systéme fungovania týchto systémov.

Samotný dokument je rozčlenený na teoretickú a praktickú časť. V teoretickej časti sa čitateľ dozvie čo vlastne vysoká dostupnosť je a v akých oblastiach sa využíva. Ďalej bude nasledovať popis rôznych softwarových riešení, ktoré je možné využit k jej dosiahnutiu a zhrnutie najznámejších problémov, na ktoré je potreba myslieť pri realizácii samotného riešenia. Na záver teoretickej časti spomeniem niektoré najznámejšie softwarové produkty z tejto oblasti. V praktickej časti realizujem jedno z voľne dostupných riešení a jej súčasťou budú aj testy porovnávajúce výkonnosť rôznych súborových systémov. Konfigurácia týchto testov môže poslúžiť ako základňa pre ďalšie testovanie a zároveň pomôcť s výberom konkrétneho riešenia v závislosti na cieľovom prostredí.

Vo svojej práci som čerpal z množstva materiálov dostupných prevažne online. Čiastočne som využil aj prácu Tomáša Zvalu[1] o vysokej dostupnosti dát na virtuálnych serveroch, ktorú som rozšíril napríklad o pohľad na systémy zabezpečujúce automatické

migrovanie procesov v prípade výpadku, podrobnejší popis vybraných oblastí a testy rýchlostí súborových systémov. Využil som tiež prácu, ktorú napísal Radek Zima[2], konkrétne časť popisujúcu možnosti ukladania dát.

Pri písaní som sa značne opieral o dokumentáciu riešení Pacemaker, DRBD a RHCS.

Kapitola 1

Vysoká dostupnosť

V tomto oddiele sa budem venovať priblíženiu pojmu vysokej dostupnosti a bežným riešeniam, ktoré sa používajú na riešenie tejto problematiky. Predvediem spôsob počítania dostupnosti z manažérskeho pohľadu a problém v rozlíšení nedostupnosti systému a služby.

1.1 Čo je vysoká dostupnosť

Vysoká dostupnosť, tiež nazývaná RAS[1] označuje počítačový systém, ktorý sa dokáže rýchlo obnoviť z výpadku. Môže nastať minúta alebo dva nedostupnosti, počas ktorých systém migruje, potom ale bude ďalej pokračovať v činnosti bez zásahu administrátora. Neznamená to to isté ako fault-tolerant systém, kde je nepretržitá prevádzka navrhovaná tak, aby bolo možné pokračovať pri výpadku bez jeho zaznamenania. Systémy vysokej dostupnosti tiež umožňujú upravovať jednotlivé komponenty systému, alebo reštartovať jednotlivé servery bez nutnosti celé riešenie dočasne odstaviť[3].

Vysoká dostupnosť môže byť pochopená viacerými spôsobmi. Musíme rozlišovať, či hovoríme o plánovanej alebo neplánovanej nedostupnosti. Plánovaná nedostupnosť zahŕňa prípady kedy systém vypneme zámerne, či už v dôsledku reštartu po inštalácii, aktualizácií, alebo zmene konfigurácie systému. Tomu zvyčajne predchádzajú prípravy takejto udalosti na úrovni managementu a taktiež komunikácie so zákazníkmi. Neplánované výpadky sú ťažko predvídateľné a ich príčiny môžu byť rôzneho pôvodu - od hardwarových a softwarových problémov až po chyby ľudí a prírodné katastrofy. Príkladom môžu byť

[1] Reliability, Availability, Serviceability

výpadky RAM pamätí, pevných diskov, procesorov, sieťovej infraštruktúry, alebo útoky z internetu. Plánované výpadky nemajú veľký vplyv na obavy o dostupnosť systémov, keďže sú realizované prevažne v časoch, kedy tieto systémy nie sú využívané a zákazníci sa na nich môžu dopredu pripraviť.

Ako najjednoduchšie riešenie sa v praxi používa pravidelné zálohovanie. Zálohovanie síce rieši problém obnovy dát a všetky seriózne firmy dôležité dáta zálohujú, ale to je len prvý a krátkozraký krok, pretože obnova záloh je zdĺhavý proces sprevádzaný výpadkom. Trvanie nedostupnosti môže tvať niekoľko hodín, až dní v závislosti od závažnosti a času poruchy. Pri obnove zo záloh je totiž často potrebné zaobstarať hardware podobný tomu predchádzajúcemu, nainštalovať pôvodný operačný systém, potrebné aplikácie a značnú dobu tiež trvá kopírovanie veľkého množstva dát. Tiež sa líši reakčná doba v pracovnom čase a počas víkendu. Trvanie výpadku v rádu hodín alebo dní je však často neakceptovateľné.

V mnohých prípadoch je výpadok systému tolerovateľný len počas niekoľkých sekúnd. Ak na pár minút prestane fungovať elektronická pošta alebo konferenčné hovory tak si väčšina ľudí ide uvariť kávu. Avšak v prípade, že sa v letovej veži namiesto polohy lietadiel ukáže čierna obrazovka, je tento stav tolerovateľný len na pár sekúnd. Software použitý v praktickej časti (Pacemaker) slúži okrem iného práve na udržanie stálej prevádzky letovej spoločnosti DFS.

Systémy vysokej dostupnosti síce nie sú životne dôležité v každej situácii, avšak väčšina poskytovateľov elektronických služieb k nim smeruje už len z dôvodu prilákania zákazníkov. Či už ako marketingový plán alebo spôsob ako predísť migrácii klientov ku konkurencii v dôsledku príležitostnej nedostupnosti ich služieb. Napríklad hostingy internetových stránok často ponúkajú priestor zdarma. Od plateného sa väčšinou líši okrem iného aj garantovanou dostupnosťou.

Pri počítaní dostupnosti systému treba brať do úvahy aj rozhodnutie ako pristupovať k nedostupnosti konkrétnej služby. Nie sú výnimočné prípady kedy je operačný systém v poriadku, ale konkrétna aplikácia neodpovedá. Stáva sa to napríklad pri preťažení aplikácie väčším množstvom požiadavkov ako je schopná spracovať, aj keď je systém v poriadku. Vidíme, že v tomto prípade sa bude aplikácia užívateľom javiť ako nedostupná a zároveň administrátor môže tvrdiť 100% dostupnosť.

Dopady systémových výpadkov sa dajú rozdeliť na krátkodobé a dlhodobé. Medzi tie

krátkodobé patria stratený zisk a produktivita. Ich cena sa vyčísľuje oveľa ľahšie, avšak často tvoria len špičku ľadovca. Z dlhodobého hľadiska firma prichádza o svoje dobré meno, lojalitu zákazníkov, investorov. Výnimkou nie sú zmluvné pokuty, počítať treba aj s cenou obnovenia dát. Plný dopad trojdňového výpadku dobre znázorňuje príklad firmy RIM (BlackBerry), ktorá vyčíslila straty vo výške 54 miliónov dolárov. Na obrázku 1.1 je vidieť odhadovaný dopad hodinového výpadku pre rôzne odvetvia.

Industry	Hourly Cost of Downtime	Lost Revenue per Employee
Energy	$2,817,846	$569
Telecommunications	$2,066,245	$187
Manufacturing	$1,610,654	$134
Finance/Brokerage	$1,495,134	$1,080
Information Technology	$1,344,461	$184
Insurance	$1,202,444	$371
Retail	$1,107,274	$244
Pharmaceuticals	$1,082,252	$168
Banking	$996,802	$131
Food Processing	$804,192	$153
Consumer	$785,719	$128
Chemicals	$704,101	$195
Average	$1,010,536	$206

Obr. 1.1: Cena hodinového výpadku v rôznych odvetviach [4]

V praxi je často spomínaná percentuálna dostupnosť systémov, definovaná aj ako „počet deviatok“. Na prvý pohľad sa môže zdať číslo 99% veľa, z obrázku 1.2 však možno vyčítať, že tomu odpovedajúca doba výpadku môže byť nepríjemná a to najmä v závislosti na čase kedy sa vyskytne. Vysoká dostupnosť zvyčajne znamená 3 a viac deviatok a počítajú sa do nej aj plánované výpadky [5].

Konfigurácie využívajúce vysokú dostupnosť sú využívané predovšetkým v odvetviach, kde by výpadok mal veľký dopad na užívateľov a tých ktoré vyžadujú 24-hodinovú prevádzku, napríklad nemocnice, banky, letecké spoločnosti alebo nukleárne zariadenia.

Pri predstavení pojmu vysoká dostupnosť je potrebné vysvetliť význam ďalšej dôležitej skratky SPOF[2]. Tento všeobecný pojem sa používa pre komponent systému ktorý v prípade chyby spôsobí znefunkčnenie celého systému. Pritom nezáleží či je komponent hardwarový, softwarový alebo elektrický. Príkladom môže byť napájanie vysoko dostupného serverového klastra z jedného elektrického zdroja. V prípade chyby zásuvky je celá redundantná infraštruktúra zbytočná [6].

[2]Single Point of Failure

PERCENTAGE UPTIME	PERCENTAGE DOWNTIME	DOWNTIME PER YEAR	DOWNTIME PER WEEK
98%	2%	7.3 days	3 hours, 22 minutes
99%	1%	3.65 days	1 hour, 41 minutes
99.8%	0.2%	17 hours, 30 minutes	20 minutes, 10 seconds
99.9%	0.1%	8 hours, 45 minutes	10 minutes, 5 seconds
99.99%	0.01%	52.5 minutes	1 minute
99.999%	0.001%	5.25 minutes	6 seconds
99.9999% ("six 9s")	0.0001%	31.5 seconds	0.6 seconds

Obr. 1.2: Percentná dostupnosť v závislosti na dobe dostupnosti systému [6]

1.2 Test desiateho poschodia

Tento termín vymyslel Steve Traugott a týka sa situácie kedy potrebujeme obnoviť prevádzku systému do pôvodného stavu a to v čo najkratšom čase po výpadku. Test je založený na jednoduchej otázke „Môžem zobrať akýkoľvek bežiaci stroj ktorý nikdy nebol zálohovaný a vyhodiť ho z desiateho poschodia tak, aby firma neprišla o dáta a obnoviť pôvodnú prevádzku v intervale 5-10 minút?“ Ak môžte odpovedať áno, testom ste prešli [7].

Práca systémových a aplikačných administrátorov vo veľkých firmách spočíva aj v príprave na takéto úlohy. Nie je výnimkou, že sa pravidelne testuje pripravenosť personálu obnoviť pôvodnú prevádzku vypnutím niektorého z počítačov - samozrejme v kontrolovaných podmienkach a s možnosťou opätovného nasadenia pôvodného systému v prípade poruchy.

1.3 Ako je možné zvýšiť dostupnosť

Existuje obrovské množstvo spôsobov ako zabezpečiť, že systémy budú fungovať stabilnejšie a vydržia dlhšie. Tým najjednoduchším je často nákup kvalitného hardwaru a pravidelná aktualizácia, avšak každý komponent má obmedzenú životnosť a istú náchylnosť k pokazeniu sa. V nasledujúcich odstavcoch zhrniem tie najpoužívanejšie so zameraním na

tri hlavné oblasti.

1.3.1 Dostupnosť úložného priestoru

Dáta sú základným kameňom, na ktorom stavia každá firma. Či už sú to firemné faktúry, zoznamy zákazníkov alebo internetové stránky, žiadna firma si nemôže dovoliť o ne prísť. Je nepredstaviteľné že by banka svojim klientom oznámila, že stratila záznamy o paňažných zostatkoch na ich účtoch. Tieto informácie sú uchovávané na pevných diskoch, ktorých dostupnosť sa dá zvýšiť jedným z nasledujúcich spôsobov.

RAID

Toto riešenie používa takmer každá firma - či už malá alebo veľká. RAID[3] môže byť hardwarový aj softwarový. Každý z nich má svoje výhody:[8]

Softwarový raid je zvyčajne zdarma ako súčasť operačného systému a poskytuje častokrát väčšiu konfiguračnú flexibilitu. S vývojom CPU sa do istej miery zvyšuje aj výkon RAIDu, avšak niekedy je práve tento výkon potrebný pre iné aplikácie, čo môže byť nevýhodou. Nástroje na ich správu a konfiguráciu sú zvyčajne špecifické pre daný operačný systém, čo vyžaduje viac času administrátorov. Softwarový raid je vhodnejší pre použitie doma alebo v menších firmách. Známy nástroj na konfiguráciu RAIDu v linuxe je napríklad mdadm.

Hardwarový raid je fyzická karta, ktorá samozrejme nie je zdarma. Ich cena však prináša radu výhod, ako odbremenenie CPU a RAM alebo jednoduchšiu správu. Každá karta je však špecifická v závislosti na výrobcovi, čoho následkom je menej dostupná technická špecifikácia. Systém sa stáva závislým na konkrétnej karte, čo môže byť problém v prípade jej chyby a nutnosti obnovy dát pomocou iného hardwaru. Možnosti konfigurácie sa veľmi líšia v závislosti na type karty a typicky aj cene. V prípade chyby výhodu predstavuje jednoduchšia výmena diskov a podpora hot-swap. Hardwarový raid je vhodný najmä pre systémy s vysokou záťažou.

RAID pole je možné rozlíšiť podľa spôsobu zapojenia diskov [6]. Tie bežné sa označujú číslovaním od RAID-0 po RAID-6. Existuje však mnoho iných konfigurácií. Niektoré

[3]Redundant array of independent disks

z nich je možné kombinovať, čoho príkladom je RAID-10. Jedny z najrozšírenejších sú nasledujúce tri typy:

RAID-0 sa vyznačuje nulovou tolaranciou k poškodeniu disku. V praxi takéto zapojenie vyzerá ako keby sme sériovo spojili 2 disky. Jeho výhodou je predovšetkým zvýšenie rýchlosti prístupu k disku a zmenšenie počtu diskových jednotiek - spája menšie disky do jedného väčšieho.

RAID-1 niekedy nazývaný ako zrkadlenie diskov. Disky sú zapojené paralelne a majú rovnaký obsah. Tento spôsob je využívaný v prípade, kedy kladieme dôraz na zachovanie dát v prípade chyby jedného z diskov. Zlyhanie užívateľ nezaznamená, všetky operácie prebiehajú na vrstve pod súborovým systémom.

RAID-5 vyžaduje minimálne 3, bežne sa však používa 5 diskov. V tomto zapojení sú disky reťazené a zároveň je časť z nich využitá na redundanciu. V prípade výpadku niektorého z nich sú stratené dáta dopočítané zo zvyšných diskov. Je vyhľadávaný hlavne kôli svojej nízkej cene, avšak nie je vhodný pre systémy, ktoré kladú dôraz na vysoký výkon.

Systém RAID rieši len problém na úrovni výpadku diskov, nie dostupnosť celého systému (aj keď k nej prispieva). Ak nastane chyba v akomkoľvek inom hardwarovom komponente, nastane výpadok. RAID taktiež nerieši chyby ľudského faktoru a aplikácií a v žiadnom prípade nie je náhradou pravidelného zálohovania.

SAN

SAN[4] je vysoko rýchlostná dedikovaná sieť, ktorá spája rôzne druhy dátových zariadení s asociovanými servrami. Jednoduchšie povedané je to sieť, ktorá spája počítače so zariadeniami na ukladanie dát. Na vybudovanie takejto siete je možné použiť viacero technológií, napríklad Fibre Channel alebo iSCSI. Technológia SAN podporuje zrkadlenie diskov, zálohovanie, obnovu a migráciu dát z jedného úložného zariadenia na iné a zdieľanie dát medzi rôznymi servrami v sieti. Oproti lokálnym diskom má viacero výhod [9]:

- Odstraňuje vzdialenostné limity lokálne pripojených diskov
- Zvyšuje výkon, dáta sú dostupné rýchlosťou v jednotkách Gb/s

[4]Storage Area Network

- Zvyšuje spoľahlivosť umožnením viacerých prístupových ciest k úložisku
- Jednoduchšie možnosti obnovy po havárii. Diskové polia môžu byť zrkadlené do iných lokalít
- Jednoduchšia administrácia a flexibilita, keďže káble a úložiská dát nemusia byť fyzicky presúvané ak ich chceme premiestniť na iný server

NAS

Sieťové úložisko alebo NAS[5] je počítač s pripojením na sieť, ktorý má fungovať ako dátové úložisko pre ostatné zariadenia. NAS zvyčajne nemajú konektory pre display alebo klávesnicu. Namiesto toho sú konfigurovateľné pomocou webového rozhrania. Ich operačný systém je často upravený a okresaný. Úložisko dát sa zvykne rozkladať na hot-swap diskovom poli v RAIDe. Pre prístup k dátam NAS používa protokoly ako NFS[6] na unixových systémoch alebo SMB[7] známy aj pod prezývkou Samba, ktorý je populárny na Windowsoch.

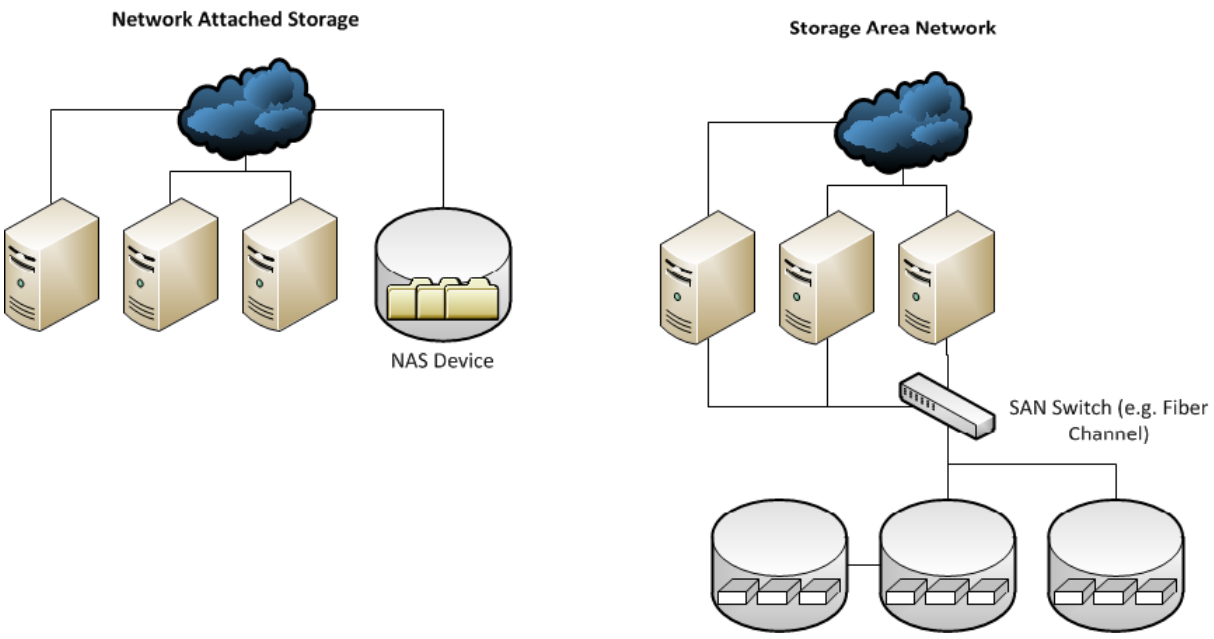
Oproti SAN systémom sú jednoducho konfigurovateľné a ich správa je jednoduchá. NAS nie je použitím limitovaný len ako dátové úložisko pre klientské a serverové stanice. Umožňuje jednoduché a nízkonákladné riešenie dátového úložiska pre servery s rozložením záťaže alebo s toleranciou k výpadku.

Hlavné rozdiely medzi SAN a NAS sa dajú zosumarizovať do troch bodov:[10]

- Pričinok vs disk: NAS poskytuje súborový systém (NFS, SMB), naproti tomu SAN poskytuje blokové zariadenie. Klientski sa tak musia v druhom prípade postarať o vlastný súborový systém.
- Výkon vs cena: SAN je zvyčajne viac výkonný ako NAS, čo sa prejavuje aj v jeho cene. Pretože SAN využívajú technológiu Fibre Channel, poskytuje oveľa vyššie rýchlosti ako IP siete
- Jednoduchosť: NAS zariadenia zvyčajne fungujú priamo po vybalení z krabice alebo s minimálnou konfiguráciou.

Rozdiel v použití v sieťovom zapojení môžme vidieť na obrázku 1.3

[5] Network Attached Storage
[6] Network File System
[7] Server Message Block



Obr. 1.3: Roziel v možnosti sieťového zapojenia NAS a SAN[10]

1.3.2 Dostupnosť aplikácií

Keď máme dátove úložisko dostatočne chránené proti výpadku, je potreba sa sústrediť na vyššiu vrstvu - aplikácie. Tie sú práve nástrojom, ktorý využívajú klienti, takže sú nedeliteľnou súčasťou každého systému. Ich dostupnosť sa dá rovnako ako pri dátovom úložisku zabezpečiť viacerými spôsobmi.

Failover

Failover znamená uvedenie náhradného systému do prevádzky v prípade, že primárny systém zlyhá. Aktuálne kópie všetkých požadovaných dát a aplikácií sú udržiavané na sekundárnom systéme, aby ho bolo možné spustiť v čo najkratšom čase. [11] Failover je dôležitou súčasťou kritických systémov, keďže automaticky presmeruje užívateľské požiadavky na záložný systém, ktorý má rovnakú alebo podobnú funkcionalitu ako ten pôvodný.

Load balancing

Pod pojmom load balancing sa rozumie rovnomerné rozloženie záťaže na dostupné servry v sieti, prípadne na disky v SAN. Vysoko dostupné aplikácie môžu fungovať na tomto princípe rozložením požiadavok medzi viaceré servery, ktoré v prípade výpadku niektorého

z nich preberú jeho prácu. Podobne funguje aj Linux Virtual Server, ktorý popíšem bližšie v kapitole 3.

1.3.3 Dostupnosť siete

Nedeliteľnou súčasťou systémov vysokej dostupnosti je zabezpečenie dostupnosti siete. To sa dosahuje napríklad použitím záložných sieťových prvkov či náhradných pripojení. Aj keď je táto oblasť vysokej dostupnosti dôležitá, v práci sa venujem dostupnosti súborových systémov a aplikácií, preto sa riešením siete nebudem zaoberať.

Kapitola 2

Technológie

Na realizáciu praktickej časti potrebujem viacero komponentov (vrstiev), ktoré na seba naväzujú. Táto kapitola je rozdelená podľa jednotlivých funkčných vrstiev. Pri každej vrstve popíšem nástroje, ktoré som využil ako aj niektoré z alternatív. Na záver kapitoly zhrniem niektoré termíny, ktoré majú spojitosť s vysoko dostupnými riešeniami.

2.1 Kominukačná vrstva

Kominukačná vrstva poskytuje klientom informácie o stave (prítomnosti/neprítomnosti) procesov na jednotlivých strojoch. Ak by sme mali len 2 nódy nebol by systém komunikácie príliš zložitý. Avšak pri zapojení dodatočných strojov komplexnosť tejto vrstvy stúpa. Pri vývoji programov, ktoré tieto funkcie požadujú je vhodnejšie využiť už otestované riešenie ako vyvíjať niečo čo už existuje. Pre názornosť zložitosti, Corosync obsahuje 55000 riadkov C kódu.

V súčasnosti existuje viacero projektov, z nich si predstavíme Heartbeat, Corosync a CMAN, ktoré implementujú nasladujúcu funkcionalitu:

- spoľahlivý prenos správ medzi nódami
- notifikácie pri prechode stroja do on-line/off-line módu
- udržanie rovnakého zoznam strojov v celom klastri

Pri porovnaní Heartbeat a Corosync zistíme, že oba sú dnes podporované a využívané v produkčných prostrediach, avšak ak sa rozhodujeme ktoré riešenie je vhodnejšie, väčšina

odborníkov odporúča Corosync (prevažne kvôli výhľadom do budúcnosti projektov) [12]. CMAN je trochu špecifický, avšak tiež ustupuje v prospech Corosyncu.

2.1.1 Heartbeat

Tento démon vznikol v roku 1999. Vo svojich začiatkoch podporoval len 2 nódy, mal príliš jednoduchý model a nedokázal detekovať výpadky na úrovni služieb. V roku 2003 ale začal Andrew Beekhof pracovať na komplexnejšom systéme a verzia 2.0.0 už obsahovala aj prvé CRM[1] s názvom Pacemaker [13].

Heartbeat je súčasťou Linux-HA projektu, ktorý okrem neho vyvíjal aj ďalšie komponenty potrebné pre postavenie vysoko dostupných klastrových systémov, vrátane veľkého množstva resource agentov pre rôzne aplikácie, knižníc a nástrojov ako Stonith, notifikačného systému a LRM[2].

Všetky tieto komponenty sa zo začiatku distribuovali ako kompletné riešenie, teda balík mal všetko potrebné aby mohol klaster plnohodnotne fungovať. Staral sa o komunikáciu, obsahoval Resource Agentov aj CRM. Po čase sa jednotlivé komponenty oddelili (napríklad Pacemaker vo verzii 2.1.14) a teraz sa s názvom Heartbeat označuje už len program zabezpečujúci komunikačnú vrstvu.

Heartbeat je starší projekt a jeho vývoj je momentálne pozastavený. Linbit prebral zodpovednosť za jeho udržiavanie, ale vyhlásil že neplánuje pridávať žiadne nové prvky. Z vyjadrenia autorov vyplýva, že bude udržiavaný pokiaľ to bude mať zmysel (pravdepodobne dovtedy, kým ho corosync nenahradí) [14].

2.1.2 Corosync

Open Source projekt, ktorý sa pôvodne nazýval OpenAIS. Vývojári narazili na problém, kedy sa vývoj začal sústreďovať prevažne na vytváranie infraštruktúry pre rôzne API postavené na OpenAIS. Založili teda nový projekt - Corosync. Použili v ňom približne 80% pôvodného kódu (jeho stabilné časti - jadro). V roku 2011 vývojári úplne zastavili prácu na OpenAIS a prešli k vývoju Corosyncu. Funcionalita OpenAIS ostala už len vo forme pluginov. Práve tie umožňujú prácu so súborovými systémami GFS2 a OCFS2 [15].

Corosync je aktívne vyvíjaný (nové vývojové verzie sú vydávané v priebehu týždňov)

[1] Cluster Resource Manager

[2] Local Resource Manager

a je momentálne najvhodnejším riešením pre zabezpečenie komunikácie pre Pacemaker [13]. Taktiež je podporovaný firmami veľkých mien ako je Red Hat alebo Novell/SUSE.

2.1.3 CMAN

CMAN je súčasťou balíka RHCS. Zabezpečoval infraštruktúru, ktorú ďalej využíval RGManager. Na klastrovom samite v roku 2008 sa rozhodlo, že nemá zmysel udržiavať vlastné riešenie, pre ktoré sa nedarí získať komunitu. Red Hat sa teda rozhodol podporiť vývoj Corosyncu, čo len zdôrazňuje predpoklad ďalšieho vývoja CMANu. RHEL[3] ním už vo verzii 5 nahradil tento komponent, vtedy ešte pod menom OpenAIS. CMAN stále existuje, avšak už len ako quorum modul pre Corosync [16].

2.2 Manažér procesov

Manažér procesov, taktiež nazývaný CRM zodpovedá za spúšťanie a zastavovanie procesov. Obsahuje logiku, ktorá zabezpečuje nielen že proces je spustený, ale aj to, že nebeží na viacerých miestach zároveň a predchádza tak poškodeniu dát. Tieto nástroje umožňujú definovať a nakonfigurovať jednotlivé nódy klastra a následne monitorovať procesy, ktoré sú na nich spustené. V prípade výpadku sú zodpovedné za presun procesu na iný, funkčný server v čo najkratšom čase. Tiež umožňujú definovať presné pravidlá, obmedzenia a priority, podľa ktorých budú procesy migrovať ako aj nastaviť ich skupiny a závislosti. Dva z CRM systémov dostupných zdarma sú Pacemaker a RGManager.

2.2.1 Pacemaker

Názov programu vznikol odvodením od prístroja ktorý kontroluje a reguluje činnosť ľudského srdca podobne ako Pacemaker udržiava v chode jednotlivé procesy v počítači. Jeho úlohou je dosiahnutie čo najvyššej dostupnosti služieb. Detekuje chyby jednotlivých procesov alebo celých serverov a pokúša sa ich spustiť v inom - funkčnom prostredí. Pacemaker sa neobmedzuje len na spúšťanie procesov - dokáže spustiť čokoľvek, čoho štart sa dá ovládať skriptom, napríklad nastavenie IP adresy, spustenie webového servra alebo pripojenie disku [13].

Skripty ktoré využíva sa delia do dvoch hlavných kategórií:

[3] RedHat Enterprise Linux

OCF skripty boli napísané podľa špecifikácii Open Cluster Framework. Musia podporovať príkazy start, stop, monitor a meta-data. Prakticky sú rozšírením LSB skriptov

LSB skripty sú štandardné spúštacie skripty, musia teda vyhovovať LSB špecifikácii. Podporujú štandardné príkazy start, stop a status

Ako komunikačné riešenie využíva už vytvorené programy, aktuálne podporuje Heartbeat a Corosync. Aj keď Pacemaker pôvodne vznikol ako CRM pre Linux-HA, ktoré zodpovedalo aj za vznik Heartbeatu, dnes sa uprednostňuje použitie Corosyncu. Taktiež pre využitie časti funkcionality Pacemakeru vyžadujúcej OpenAIS (napríklad práca s GFS2 a OCFS2) musíme ako komunikačnú vrstvu použiť Corosync.

2.2.2 RGManager

RGManager je súčasťou balíka RHCS. Z vonkajšieho pohľadu poskytuje podobnú funkcionalitu ako Pacemaker. Umožňuje administrátorovi definovať, konfigurovať a monitorovať jednotlivé procesy v klastri, prípadne ich skupiny. Je to dobre otestované riešenie, jeho konfigurácia nie je príliš zložitá. Avšak Pacemaker je pri správe procesov flexibilnejší. Andrew Beekhof ktorý stojí za jeho vývojom sa v diskusiách vyjadril, že plánom Red Hat je rgmanager v priebehu niekoľkých rokov Pacemakerom nahradiť [17].

2.3 Zdieľané blokové zariadenie

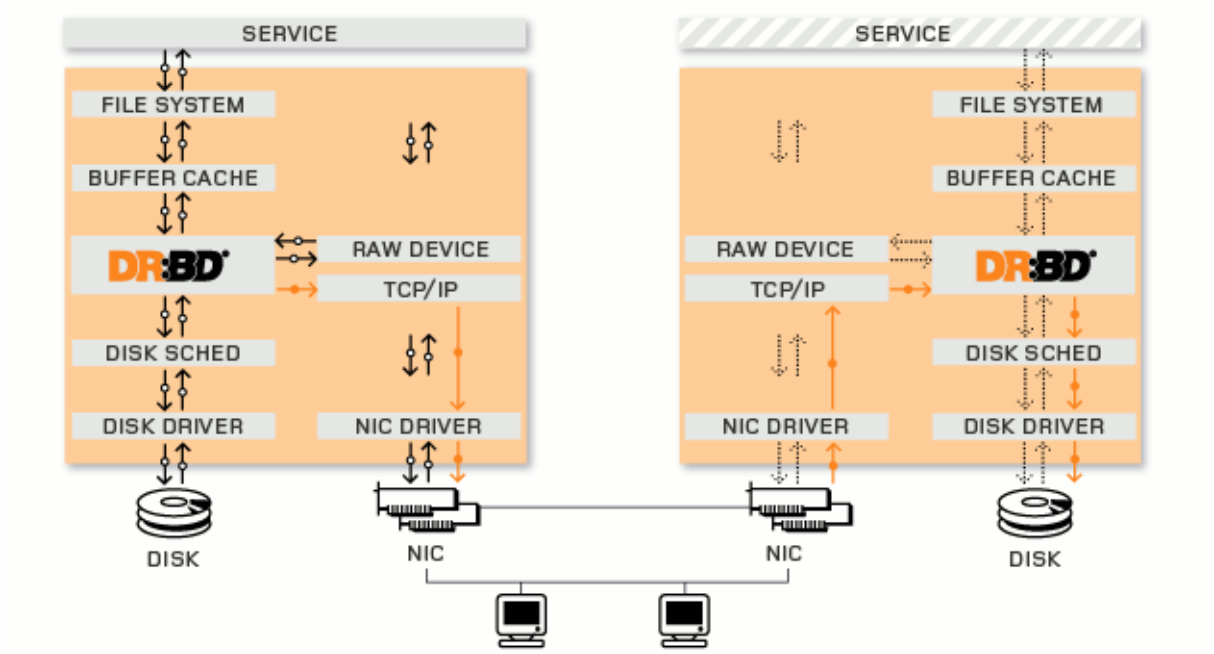
Súborové systémy využívajú pre ukladanie dát blokové zariadenie - zvyčajne pevný disk. Ten sa však bez použitia ďalších technológií nachádza v jednej fyzickej lokalite, čo vedie k problémom pri zabezpečení jeho dostupnosti. Preto zvniklo viacero technológií, ktoré umožňujú blokové zariadenia zdieľať aj na väčšie vzdialenosti.

Je dôležité poznamenať, že technológie zmieňované v tejto sekcii sa zameriavajú na zdieľanie blokového zariadenia - nie súborového systému. Takéto zariadenia je možné pomocou siete pripojiť k počítaču a následne ho využívať ako lokálne zariadenie (disk). Pred jeho použitím je teda nutné vytvoriť partície a súborový systém.

Príkladom tohoto prístupu teda nie je umožnenie súčasného prístupu 3 klientov na 1 súborový systém, ale poskytnutie 3 rôznych častí disku jedného zariadenia 3 rôznym klientom.

2.3.1 DRBD

Pojem DRBD sa často vyskytuje práve v spojení s vysokou dostupnosťou a súborovými systémami. Nie je však ďalším súborovým systémom ani priamo nezabezpečuje vysokú dostupnosť aplikácií. Predstavuje riešenie úložiska dát zrkadliace obsah blokových zariadení (hard-diskov, partícií, logických jednotiek) medzi dvoma serverami. Funkciu DRBD jednoducho vysvetlíme na príklade dvojitého papiera, s ktorým sa bežne stretávame na poštových poukážkach. Všetko čo napíšeme na vrchný papier sa automaticky objaví aj na tom spodnom. Rovnako funguje aj DRBD, len namiesto bieleho papiera používa diskové partície a čierny kopírovací papier nahrádza sieťovú kartu. Jeho presné začlenenie do systému ukazuje obrázok 2.1. Na fyzickom disku oboch serverov vytvoríme partície, tie podsunieme DRBD, ktoré z oboch vytvorí jedno spoločné zariadenie. Na neho sa potom nainštaluje súborový systém. Ďalšia práca s ním prebiaha rovnako ako s lokálnym súborovým systémom. Vo výsledku ho je môžné prirovnať k RAID-1, jediným rozdielom je, že zrkadlenie neprebieha v rámci jedného serveru ale je naňho použitá dedikovaná sieť.



Obr. 2.1: Spôsob prenosu dát systémom DRBD [18]

Prakticky je systém využiteľný najmä v prípade keď potrebujeme zabezpečiť dostupnosť dát pri hardwarovej chybe servru, ktorá presahuje výpadok disku (na ktorý by RAID-1 stačil) - napríklad vyhorenie zdroja alebo poškodenie základnej dosky. Zrkadlené servery

vďaka využitiu siete nemusia byť ani v jednej miestnosti, takže systém je odolný aj voči katastrofám ako napríklad požiar.

Keďže zmena stavu nódov a pripájanie systému nie sú vykonávané automaticky a ich pripájanie k systému ručne je kontraproduktívne (reačná doba je obvykle v rozpätí hodín) je nutné využiť nástroj, ktorý to zabezpečí. Tým je práve CRM zmienené v sekcii 2.2. Ten umožňí nielen pripojenie súborového systému ale automatizuje spúštanie a vypínanie služieb, presun zdieľanej IP adresy na primárny nód a veľa iného na základe vopred definovaných pravidiel.

Drbd je v dnešnej dobe obľúbeným nástrojom administrátorov, čo dokazuje aj fakt, že počet jeho inštalácií každý mesiac presiahne 10 000 [18]. Konfiugrácia umožňuje použitie práve dvoch nódov, ktoré sú definované ako primárny-primárny alebo primárny-sekundárny. Aktuálnou verziu je 8.4.1.

DRBD môže fungovať v dvoch módoch zápisu na disk (definovaných ako „protokol“), ktoré sa líšia v rýchlosti zápisu dát a miere bezpečnosti ktorú požadujeme:

Synchrónne Pri synchrónnom prenose je aplikácii oznámené úspešné uloženie až vo chvíli keď sú dáta uložené na primárnom servri a skopírované cez sieť na sekundárny disk

Asynchrónne Pri asynchrónnom prenose je úspešné uloženie potvrdené už vo chvíli zápisu na primárny disk. Asynchrónne prenosy sú síce rýchlejšie ale menej bezpečné. Využívané sú keď je prenosová rýchlosť siete nedostačujúca.

Primárny/Sekundárny

Táto konfigurácia funguje na princípe dvoch zariadení (nódov), pričom jeden (primárny) je pripojený k systému a s jeho obsahom sa dá normálne pracovať zatiaľ čo druhý (sekundárny) je odpojený, nedá sa pripojiť a je využívaný len na kopírovanie dát z primárneho disku. V prípade výpadku sa po vypršaní timeoutu nastaví neprístupný nód do stavu Unknown, druhý je povýšený na primárny (pomocou CRM) a je pripojený súborový systém. Takto zabezpečíme že aj na záložnom serveri budú vždy najnovšie dáta.

Primárny/Primárny

Od verzie 8.0 je možné nakonfigurovať oba nódy ako primárne. To je využiteľné hlavne pre systémy, na ktorých využívame rozloženie záťaže (load-balancing). Takáto konfigurácia

umožňuje súčasný prístup k dátam na oboch nódoch, čo môže zvýšiť rýchlosť. Keďže sú oba nódy využívané zároveň, nie je možné použiť bežný súborový systém ako napríklad ext3. To by po chvíli viedlo k poškodeniu dát. Pri tomto nastavení musíme použiť súborové systémy ako GFS2 alebo OCFS2, ktoré používajú ochranný mechanizmus na zabránenie poškodenia dát súčasnou zmenou z viacerých miest.

2.3.2 iSCSI

Small Computer Systems Interface je populárna sada protokolov pre komunikáciu predovšetkým úložných zariadení. Protokol definuje dva rôzne typy zariadení, initiators a targets. Initiators je názov pre klientov. Tí nadväzujú komunikáciu a zasielajú príkazy (požiadavky). Targets sú servery (úložné zariadenia), ktoré na požiadavky odpovedajú a vykonávajú zadané príkazy.

iSCSI využíva SCSI protokol pre prácu s blokovými zariadeniami, avšak na komunikáciu využíva TCP/IP siete. Tieto siete pritom nemusia byť dedikované, môže ich zdieľať inými sieťovými aplikáciami.

Všetky zariadenia v iSCSI sieťach (klienti aj servery) majú pridelené adresy. Príklad zvyčajného formátu je iscsi.com.acme.sn.8675309. Tieto adresy musia byť unikátne. iSCSI poskytuje tiež metódy pre autentizáciu a vyhľadávanie.

iSCSI je protokol poskytujúci podobnú funkcionalitu ako Fibre Channel bez nutnosti budovania dodatočnej infraštruktúry. Pri využívaní aplikácií vyžadujúcich vysoký výkon úložných zariadení však môže byť technológia Fibre Channel vhodnejšia [19].

2.3.3 Fibre Channel

Fibre Channel (FC) bol vyvinutý ako odpoveď na stále sa zvyšujúcu potrebu vysoko rýchlostného riešenia, ktoré by umožnilo prenášať veľký objem dát. Pri stále sa zvyšujúcej rýchlosti procesorov a periférnych zariadení začali Ethernet a SCSI obmedzovať celkový výkon systémov.

Fibre Channel je sadou štandardov, ktoré definujú technológiu prenosu dát. Jeho prenosová rýchlosť sa každou generáciou znásobuje (1Gb/s, 2Gb/s, 4Gb/s a dnes 8Gb/s) a je možné ho použiť na vzdialenosti do 10 kilometrov. Je podobný protokolom Ethernet alebo SCSI, prináša však vyššiu rýchlosť a vzdialenosť.

Fibre Channel definuje len metódu transportu dát, nezameriava sa na ich typ alebo obsah. Tým sa stáva veľmi flexibilným, keďže môže prenášať dáta medzi dvoma počítačmi, počítačom a periférnymi zariadeniami (ako diskové polia a tlačiarne) alebo dvoma periférnymi zariadeniami (pri zálohovaní na páskové mechaniky).

Pomenovanie Fibre Channel môže byť zavádzajúce, pretože technológia nie je limitovaná využitím optických vkáken, môžu byť použité aj medené vodiče [20].

Použitie Fibre Channel je však nákladnejšie ako iSCSI. Vyžaduje inštaláciu HBA karty pre každý server, využitie Fibre Channel switchu a samostatnú kabeláž. Tiež je zložitejšie spravovať dve siete, ethernetovú LAN pre užívateľov a Fibre Channel SAN pre úložisko.

2.4 Súborové systémy

Bežné súborové systémy, ktoré používame na klientských staniciach, ako NTFS vo windowse alebo ext3, ReiserFS či XFS v linuxe neboli vyvinuté s ohľadom na riešenie vysokej dostupnosti dát. Kapacity pevných diskov sa v poslednej dobe značne zvyšujú, avšak ich rýchlosť rastie oveľa nižším tempom. Preto bolo potrebné vyvinúť technológie umožňujúce rozloženie záťaže pri prístupe k dátam a zvýšenie ich bezpečnosti.

Svojou funkcionalitou spomedzi nich vyniká súborový systém ZFS vyvinutý spoločnosťou SUN. Ten umožňuje vymoženosti ako vytváranie snapshotov, konfiguráciu RAIDu alebo manažment úložiska. ZFS využíva blokové zariadenie tak, že ho priradí do „poolu“, kde sa týchto zariadení môže nachádzať viacero. Z ich kombinovanej kapacity potom vytvára dátové úložisko dostupné užívateľom.

Súborové systémy sa dajú rozdeliť do niekoľkých kategórií.

2.4.1 Sieťové súborové systémy

Sieťový súborový systém (NFS) sa vyznačuje spôsobom prístupu k súborom prostredníctvom počítačovej siete. NFS umožňuje pracovať so súbormi rovnako ako keby boli uložené lokálne. Súbory sa však fyzicky nachádzajú na inom počítači a pristupujeme k nim pomocou rôznych služieb (SMB). Na vzdialenom počítači sa zvyčajne používa bežný súborový systém. Tento spôsob umožnuje administrátorom usporiadať dáta do centralizovaných serverov na sieti.

Špeciálnym prípadom NFS sú distribuované súborové systémy, ktoré popíšem v ďalšej

sekcii.

2.4.2 Súborové systémy so zdieľaným diskom

Súborové systémy so zdieľaným diskom umožňujú počítačom v klastri súčasné využívanie blokového zariadenia, ktoré je zdieľané napríklad pomocou FC, iSCSI alebo DRBD. Čítanie a zápis prebieha podobne ako na lokálnom súborovom systéme, avšak je využívaný zamykací modul, ktorý tieto operácie koordinuje a udržuje tak konzistenciu systému. Zmeny vykonané na jednom stroji sa tak okamžite prejavia aj na ostatných. To umožňuje výpadok niektorého z nódov bez ovplyvnenia dostupnosti dát ostatných.

Zdieľané súborové systémy môžu byť symetrické, kde sú metadáta uložená priamo na nódoch klastra alebo asymetrické kde sa ako úložisko používa centralizovaný server.

Zo súborových systémov so zdieľaným diskom som vybral GFS2 a OCFS2, ktoré sú voľne dostupné a aktuálne vyvíjané a použijem ich pri testovaní v praktickej časti.

Global File System 2

Vývoj GFS začal na univerzite v Minnesote. Pri projekte na simuláciu morských prúdov vznikli dve požiadavky - potreba ukladať veľké množstvo dát a umožniť súčasný prístup k nim z viacerých serverov zároveň. GFS bol následne zakúpený firmou Sistina, ktorú spolu s ním prevzal v roku 2003 Red Hat.

GFS2 je 64-bitový symetrický súborový systém, vhodný pre aplikácie vyžadujúce SAN v ktorej každý server v klastri má rovnaký prístup k úložisku. Všetky nódy využívajú identický software a môžu so súborovým systémom vykonávať rovnaké operácie.

Systém je žurnálovací, každý nód si udržiava vlastnú kópiu. Na obmedzenie prístupu a uchovanie integrity systému využíva manažéra uzamykania DLM. GFS2 je možné použiť aj na samostatnom serveri [21].

Oracle Cluster File System 2

OCFS je už od svojho počiatku v roku 2003 vyvíjaný firmou Oracle. Je dostupný pod GNU GPL2 licenciou. Cieľom tohoto projektu bolo poskytnúť podobný výkon ako majú lokálne súborové systémy, a začleniť ho do hlavnej vývojovej vetvy linuxového jadra, čo sa podarilo v roku 2006. V súčasnosti umožňuje vytvárať klastre s maximálne 32 nódmi.

Súborový systém je možné pripojiť rovnako ako lokálny. Nelimituje počet súborov a umožňuje vytvoriť súborový systém s veľkosťou až 4 PB oproti GFS2 s 25 TB. Dôraz je kladený tiež na jeho jednoduchú správu.

Aktuálne je využívaný firmou Oracle napríklad pri virtualizácii (Oracle VM) a databázových klastroch (Oracle RAC) [22].

2.4.3 Distribuované súborové systémy

Účelom distribuovaných súborových systémov (DFS) je umožniť užívateľom nachádzajúcim sa v rôznych lokalitách zdieľať dáta a úložný priestor použitím spoločného súborového systému. DFS je implementovaný ako súčasť operačného systému každého pripojeného počítača. Klientské stanice nemajú priamy prístup k blokovému zariadeniu, k súborom pristupujú pomocou siete. Dáta môžu byť rozložené na viacerých serveroch, čo umožňuje paralelný prístup. Tým je dosahovaná vysoká rýchlosť ich čítania a zápisu [23].

Ako je už pri softwarových produktoch zvykom, distribuovaných súborových systémov existuje veľké množstvo. Vybral som z nich dva, ktoré sú voľne dostupné a odolné voči výpadku časti klastra.

GlusterFS

GlusterFS je súborový systém, ktorý agreguje úložisko viacerých počítačov do jedného celku. Podporuje lokálne úložiská alebo pripojené pomocou iSCSI, Fibre Channel a Infiniband.

Na rozdiel od iných distribuovaných súborových systémov nevyužíva metadáta súborov. Namiesto nich lokalizuje dáta v klastri pomocou hashovacieho algoritmu. To je považované za jednu z veľkých výhod vedúcich k zníženiu rizika straty dát alebo ich poškodeniu.

GlusterFS je súčasťou platformy vyvinutej firmou Gluster. Tá poskytuje GUI nástroje na jeho inštaláciu a administráciu. Gluster dokáže dáta zrkadliť a môže byť nakonfigurovaný ako kompletne redundantné riešenie. Gluster podporuje rôzne linuxové operačné systémy. Využitie nachádza napríklad v cloude alebo pri archivácii. GlusterFS je open-source, dostupný pod GNU AGPL3 licenciou [24].

Lustre

Vývoj Lustre začal z iniciatívy vlády USA, ktorá ho aj financovala. V súčasnosti ho zastrešuje Oracle, ktorý ho prebral spolu s firmou Sun Microsystems. Lustre je open-source, distribuovaný pod GNU GPL licenciou.

Pri jeho návrhu bolo cieľom vytvoriť škálovateľný a vysoko výkonný súborový systém pre superpočítače. Lustre tento cieľ dosiahol, dnes je využívaný na 15 z 30 najvýkonnejších počítačoch sveta. Je populárny predovšetkým vo výskume, finančnom a mediálnom sektore [25].

Lustre je objektovo založený systém pozostávajúci z troch základných komponentov - metadátového serveru (MDS), serverov na úschovu objektov (OSS) a klientov. Medzi jeho hlavné prednosti patria [26]:

Škálovateľnosť Rozšírenie súborového systému je jednoduché - úložisko môže byť pridávané podľa potreby prostredníctvom LAN alebo WAN

Rýchlosť Vysoká rýchlosť je dosahovaná paralelným prístupom medzi klientami a servermi. Súčet rýchlostí prístupu k dátam môže dosahovať až stovky GB/s

Dostupnosť Redundantné servery a failover úložísk zabezpečujú vysokú dostupnosť

2.5 GUI

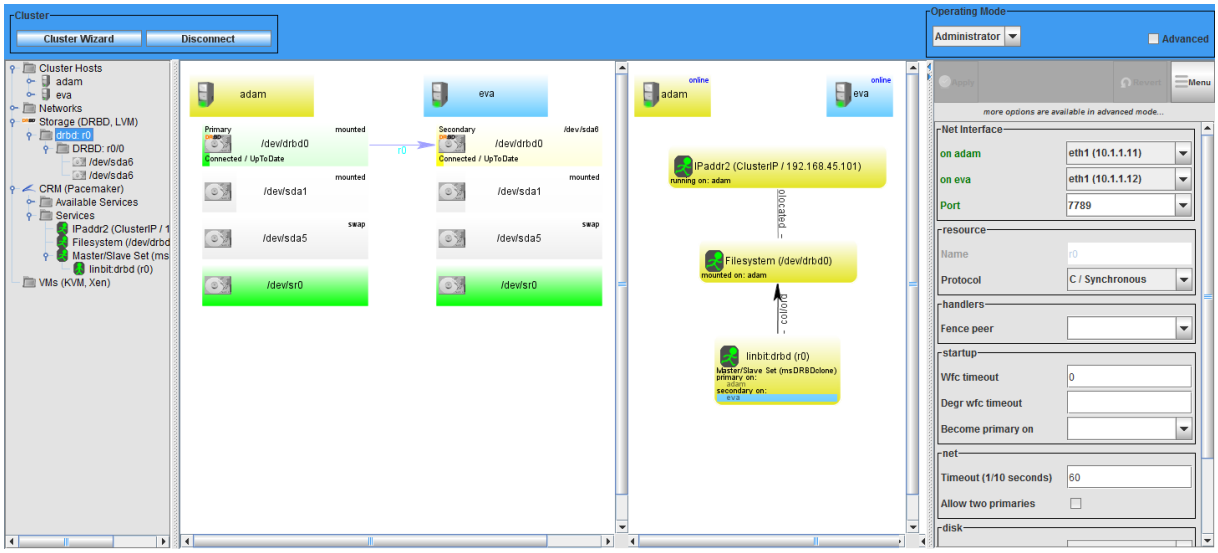
GUI nástroje uľahčujú konfiguráciu klastra, avšak nie sú natoľko intuitívne, aby bolo možné klaster nakonfigurovať bez základného prehľadu o tom ako funguje. Nasledujúce nástroje slúžia na ovládanie CRM bez použitia príkazového riadku. Dva z nich, DRBD-MC a Pacemaker GUI som pri konfigurácii klastra vyskúšal.

2.5.1 DRBD Management Console

Tento nástroj je vyvinutý v Jave a jeho možnosti sú najširšie. Jeho funkcionalitu môžme rozdeliť do 3 oblastí:

- Inštalácia klastra. Dokáže automaticky nainštalovať Heartbeat, Corosync alebo Pacemaker

- Správa DRBD a LVM. Umožňuje vytváranie zariadení, ich následnú konfiguráciu a správu klastra
- Správa CRM. Umožňuje konfiguráciu globálnych nastavení, vytváranie pravidiel pre konkrétne služby, ich migráciu a veľa ďalšieho



Obr. 2.2: Grafické rozhranie DRBD Management Console

Ku klastru sa pripája pomocou protokolu SSH, cez ktorý posiela konfiguračné príkazy. Rozhranie pôsobí na prvý pohľad profesionálne, ale s neúplnými znalosťami klastrových systémoch som sa v ňom miestami trochu strácal. Osobne by som ocenil rozdelenie prostredia do módov pre začiatočnikov a pokročilých, ako to je v Pacemaker GUI.

2.5.2 Pacemaker GUI

Tento nástroj je známy pod viacerými menami, napríklad heartbeat gui alebo pacemaker python gui. Na serveri je spustený démon, na ktorý sa klient pripája. Nástroj je určený výhradne pre ovládanie Pacemakeru. Gui poskytuje 3 módy pracovného prostredia jednoduché, expertné a hackerské, v závislosti ako detailne chceme klaster konfigurovať. Problémom môže byť, že pre jeho spustenie z windowsov musíme mať nastavený X forwarding. Jeho autorom je Andrew Beekhof.

2.5.3 HAWK

Hawk vznikol ako náhrada v prípadoch kedy z nejakého dôvodu nechceme alebo nemôžme využiť niektoré z predchádzajúcich GUI. Je prístupný pomocou webového rozhrania, čo môže byť výhodou napríklad v prípade prísnej firemnej politiky ohľadom SSH prístupov. Jeho funkcionalita nepokrýva všetky oblasti a aj samotná stránka programu hovorí, že sa ľahko môžme dostať do situácie kedy budeme musieť použiť príkazový riadok.

2.5.4 Conga

Rozhranie Conga je distribuované s High Availability Add-On od Red Hatu. Pozostáva z démona Ricci, bežiaceho na servri, ktorý sa stará o distribúciu klastrovej konfigurácie a aplikácie Luci, ktorá poskytuje užívateľské rozhranie.

2.6 Split-brain

Pojem split-brain neoznačuje priamo technológiu, definuje však problém, ktorý z jej využitia vyplýva. Preto popíšem čo sa pod názvom split-brain skrýva a zmienim metódy, ktoré tento problém riešia.

Po prvotnom zoznámení s funkcionalitou Pacemakeru a iných HA riešení môže vzniknúť otázka, či je to naozaj také jednoduché. Veď len spúšťa, zastavuje a kontroluje procesy podľa vopred nakonfigurovaných pravidiel. Pri realizácii ale narazíme na problém s názvon split-brain. To je situácia, kedy sa kvôli výpadku komunikácie klaster rozdelí na 2 alebo viacero častí, ktoré začnú fungovať nezávisle na sebe. Obe časti sa pritom domnievajú, že práve tá druhá je nefunkčná. V lepšom prípade sa nič nestane, v horšom ostane disk plný poškodených dát.

Jadro problému spočíva v neschopnosti rozlíšiť či je počítač vypnutý alebo len prestal komunikovať so svojim okolím. Aký je v tom rozdiel? Predstavme si nasledujúcu situáciu [27]:

Máme 3 počítače, Adam, Eva a Tom zapojené v klastri. Tom má pripojené vzdialené úložisko dát (diskové pole). Zrazu ale Tom prestane fungovať a nedá sa naňho pripojiť. Môžme bezpečne pripojiť diskové pole na Adama a pokračovať v prevádzke? Čo ak je Tom stále zapnutý a na disk zapisuje? Ocitneme sa v situácii kedy na disk zapisujú súčasne dva počítače, a to môže ľahko skončiť znehodnotením dát na ňom.

Jednoduchým riešením by bola redundancia komunikácie medzi nódmi, napríklad záložné wifi spojenie. Stále ale môže nastať situácia, kedy sa stane niečo nepredvídané a server prestane komunikovať. Preto je potrebné nastaviť fencing.

2.6.1 Fencing

Fencing je riešenie postavené na „oplotení“ chybného nódu. To mu zabráni pristupovať k zdieľaným prostriedkom, v tomto prípade k disku. Vyriešil sa tak problém kedy bolo nemožné odpovedať na otázku či je počítač vypnutý alebo nedostupný - teraz na odpovedi nezáleží.

Riešenie pomocou odstrihnutia sa dá realizovať dvoma spôsobmi:

Na úrovni zdrojov Týmto prístupom zabránime chybnému počítaču, aby pristupoval k jednotlivým zdrojom, ktoré využíva. Napríklad ak by bol zdieľaný disk pripojený

pomocou switchu, bolo by možné zakázať prístup k tomuto disku na úrovni switchu.

Na úrovni nódov Pri tomto spôsobe nie je potrebné sa zaoberať tým, aké zdroje nód využíva, alebo ako mu k nim zabrániť prístup - zvyčajne ho jednoducho reštartujeme. Je to jednoduchšie, aj keď trochu drastickejšie riešenie, realizované pomocou techniky nazývanej STONITH[4].

Dôležitým znakom techniky oplotenia je, že k jeho realizácii nepotrebujeme akékoľvek informácie z chybného nódu, alebo jeho spoluprácu.

Pokračujem v príklade: Adam a Eva použijú fencing a zabránia tak Tomovi aby zapisoval na zdieľaný disk. Adam si pripojí disk a môže pokračovať v bežnej prevádzke. Avšak čo zatiaľ robí Tom? Ak je stále zapnutý tak isto zistil že Adam a Eva sú nedostupní. Rovnako ako oni sa snaží oplotiť chybné prvky (z jeho pohľadu Adama a Evu). Ktorý z nich bude prvý? Týmto spôsobom je možné sa dostať až do cyklu, kedy sa nódy budú pri spustení navzájom reštartovať. Takéto nepredvídateľné správanie je nepriateľné, a tak je nutné využiť ďalšiu techniku - hlasovacie quorum.

2.6.2 Quorum

Quorum je technika, ktorou je možné zistiť, ktorá časť pôvodného klastra by mala ostať aktívna pri výpadku - je jej povolené zapínať služby. Najjednoduchšou technikou je vybrať tú skupinu, ktorá obsahuje viac počítačov. Toto riešenie so sebou ale prináša jeden problém. Čo v prípade ak sa klaster skladá len z dvoch serverov? Pri výpadku komunikácie medzi nimi žiaden nemá väčšinu, takže žiaden nemôže spúšťať procesy. Nutnosť získať quorum sa síce v konfigurácii dá vypnúť, ale nedoporúča sa to. Existujú hardwarové aj softwarové metódy, ktoré tento problém riešia.

Metód je veľa, napríklad:

- Hardwarovou metódou môže byť vyhradenie partície na externom disku dostupnej obom nódom. Ktorý nód dokáže túto partíciu pripojiť, ten sa považeje za funkčný
- Softwarovým riešením môže byť quorum démon. Existuje viacero implementácií, napríklad od Linux-HA. Funguje podobne ako prvá metóda, ale prináša niektoré výhody. Napríklad dokáže spoľahlivo fungovať pri väčších geografických vzdialenostiach

[4]Shoot The Other Node In The Head

- Riešením môže byť aj uprednostnenie toho nódu, ktorý dokáže komunikovať s IP adresou na vonkajšej sieti. Je to síce najjednoduchšie, ale v prípade že zlyhá komunikácia medzi dvoma nódmi a pripojenie na internet ostane funkčné problém ostáva nevyriešený

V mojom príklade Adam a Eva tvoria väčšinu, získali quorum a môžu spúšťať procesy. Tom aj bez toho aby komunikoval vie, že má len 1 hlas z 3, takže (podľa konfigurácie) môže byť reštartovaný, prípadne počkať na zásah správcu [27].

2.6.3 Stonith

Pod touto skratkou vystupuje zariadenie, fungujúce presne podľa svojho názvu - jeho úlohou je zastreliť druhý nód. To znamená, že ho čo najrýchlejšie vypne alebo odpojí od zdrojov tak, aby nemohol vplývať na funkčnosť zvyšku klastra. Odstrelený nód sa totiž po reštarte dostane v rámci klastra do submisívneho postavenia a tým mu zabránime replikovať poškodené dáta na ostatné nódy [28].

Realizovať STONITH môžme viacerými spôsobmi, z ktorých je dobré vybrať ten čo najmenej závisiaci na zvyšku systému. Delia sa do 5 základných kategórií [29]:

1. UPS (Uninterruptible Power Supply) Poskytujú kontrolu záťaže a monitorovanie zariadení. Taktiež umožňujú individuálnu kontrolu napájania jednotlivých serverov

2. PDU (Power Distribution Unit) Zdroj napájania cez ktorý sú pripojené ostatné zariadenia. V prípade výpadku zabezpečuje dočasnú dodávku energie

3. Blade power control zariadenia sú vhodným riešením ak je klaster postavený na niekoľkých blade servroch. Musí ale byť schopný ovládať jednotlivé počítače

4. Lights-out zariadenia sú menej kvalitné ako UPS, pretože zdieľajú zdroj napájania so svojím hostiteľom

5. Testovacie zariadenia sú zvyčajne viac zhovievavé k hardwaru, vypínajú počítač „jemnejšie“. Mali by sa však využívať len pre testovacie účely. V produkčnom prostredí ich nahradzujú skutočné STONITH zariadenia.

Väčšina pluginov umožňuje reštartovať alebo vypnúť chybný nód. Nie vždy je však vhodné štartovať CRM pri štarte OS, pretože je možné dostať sa do stavu kedy nód

naštartuje a začne plne fungovať bez toho, aby sme mali šancu diagnostikovať čo bolo príčinou výpadku.

Kapitola 3

Možné riešenia

Spôsobov ako postaviť vysoko dostupné riešenie je veľa, dostupných zdarma aj komerčných. V tejto kapitole sa budem venovať len niektorým z nich, zameriam sa hlavne na tie, ktoré som v praktickej časti využil. V práci sa nesústredím na výber toho najvhodnejšieho riešenia, preto niektoré z nich zmienim len okrajovo.

3.1 Linux-HA

Linux High-Availability je nekomerčným projektom. Pozostáva z viacerých komponentov, z ktorých každý zabezpečuje inú časť funkcionality. Najhlavnejšími z nich sú [30]:

- démon zabezpečujúci komunikáciu serverov v klastri na sieťovej úrovni
- manažér zodpovedný za spúšťanie skriptov a kontrolu behu jednotlivých služieb
- sada skriptov slúžiaca na obsluhu jednotlivých služieb, napríklad pripájanie diskov a nastavenie sieťových rozhraní. Práve jeden z týchto skriptov používam v praktickej časti
- databáza udržiavajúca konfiguráciu, ktorá je upravovaná na to určenými nástrojmi. Tieto nástroje tiež kontrolujú syntaktickú správnosť konfigurácie

Kedysi bol distrubovaný ako kompletné riešenie, časom sa však jednotlivé komponenty oddelili a je ich možné využiť aj v kombinácii s inými nástrojmi. V praktickej časti som použil manažéra procesov, ktorý vznikol vďaka tomuto projektu a ovládacie skripty.

3.2 Red Hat Cluster Suite

RHCS, najnovšie premenovaný na High Availability Add-On pokrýva dva rôzne kategórie vysokej dostupnosti, failover a IP load-balancing (pôvodne nazývaný Piranha). Písmeno „R“ v RHCS naznačuje, že je dostupný len v RHEL, avšak nie je to tak. Balík je dostupný napríklad v CentOSe (prakticky RHEL bez licencie), Fedore alebo Ubuntu.

Je rovnako ako Linux-HA zložený z komponent, ktoré sa podľa zamerania dajú rozdeliť do štyroch celkov, zabezpečujúcich infraštruktúru, manažment služieb, administráciu a load-balancing [31]. Avšak predpoklad, že Red Hat všetky tieto komponenty sám vyvíja sa ukázal ako chybný. Niektoré z nich totižto využívajú open-source produkty, napríklad Corosync, LVS a plánuje aj prechod na Pacemaker. Pri inštalácii GFS2 v praktickej časti sa ako jedna zo závislostí preberá napríklad CMAN, ktorý je komponentou RHCS.

3.3 Linux Virtual Server

Linux Virtual Server pristupuje k problému vysokej dostupnosti iným spôsobom ako Linux-HA a RHCS. Nerieši ho na úrovni jednotlivých serverov v klastri, namiesto toho využíva load-balancer, na ktorý sa klienti pripájajú. Ten rozdeľuje požiadavky medzi servery. Load-balancer má tiež za úlohu periodicky kontrolovať dostupnosť serverov, na ktoré požiadavky posiela a v prípade že niektorý z nich neodpovedá tak ho prestane používať. Keď server opäť začne odpovedať začne ho opäť používať. Takýmto spôsobom efektívne maskuje nedostupnosť serverov. Tiež z pohľadu administrátora je jednoduché pridať ďalší server do klastra bez nutnosti reštartovať celý systém [32].

3.4 Alternatívy

Ako som načrtol v úvode kapitoly, neexistuje jedno univerzálne riešenie, je ich veľa a líšia sa ako cenou tak určením. Za zmienku určite stoja:

Solaris MC je operačný systém pre počítače v klastri. Umožňuje skupine serverov vystupovať ako jeden výkonný počítač. Známy je jednoduchou administráciou. Software zabezpečujúci jeho vysokú dostupnosť sa nazýva Sun Cluster

TurboLinux High Availability Cluster obsahuje viacero komponentov zabezpečujú-

cich load-balancing, škálovateľnosť a vysokú dostupnosť. Virtualizuje viacero nezávislých serverov do spoločnej siete vystupujúcej pod jednou IP adresou

Steeleye Lifekeeper umožňuje klientom použitie vlastných skriptov, ktoré budú kompatibilné s Microsoft Cluster Suite. To však prichádza s obmedzením použitia buď Lifekeeperu alebo MCS, nie oboch zároveň

Veritas Cluster Server je produkt Symantecu fungujúci na Linuxe aj Windowse umožňujúci failover v prípade výpadku

Microsoft Cluster Service nájdeme v serverovej edícii Windows. Tam je jedným z troch komponentov zabezpečujúcich klastrovanie. Ďalšími sú Network Load Balancing a Component Load Balancing

UCARP je jednoduchý nástroj umožňujúci niekoľkým hostiteľom zdieľať IP adresu za účelom automatického failoveru

Ďalšie podobné produkty zahŕňajú napríklad Fujitsu PrimeCluster, HP Serviceguard, IBM PowerHA, NEC ExpressCluster, Oracle Clusterware alebo SUSE Linux Enterprise HA Extension.

Kapitola 4

Realizácia

Pôvodný návrh tejto práce počítal s realizáciou a testovaním celého riešenia na reálnych serveroch. Tie sa mi ale v čase písania práce nepodarilo zabezpečiť, takže praktická časť prebehla na virtuálnych serveroch. Tým sú podstatne ovplyvnené niektoré z testov, avšak verím, že aj pomocou tohoto dokumentu bude možné riešenie veľmi jednoducho realizovať a otestovať aj v reálnom prostredí.

Virtualizačné riešenie, ktoré som použil je trial verzia VMware Workstation. Vybral som ho z dôvodu podpory viacerých snapshotov a dobrou integráciou s Windows 7, ako aj preto že som ho v minulosti používal a mám s ním dobré skúsenosti.

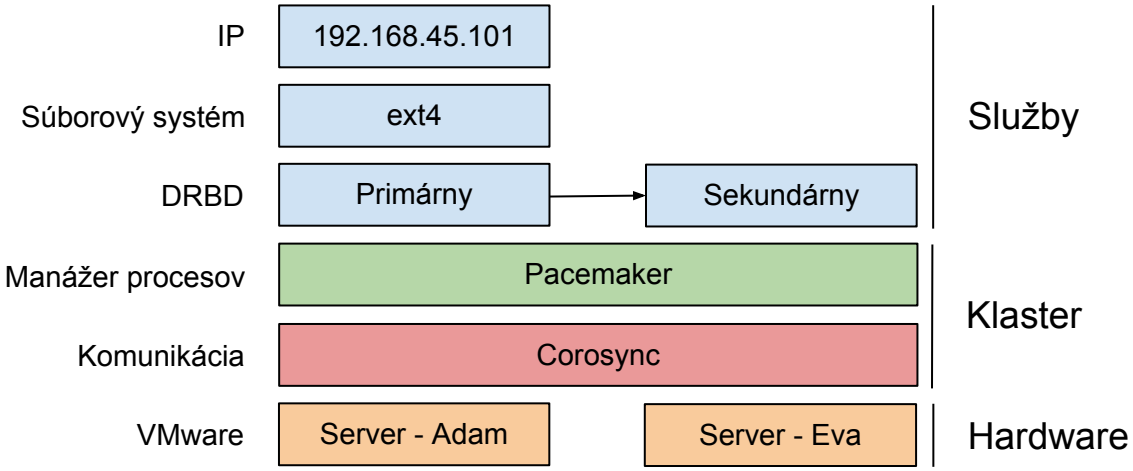
V ňom som vytvoril 2 virtuálne stroje Adama a Evu. Ich hardwarová konfigurácia je rovnaká, avšak v reálnom nasadení to nie je vyžadované. Jednotlivé komponenty nájdeme v tabuľke 4.1.

Komponent	Parametre
Operačná pamäť	1 GB
Pevný disk	20 GB SCSI
Procesor	1 jednojadrový procesor
Sieťová karta 1	NAT pripojený na internet
Sieťová karta 2	Host-only rozhranie
CD/DVD	Pripojený iso obraz OS

Tabuľka 4.1: Parametre virtuálnych strojov

Ako je už v úvode spomenuté, pokúšam sa nakonfigurovať riešenie pre malé firmy alebo

domáce použitie, takže je nevyhnutné brať na zreteľ cenu. Preto som vyberal technológie, ktoré sú dostupné zdarma. Hardwarové nároky použitého systému sú tiež minimálne. Schému celého riešenia, ktoré nakonfigurujem vidíme na obrázku 4.1.



Obr. 4.1: Schéma finálneho riešenia klastra

4.1 Operačný systém

Prvé kolo rozhodovania bolo relatívne ľahké. Windows alebo Linux? Windows je síce rozšíreným operačným systémom, ale je licencovaný. Taktiež riešenia dostupné zadarmo sú väčšinou produkované komunitou, ktorá je sústredenejšia okolo Linuxu.

Druhým kolom bolo vybrať tú správnu distribúciu Linuxu. Existuje ich ale veľké množstvo, tak ktorá je tá najlepšia? Každá distribúcia má svoje pre a proti, ja som si vybral serverovú edíciu Ubuntu z nasledujúcich dôvodov:

1. Ubuntu poznám. To síce nijako neopodstatňuje jeho použitie z profesionálneho hľadiska, avšak verím, že väčšina administrátorov uvažuje podobne. Prečo inštalovať neznáme riešenie na neznámy OS? Tiež v prípade, že firma už pre servery využíva konkrétny OS, nie je zvykom stažovať administrátorom prácu udržiavaním rozličných OS.

2. Ubuntu server sa vyvíja veľmi rýchlo, nové verzie sú vydávané každých 6 mesiacov. Jedným z cieľov práce je otestovať reálne riešenie, tak prečo nie práve na menej konzervatívnom systéme

3. Ubuntu je postavené na Debiane, ktorý obsahuje obrovské množstvo predkompilovaných balíčkov. Ubuntu túto základňu ešte viac rozširuje a novšie aplikácie sú k dispozícii oveľa skôr ako v prípade niektorých iných distribúcií

4. Ubuntu poskytuje mimo komunitných fór aj platenú podporu. Rovnako platenú podporu poskytuje napríklad Red Hat (podpora je na fórach hodnotená dosť slabo), avšak možno práve fakt, že Ubuntu je menej komerčný bude znamenať že táto podpora bude kvalitnejšia.

Mnou použité riešenie ale obdobne funguje aj na ostatných distribúciách. Najväčšie rozdiely budú pozorovateľné pravdepodobne pri samotnej inštalácii balíčkov a následnom hľadaní konfiguračných súborov. S výnimkou vlastného rozdelenia disku som pri inštalácii OS použil prednastavé hodnoty.

Vybral som si aktuálnu verziu Ubuntu 12.04. Všetok software som inštaloval pomocou správcu balíčkov apt zo štandardných repozitárov.

4.2 Dostupnosť dát

Počínajúc touto kapitolou sa začnem prakticky venovať vybraným technológiám z predchádzajúceho textu. Použitie RAIDu pre diskové pole je, ako som v sekcii 1.3.1 popísal, prvým krokom k dosiahnutiu vyššieho zabezpečenia dát. V prípade výpadku tak nie je nutné odstaviť celý server, stačí vymeniť chybný disk. Vo virtuálnych servroch som ale túto vrstvu vynechal, pretože sa chcem zaoberať predovšetkým vysokou dostupnosťou s ohľadom na prevenciu výpadkov akéhokoľvek komponentu, nie len pevného disku.

Použité 20 GB disky som rozdelil čo najjednoduchšie. Časť z nich som nechal nevyužitú kvôli možnostiam ďalšieho testovania. Jeho presné rozdelenie ukazuje tabuľka 4.2. Disky na oboch serveroch sú rozdelené rovnako. Pri reálnom nasadení nie je nutné rovnaké rozdelenie disku, avšak partície vyhradené pre DRBD musia mať rovnakú veľkosť.

Partície sda6 som použil ako podklad pre DRBD zariadenie. Jeho konfiguráciu popíšem v nasledujúcej kapitole.

Partícia	Bod pripojenia	FS	Veľkosť	Typ	Využitie
sda1	\root	ext4	7 GB	primárna	OS
sda5	swap	swap	1 GB	logická	swap
sda6	nevyužitá	-	5 GB	logická	testy FS s DRBD
sda7	nevyužitá	-	5 GB	logická	testy FS bez DRBD
-	voľné miesto	-	2 GB	-	rezerva

Tabuľka 4.2: Tabuľka rozdelenia disku

4.2.1 DRBD

Inštalácia nástrojov potrebných pre jeho správu prebehla bez problémov. Pri prvom použití bolo treba načítať modul jadra s názvom drbd. V niektorých distribúciách je potrebné tento modul nainštalovať, v mojom prípade ho už jadro obsahuje. Konfigurácia sa delí na 2 časti, globálnu a konfiguráciu samotnej DRBD partície.

Globálna časť umožňuje definovať správanie DRBD aplikácie. Definujeme tu napríklad požadované reakcie v prípade výpadku niektorého z diskov, timeouty, rýchlostné limity alebo protokol (synchrónny, asynchrónny) ktorý chceme použiť.

Partície definujeme tak, že špecifikujeme podkladové partície, ktoré má DRBD použiť, adresy servrov, na ktorých sa nachádzajú a názov zariadenia, ktoré má vytvoriť. Takýchto partícií môžeme definovať viacero.

Konfiguračné súbory sú na oboch servroch rovnaké. DRBD nezrkadlí konkrétne súbory a priečinky, pracuje len na úrovni blokového zariadenia ako je popísané v kapitole 2.3. Aby som mohol toto zariadenie využiť, musel som na ňom vytvoriť súborový systém. Práve jeho výberu sa budem venovať v ďalšej kapitole. Pre zrkadlenie dát som použil samostatnú sieťovú kartu, pretože aj keď sú výsledky testov skreslené v dôsledku virtualizácie, v reálnom nasadení je to odporúčané. Konfiguračný súbor je zobrazený vo výpise 4.1.

```
resource r0 {
  device    /dev/drbd0;
  disk      /dev/sda6;
  meta-disk internal;

  on adam {
```

```
    address    10.1.1.11:7789;
  }
  on eva {
    address    10.1.1.12:7789;
  }
}
```

Výpis 4.1: Konfiguračný súbor DRBD zariadenia

Prednastavený limit maximálnej rýchlosti pre synchronizáciu je vhodné upraviť pomocou konfiguračnej položky „rate“. Obmedzenie rýchlosti je využiteľné najmä v prípade, kedy existuje riziko zahltenia sieťovej karty pri inicializácii. Ďalšiu replikáciu dát tento limit neovplyvňuje.

4.2.2 Súborový systém

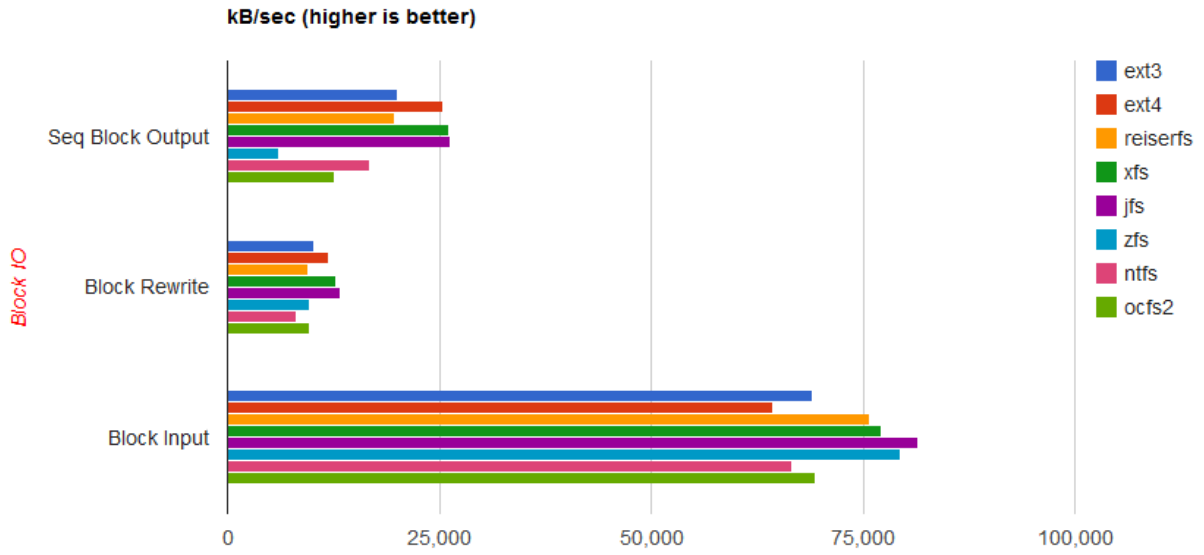
Súborových systémov je veľa, preto som sa zameral na tie najpoužívanejšie. Medzi testované som zahrnul ext3, ext4, reiserfs, zfs, jfs a xfs. Zo systémov so zdieľaným diskom som chcel v testoch zahrnúť OCFS2 a GFS2. GFS2 sa mi však v mojej konfigurácii nepodarilo spustiť kôli problémom v komunikácii klastra, spôsobenými pravdepodobne chybou medzi vrstvami Corosync a CMAN. Porovnanie ich výkonnosti je možné nájsť napríklad v dokumente [33].

Na testovanie súborových systémov som si vybral voľne dostupný nástroj bonnie++ verzie 1.96 a na prevod do grafickej podoby php skript bonnie2gchart. Test pozostával z dvoch častí, testu rýchlosti zápisu a čítania dát a testu počtu operácií s metadátami súborov, ktoré sa vykonajú za jednu sekundu. Tieto testy majú rôzne praktické využitie:

I/O Dáta Tento test je dôležitý - ak chceme súborový systém použiť na prácu s menším množstvom veľkých súborov. Vhodné využitie je napríklad pre ftp server. Test prepisovania je dôležitý v prípade, že nami využívané aplikácie často upravujú už existujúce dáta.

Metadáta Rýchlosť práce s metadátami je dôležitá v prípade práce s menšími súbormi. Pri rozbaľovaní archívu s veľkosťou 10 MB, ktorý obsahuje stovky súborov bude rýchlosť práce s metadátami zaujímavejšia ako rýchlosť zápisu na disk. Test je vhodný napríklad pre mailové servery, tmp partíciu alebo squid proxy.

Do testovania som pre zaujímavosť zahrnul aj ntfs partíciu, ktorá je štandardom na windowsoch. Testy práce s dátami dopadli vyrovnane ako je vidno na obrázku 4.2. Pri testovaní som sa snažil nezaťažovať hostiteľský OS nepotrebnými aplikáciami, avšak pri opätovnom spustení sa výsledky toho istého testu mierne líšili. Odchýlky však boli malé, na grafe sú znázornené výsledky jedného testu.

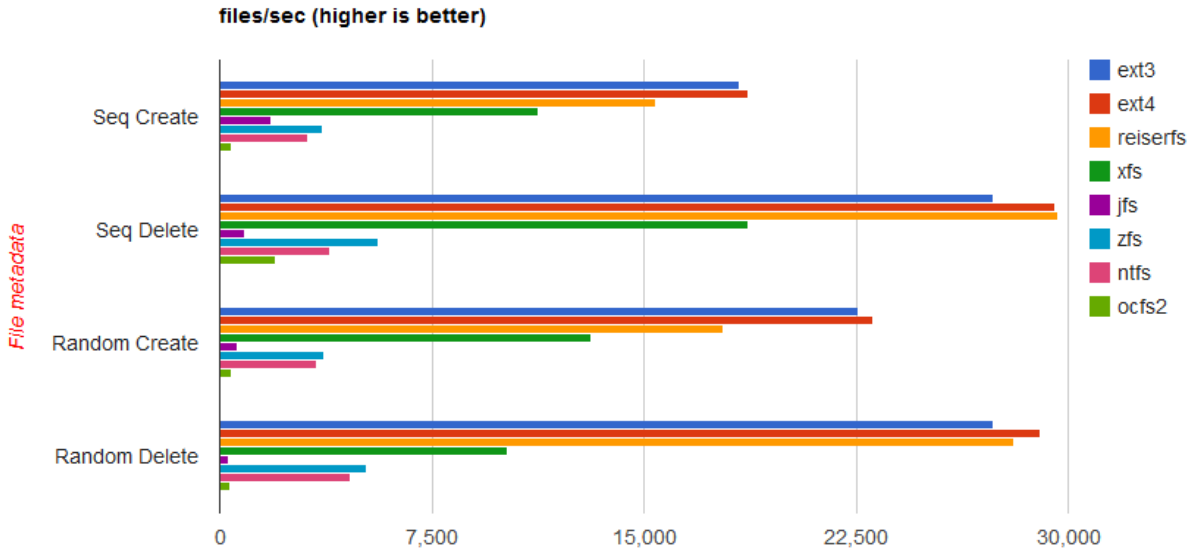


Obr. 4.2: Testy rýchlosti s použitím DRBD

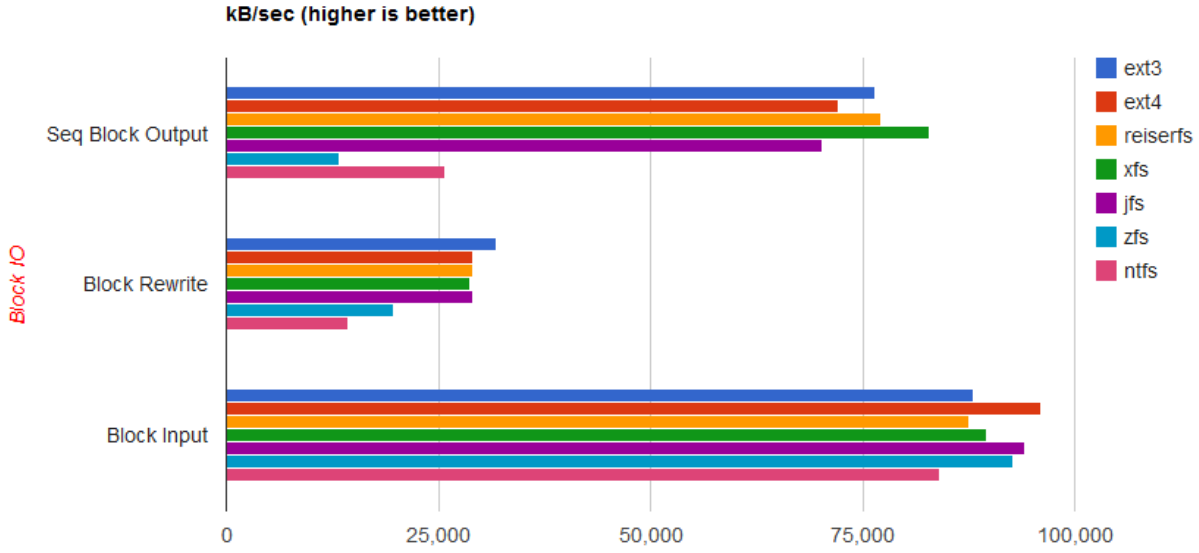
Pri výbere súborového systému zameraného na prácu s malými súbormi sa ako najvhodnejší kandidáti ukázali ext3, ext4 a reiserfs. Keďže ext4 je nasledovníkom ext3 a budúcnosť reiserfs bola istú dobu neistá, mojou voľbou by bol ext4. Pri vytváraní súborových systémov ma mierne zarazil fakt, že len niektoré z nich (xfs a jfs) vyžadujú potvrdenie pri prepísaní partície s už existujúcim súborovým systémom. Pritom malou chybou v čísle partície (sda6 vs sda7) pri jeho vytváraní môže administrátor zmazať všetky údaje uložené na danej partícii. NTFS nástroje partíciu dokonca bez varovania prepíšu nulami.

Pre porovnanie rýchlosti som tie isté testy zopakoval bez použitia DRBD zariadenia. Zápis dát prebehol dva až tri krát rýchlejšie, rýchlosť čítania dát je porovnateľná. Presné výsledky sú znázornené na obrázku 4.4. Musím ale pripomenúť, že testovacím prostredím je VMware, ktoré využíva jediný fyzický disk hostiteľského systému. Toto obmedzenie pravdaže vyplýva z hardwarovej konfigurácie môjho počítača. Preto napríklad zápis dát pri replikácii pomocou DRBD musel prebehnúť 2 krát na tom istom disku.

Výberom súborového systému a jeho inštaláciou na DRBD partíciu som dosiahol, že



Obr. 4.3: Testy rýchlosti práce s metadátami



Obr. 4.4: Testy rýchlosti bez použitia DRBD zariadenia

v prípade výpadku jedného zo strojov sú rovnaké dáta prístupné na druhom bez nutnosti obnovy zo zálohy alebo dodatočnej konfigurácie. Vytvoril som RAID-1 nezávislý na chybe v rozsahu servera. Vysoko dostupné dáta sú však bez aplikácií, ktoré ich sprístupnia užívateľom nepoužiteľné, preto v nasledujúcej časti predstavím konfiguráciu vysoko dostupného riešenia pre aplikácie.

4.3 Dostupnosť aplikácií

Pri realizácii tejto časti som sa rozhodoval, ktoré komponenty použiť. RGManager alebo Pacemaker? Corosync alebo Heartbeat? Zvolil som si cestu čo najjednoduchšieho riešenia s prihliadnutím na vyhliadky jednotlivých projektov. Corosync zabezpečuje časť funkcionality CMANu, potrebného pre RGManager. Pacemaker má časom ale RGManager nahradiť. Vybral som teda riešnie zostavené z čo najmenšieho počtu komponentov, ktoré vydržia čo najdlhšie. Corosync a Pacemaker.

4.3.1 Kominukačná vrstva

Pri inštalácii Corosyncu je potrebné vygenerovať zdieľaný kľúč, ktorý slúži na autentizáciu jednotlivých serverov a aby vedeli že do daného klastra patria. Jediným problémom, na ktorý som natrafil bola prednastavená hodnota „start=no“ v jednom z konfiguračných súborov. Corosync kvôli nej pokusy o štart ignoroval bez výpisu akýchkoľvek dodatočných informácií.

4.3.2 Manažér procesov

Ako manažéra procesov som použil Pacemaker. Konfiguroval som ho pomocou nástroja príkazového riadku crm, ktorý poskytuje rozhranie k samotnému xml súboru, v ktorom sú nastavenia uložené. Rovnaký výsledok sa dá dosiahnúť použitím testovaných GUI zmienených v kapitole 2.5. Pre názornú ukážku som použil drbd v móde primárny/sekundárny a na ňom vytvoril súborový systém ext4. Na aktívnom nóde bude prístupná IP adresa, ktorú môže využívať ľubovoľná služba.

Pre samotnú konfiguráciu bolo nutné nastaviť niekoľko pravidiel, ktorých reálny zápis možno vidieť vo výpise 4.2. Konfigurácia sa skladala z pravidiel definujúcich:

Primitívy ktoré definujú jednotlivé služby. V tejto konfigurácii sú použité 3 primitívy pre DRBD, súborový systém a IP adresu

Kolokácie definujú nutnosť spustenia služieb spoločne. V mojom prípade sú 2 a definujú že IP adresa môže byť spustená len na nóde s pripojeným súborovým systémom a ten bude pripojený vždy na primárnom nóde

Poradie hovorí ako z názvu vyplýva o poradí spustenia služieb. Súborový systém nemôže byť pripojený skôr ako DRBD zariadenie

Priľnavosť definuje ako veľmi chceme, aby služba ostala bežať na nóde, na ktorom je. V prípade že by som túto hodnotu nenastavil, služby by samovoľne migrovali podľa uváženia Pacemakeru

Vlastnosti definujú všeobecné správanie klastra. Ja som ich použil na zrušenia vynucovania vlastností Stonith a Quorum, ktoré pre potreby ukážky nie sú nevyhnutné, avšak v produkčnom nasadení sa na ne nesmie zabudnúť

```
root@eva:~# crm configure show
  Primitívy
    primitive ClusterIP ocf:heartbeat:IPaddr2 \
        params ip="192.168.45.101" cidr_netmask="24" \
        op monitor interval="30s"
    primitive DRBD ocf:linbit:drbd \
        params drbd_resource="r0"
    primitive fs_ext4 ocf:heartbeat:Filesystem \
        params device="/dev/drbd0" directory="/mnt" fstype="ext4" \
        meta target-role="Started"
  Kolokácie
    colocation drbd-with-ip inf: ClusterIP fs_ext4
    colocation fs_on_drbd inf: fs_ext4 msDRBDclone:Master
  Poradie
    order fs_ext4-after-DRBD inf: msDRBDclone:promote fs_ext4:start
  Vlastnosti
    property stonith-enabled="false" no-quorum-policy="ignore"
  Priľnavosť
    rsc_defaults $id="rsc-options" resource-stickiness="100"
```

Výpis 4.2: Čiastočný výpis konfigurácie crm

Konfigurácia sa pri zmene automaticky propaguje na ostatné nódy v klastri. Funkcionalitu riešenia v praxi názorne predvediem v ďalšej kapitole.

4.4 Čo som vytvoril

Výsledok práce predvediem na názornom príklade. Na začiatku tohoto testu sú oba servery online a všetky služby sú spustené. Test spočíva vo vypnutí primárneho serveru takzvane „natvrdo“ pomocou VMware. Pomocou systémových nástrojov (df, grep, crm_mon, cat) predvediem zmenu stavu častí systému, ktoré Pacemaker ovláda - DRBD disku, pripojenia súborového systému a spustenia IP adresy. Z výpisu niektorých príkazov som odstránil nedôležité detaily pre lepšiu prehľadnosť.

4.4.1 Pred výpadkom

Nástroj crm_mon slúži na zobrazenie stavu jednotlivých služieb, jeho výpis je na oboch serveroch identický. Služby sú teraz spustené na Adamovi. Parameter -1 slúži na jednorázový výpis stavu klastra. Vo výpise je vidieť zoznam nakonfigurovaných primitívov.

```
root@adam:~# crm_mon -1
  Online: [ adam eva ]
    ClusterIP       (ocf::heartbeat:IPaddr2):       Started adam
    Master/Slave Set: msDRBDclone [DRBD]
      Masters: [ adam ]
      Slaves: [ eva ]
    fs_ext4         (ocf::heartbeat:Filesystem):    Started adam
```

Nasledujúce príkazy dokazujú, že Adam je primárnym serverom a je na ňom pripojená DRBD partícia, zatiaľ čo sekundárny server je neaktívny.

```
root@adam:~# df -h | grep mnt
  /dev/drbd0       4.7G  198M  4.3G   5% /mnt
root@adam:~# cat /proc/drbd | grep cs
  0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----

root@eva:~# df -h | grep mnt
root@eva:~# cat /proc/drbd | grep cs
  0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r-----
```

4.4.2 Po výpadku

Po vypnutí primárneho serveru (Adam) sekundárny (Eva) detekuje jeho neprítomnosť a začne zapínať jednotlivé služby v nakonfigurovanom poradí. Po chvíli je z výpisu zrejmé, že sa všetky spustili na serveri Eva. Celý proces od detekcie po spustenie poslednej služby trval približne 5 sekúnd.

```
root@eva:~# crm_mon -1
  Online: [ eva ]
  OFFLINE: [ adam ]
    ClusterIP       (ocf::heartbeat:IPaddr2):       Started eva
    Master/Slave Set: msDRBDclone [DRBD]
      Masters: [ eva ]
      Stopped: [ DRBD:0 ]
    fs_ext4         (ocf::heartbeat:Filesystem):    Started eva
```

Keďže Adam je už vypnutý a Eva s ním nemá spojenie, je v DRBD výpise označený ako unknown. Pacemaker nastavil DRBD na Eve do stavu primary a pripojil súborový systém.

```
root@eva:~# cat /proc/drbd | grep cs
  0: cs:WFConnection ro:Primary/Unknown ds:UpToDate/DUnknown C r-----
root@eva:~# df -h | grep mnt
/dev/drbd0       4.7G  198M  4.3G   5% /mnt
```

V prípade, že sa dáta na primárnom DRBD zariadení počas nedostupnosti druhého nódu zmenia, je po jeho opätovnom pripojení automaticky inicializovaná synchronizácia dát. Kopírujú sa len zmenená bloky, nie celé zariadenie ako pri inicializácii DRBD. Výpis znázorňuje priebeh synchronizácie.

```
root@adam:/mnt# cat /proc/drbd
  0: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-----
        [===========>.....] sync'ed: 60.0% (8872/20188)K
```

```
finish: 0:00:00 speed: 11,316 (11,316) K/sec
```

Záver

Na začiatku písania práce som si stanovil 1 teoretický a 2 praktické ciele. Cieľom teoretickej časti bolo uviesť čitateľa do problematiky vysokej dostupnosti aplikácií a dátových úložísk. V praktickej časti som jedno z týchto riešení realizoval a porovnal rýchlosti a vhodnosť rôznych súborových systémov použiteľných v danej konfigurácii.

Výsledky testov značne ovplyvnila hardwarová konfigurácia, na ktorej som ich realizoval. Pri použití jediného fyzického disku sa veľké rozdiely v rýchlosti práce s veľkými blokmi dát neprejavili. Značné rozdiely však ukázali testy práce s metadátami súborov, kedy súborové systémy ext3, ext4 a reiserfs dosahovali mnohonásobne vyššie rýchlosti ako jfs, zfs, ntfs a ocfs2.

Riešenie som realizoval pomocou DRBD pre zabezečenie vysokej dostupnosti dát a Pacemakeru pre aplikácie. Obe tieto nástroje sú voľne dostupné. Vo svojej práci Tomáš Zvala zmieňuje nezrelosť voľne dostupných projektov, avšak ja som pri inštalácii a konfigurácii nenarazil na žiadne zásadné problémy. Finálna konfigurácia pracuje podľa očakávaní.

Literatúra

[1] T. Zvala, "Zajištění vysoké dostupnosti dat u virtuálních serverů," Master's thesis, Vysoká škola ekonomická v Praze, 2009.

[2] R. Zima, "Zajištění dostupnosti it infrastruktury pro podporu firemních procesů," Master's thesis, Vysoká škola ekonomická v Praze, 2004.

[3] P. M. Encyclopedia, "High availability definition." http://www.pcmag.com/encyclopedia_term/0,1233,t=high+availability&i=44246,00.asp.

[4] I. Constant Data, "Managing the cost of downtime." http://costkiller.net/tribune/Tribu-PDF/MANAGING-THE-COSTS-OF-DOWNTIME.pdf, 2004.

[5] K. Schmidt, *High Availability and Disaster Recovery*. Springer, 2006.

[6] E. Marcus and H. Stern, *Blueprints for High Availability Second Edition*. Wiley Publishing, 2003.

[7] S. Traugott, "Disaster recovery." http://www.infrastructures.org/bootstrap/recovery.shtml.

[8] V. Gite, "Software vs hardware raid." http://www.cyberciti.biz/tips/raid-hardware-vs-raid-software.html, 2009.

[9] M. Stopka, "Storage area network – 1 (úvod)." http://www.abclinuxu.cz/clanky/storage-area-network-1-uvod, 2010.

[10] S. Kline, "Difference between nas and san - 3 considerations." http://www.turbotekcomputer.com/resources/small-business-it-blog/bid/58074/Difference-Between-NAS-and-SAN-3-Considerations, 2011.

[11] P. M. Encyclopedia, "Failover definition." http://www.pcmag.com/encyclopedia_term/0,1233,t=failover&i=42980,00.asp.

[12] S. Kahlouch, "Heartbeat vs openais." http://answerpot.com/showthread.php?124007-Heartbeat+vs+OpenAIS, 2010.

[13] Beekhof, "Faq - clusterlabs." http://clusterlabs.org/wiki/FAQ#Organizational, 2010.

[14] Beekhof, "Linux-ha." http://linux-ha.org/wiki/Main_Page, 2011.

[15] OpenAIS, "Dokumentácia openais." http://www.openais.org/doku.php.

[16] C. Caulfield, "Whatever happened to cman?." http://people.redhat.com/ccaulfie/docs/Whither%20cman.pdf, 2009.

[17] A. Beekhof, "Documentation for cluster beginner and pacemaker vs rgmanager." http://www.redhat.com/archives/linux-cluster/2011-May/msg00023.html, 2011.

[18] Linbit, "Drbd dokumentácia." http://www.drbd.org/.

[19] S. Bigelow, "iscsi vs. fibre channel explained." http://www.cuttedge.com/files/iscsi_vs_fiberchannel_explain.pdf, 2007.

[20] T. Weimer, "Fibre channel fundamentals." http://www.unylogix.com/data_storage/raid_san/PDFs/White_Paper_Fibre_Channel_Fundamentals.pdf.

[21] S. Whitehouse, "Cluster suite overview." http://kernel.org/doc/ols/2007/ols2007v2-pages-253-260.pdf, 2007.

[22] S. Mushran, "Ocfs2 a cluster file system for linux." http://oss.oracle.com/projects/ocfs2/dist/documentation/v1.6/ocfs2-1_6-usersguide.pdf, 2010.

[23] E. Levy and A. Silberschatz, "Distributed file systems: Concepts and examples." http://courses.cs.vt.edu/cs5204/fall02/Papers/FileSystems/DistributedFileSystemsSurvey.pdf, 1990.

[24] I. Gluster, "Introduction to gluster." http://www.think88.com/Examples/Introduction_to_Gluster.pdf, 2010.

[25] F. Wang, S. Oral, G. Shipman, O. Drokin, T. Wang, and I. Huang, "Understanding lustre filesystem internals." `http://wiki.lustre.org/images/d/da/Understanding_Lustre_Filesystem_Internals.pdf`, 2009.

[26] whamcloud, "New to lustre?." `http://www.whamcloud.com/lustre/support/new-to-lustre/`.

[27] A. Robertson, "Split-brain, quorum, and fencing - updated." `http://techthoughts.typepad.com/managing_computers/2007/10/split-brain-quo.html`, 2007.

[28] F. ČVUT, "Crm (resource agents)." `https://support.dce.felk.cvut.cz/mediawiki/index.php/CRM_(Resource_Agents)`, 2011.

[29] OpenSuse, "Fencing and stonith." `http://doc.opensuse.org/products/draft/SLE-HA/SLE-ha-guide_sd_draft/cha.ha.fencing.html`.

[30] H. Y. Youn, C. Yu, D. S. Han, and D. Lee, "The approaches for high available and fault-tolerant cluster systems."

[31] R. Hat, "Cluster suite overview." `http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Cluster_Suite_Overview/index.html`, 2010.

[32] LVS, "Linux virtual server." `http://www.linuxvirtualserver.org/`.

[33] G. Paternò, "Filesystem comparison nfs, gfs2, ocfs2." `http://www.gpaterno.com/publications/2010/dublin_ossbarcamp_2010_fs_comparison.pdf`, 2010.

Zoznam obrázkov

Zoznam tabuliek